

Jfactor

Yngvi Rafn Yngvason

Supervisor: Mark O'Brien

Faculty of Information Technology,
University of Akureyri.

Submitted April 2004, in partial fulfilment of
the conditions of the award of the degree BSc.

I hereby declare that this dissertation is all my own work,
except as indicated in the text:

Signature _____

Date 14/4/2004

Abstract

This dissertation describes the analysis, design, implementation and evaluation of the Jfactor project. Samhjálp, the organisation that will be using this system, is a drug rehabilitation organisation which has its offices located in Reykjavík and the main treatment centre and other facilities dispersed around the Reykjavík area. The environmental restrictions set to this system are that it must run in a closed environment yet have the flexibility of a distributed system since Samhjálp is geographically distributed organisation. This system will contain highly sensitive personal information and therefore requires high security measures. This system will, when completed, be able to maintain and analyse data regarding the clients receiving treatment at this treatment centre giving the organisation a clear picture on the situation of both clients and its staff. With some modification i.e. translation to native languages, this system can be used in any treatment centre in the world. Number of clients using this system is not limited. However, it will be designed to serve at least fifty users simultaneously to begin with. The goal of the system is to present an overall picture of the situation of the drug world in each country. Furthermore it will be possible to combine statistical data from all over the world and thus creating a picture of drug abuse and other addictive behaviours in various parts of the world. This project will however only contain the first module of the Jfactor system, the registration module of staff members and client information.

Table of contents

1	Motivation for the work	6
1.1	Organisation overview	6
1.2	The need	7
2	Description of the work.....	8
2.1	Analysis process	8
2.1.1	HLK.....	8
2.1.2	Other services	9
2.2	System overview	9
2.3	Background reading	9
3	Related work	9
4	Design	10
4.1	Security design	10
4.1.1	Virtual Private Network (VPN)	10
4.1.2	Transmission Control Protocol (TCP) wrappers.....	11
4.1.3	Internet Protocol (IP) tables.....	11
4.1.4	Firewall unit.....	11
4.1.5	Intrusion Detection system (IDS)	11
4.1.6	Authentication	11
4.2	Interface design	12
4.3	Application design.....	13
4.3.1	Presentation layer (webfiles directory)	14
4.3.2	Domain layer	15
4.3.2.1	Service layer.....	15
4.3.2.2	Active Records.....	15
4.3.3	Database layer.....	16
4.4	Database design.....	17
5	Implementation	19
5.1	Resources	19
5.2	Problems encountered	20
5.3	Technology used.....	20
5.3.1	Hyper Text Transfer Protocol (HTTP)	21
5.3.2	Hyper Text Mark-up Language (HTML)	22
5.3.3	Java Server Pages (JSP).....	22
5.3.3.1	JSP Elements.....	23
5.3.3.2	Navigation and Control.....	25
5.3.4	Java Servlets	26

5.3.5	Java Classes	27
5.3.6	SQL - Database Sublanguage	27
5.4	Work and dataflow summary of the application.....	28
5.4.1	Data insert.....	28
5.4.2	Data retrieval	28
5.5	Interface implementation.....	29
5.5.1	Preventing caching and storing of data.....	29
5.5.2	Login page.....	31
5.5.3	Adding staff members.....	31
5.6	Application implementation	32
5.6.1	Presentation layer (webfiles directory)	32
5.6.1.1	Checking the social security number (SSN or Kennitala).....	32
5.6.1.2	Staff JSP's.....	33
5.6.2	Service layer	33
5.6.3	Domain layer (Active Records)	33
5.6.3.1	Staff package.....	34
5.6.3.2	Accessories package.....	35
5.6.3.3	Client package.....	36
5.6.3.4	Justice facilities package	37
5.6.3.5	Other classes	38
5.6.3.6	Committing data to database	39
5.6.4	Database layer.....	39
5.7	Database implementation	39
5.7.1	Database connection	39
5.7.2	SQL queries	39
5.7.3	Database tables	40
5.7.4	Grants	40
5.7.5	Users table	40
6	Testing.....	40
6.1	Unit testing	40
6.2	Other types of testing performed	40
6.3	Viewing the JSP page	41
7	Evaluation	41
7.1	Project's quality.....	41
7.2	Project's future	42
7.3	Conclusion.....	43
8	References	43

9	Bibliography.....	44
10	Index.....	45
11	Glossary	46
	Appendix A Software Requirements Specification.....	47
	Appendix B Analysis documents	73
	Appendix C System Design.....	87
	Appendix D Code listing.....	93
	Digital copy of code	190

Table of figures

Figure 1: Virtual Private Network (VPN) in action.....	11
Figure 2: The login window to the Jfactor system.....	12
Figure 3: The top part of the treatment manager's interface.	13
Figure 4: Package diagram of the Jfactor application.....	14
Figure 5: Sequence diagram of the data flow in Jfactor from browser to database and back.....	15
Figure 6: Dependencies between sub-packages and the layer above and below.	16
Figure 7: The current database design for the Jfactor application.	18
Figure 8: A diagram showing how different technologies flow together.	21
Figure 9: The JSP processing and translation structure (Bergsten, H.).	23
Figure 10: Administrators interface. Adding new staff member and login information.....	31

1 Motivation for the work

Samhjálp's rehabilitation centre is an extraordinary organisation. The treatment is returning the highest percentage of productive individuals back into society of any addict treatments available in Iceland. So knowing, that all client records and other paperwork are mostly done using pen and paper, it seemed obvious that the organisation is in desperate need for a computerized system. When talking to the staff it came very clear that the need was there indeed.

1.1 Organisation overview

Samhjálp is a Christian organisation owned by the Pentecostal church in Iceland. It was founded in 1973. Since then it has grown significantly and flourished. Samhjálp is not only helping alcoholics and drug abusers. Clients that are addictive to gambling, pornography or any other type of addiction come also for treatment. Samhjálp's finances are combined with government funds and individual contributions. Many volunteers participate in various tasks.

All these thirty years of operation all files and information on clients have been paper based. No software has been built to handle the tasks that need to be computerised. Only financial and standard office applications are available. This has resulted in high frequency of repetition by the staff members, especially those who are directly linked to the treatment centre at Hlaðgerðarkot (HLK). This repetitive work involves writing the same information over and over again like identity and medication information.

Presently there are 27 staff members and two contractors working for Samhjálp. Some of the staff members are part time. Those who will be the users of the system, when fully expanded, are counsellors, treatment managers, administrators, doctor, pharmacist and watchmen.

Samhjálp is a geographically distributed organisation and thus consists of several organisational units. Today those units are the following:

- *Headquarters* – located at Hverfisgata 42, Reykjavik, where all the offices and some of the counselling facilities are.
- *Hlaðgerðarkot* – the main rehabilitation centre, located outside of Reykjavik in Mosfellsdalur. (*see picture to the right*)
- *Stoðbylið* (Support shelter) located at Hverfisgata 42, Reykjavik, where clients can stay after treatment or are in continued therapy (called challenge) while establishing a healthy foundation before returning back into society.



-
- *M-18 and M-20* are homes owned by the Reykjavik Social Services (RSS) but run by Samhjálp.
 - *Cafeteria* located at Hverfisgata 44, Reykjavik, where homeless and poor people come for a free meal.

See Appendix B for further reference.

At HLK where this part of the project will be used to start with, are beds for 30 individuals in treatment. In the support shelter are 14 individuals and 16 in total at M-18 and M-20. The cafeteria's daily visits are from 40 up to 100 individuals that come for a free meal every day. So Samhjálp in total is helping 60 individuals on a fixed daily basis and up to 100 others through food distribution and fellowship.

For detailed information on the activities and processes see Appendix B.

1.2 The need

The administrators need a system that allows the ability to seek for behavioural patterns among clients and monitor the counsellor's success to be able to backup the counsellors when needed. They also wanted the ability to monitor prices of the abused drugs, be able to see proportionally how many are abusing each drug type and the quantity, and last but not least to see the client's financial, social, health, housing and family situations and to be able to seek for patterns within the given criteria.

Counsellors need a system that would relieve them from the repetitive process of writing the same personal information and enable them to enter client's information in a secure and efficient way along with the possibility to view client's former treatments and their addictive history.

Treatment managers need a system that in addition would enable them to calculate the number of clients per month, number of applications per month, used beds per month, number of males and females etc. They also need a system that could handle the distribution of medications to clients as well as entering client's personal information when arriving at the treatment centre.

The doctor needs a system to record information on the client's health and to write medication prescriptions for the clients to handle other diseases they might have.

The pharmacist needs to access those prescriptions to process them.

Watchmen need a system to register all phone calls and visits to the treatment centre.

These factors and the fact that I know many of the staff and some former clients personally, gave me the motivation needed to design a system that will in the future, fulfil these needs. However, the part that is being implemented in this version of the software and described in this dissertation is the registration module where administrators, supervisors and counsellors can record personal information regarding staff members and clients into the database.

2 Description of the work

This final year project is only the starting point in developing Jfactor. This software will take at least two more years to develop, assuming I will be working on it with an access to a counsellor that has good programming experience. In that time the system should be functioning with most of the requested features. The complete set of requirements is found in Appendix A – Software Requirements Specification.

But why this name, Jfactor? It is derived from the phrase ‘Jesus factor’ which came up when the government of the United States of America did a research on all addict rehabilitation centres in the country. They found no implication of difference between treatment centres, except for one, in the Christian treatment centres. This difference got named the Jesus factor since no other explanation could be found than the belief in Jesus Christ was doing more for clients than anything else in the treatment. Let us start by give a little background on the organisation and an overview of the analysis process.

2.1 Analysis process

When this project started last summer the becoming users of the system were interviewed to gain understanding of the processes and the staff’s workflow. The staff was very helpful in the analysis process. They showed a lot of interest in the system but this interest varies in some respect. The chief administrator sees among other things the opportunity for data mining and pattern seeking. Such information would be extremely helpful but will have to wait until later.

2.1.1 HLK

The treatment managers see the convenience in calculating number of clients per month, how many beds were used last month and number of women and men were in treatment for the last month. They also need to see the number of applications for each individual on a given time period. The system needs to keep track of all medication given to clients in ‘detox’ (A procedure that is also known as a ‘step down’). This includes medication that clients need for handling other diseases. The treatment managers need the system to be able to print out various forms both for government, client and individual purposes. They are the ones that welcome new clients for treatment and therefore need to insert some information about the client plus assigning a counsellor to the client.

The counsellor will then continue with that work and fill in the gaps plus adding in detailed personal information regarding the client’s situation and possible solutions to his crisis.

The doctor would enter his notes about the client during or after examination and be able to prescribe medication.

The pharmacist will be able to access those prescriptions, prepare the order and notify the treatment manager who will distribute the medication to the clients.

The watchmen, who are ‘trusted’ clients in treatment, answer the phone and record the arrival and departure of every visitor that comes to the premises. They write down every caller and visitor plus every application for a treatment. For more background reading see detailed description of the HLK processes in Appendix B.

2.1.2 Other services

Samhjálp’s task is not bound by HLK only. There are several other branches on the tree. Among them are M-18 and M-20 that are homes owned by the Reykjavík Social Services (RSS) but run by Samhjálp. Other services are a cafeteria where homeless and poor people come for a free meal and support shelter where clients can stay after treatment or are in continued therapy (called challenge) while establishing a healthy foundation before returning back into society. See Appendix B for further reference.

2.2 System overview

This system is web-based. The server will be located at Samhjálp’s headquarters. The organisation is distributed so the need for a centralised system with a multi client access is needed. This system is meant to achieve the recording of all necessary information about the clients such that information fed into the system will give the maximum benefit for both the client’s treatment and staff working environment. The nature of the information entered is very sensitive and personal, yielding high security measures to protect the information at all costs. These measures will, however, not be implemented until the system will be set up at Samhjálp. The system’s user interface will be designed using ergonomics guidelines. The user interface will use both HTML and JSP pages to interact with the system. JSP and servlets will check the data and use HTTP requests and responses to communicate with the browser along with creating and unpacking objects that are sent and received towards the database. The Domain layer will handle all the database interaction using connections created by the Database layer. The connections are created using a standard JDBC driver. This will in the future be changed by implementing connection pooling.

2.3 Background reading

When designing and building a system like Jfactor a number of issues need to be addressed. Therefore a background reading was performed on various databases, servers and web technologies. These issues will be addressed later. Now let us look at other existing systems.

3 Related work

A search of the Internet was made for similar applications. None were found. It is more than likely that someone has made a system that is intended to do something similar. No software was located that could satisfy Samhjálp needs. There is one application that is being used by SÁÁ which is Iceland’s largest addict

rehab centre. They are using the SAGA system, written by eMR, which has been implemented in all major hospitals in Iceland. The feedback from users is that the health service version is quite good but the hospital version is totally not meeting the hospital's need. The SÁA is using the health service version and seems to be working satisfactorily. However, the SÁA management has already knocked on eMR's door for radical changes to the system.

Technically, this project is not introducing any new technology. However, this already and well known existent technology has not been used in this manner to my knowledge. Websites with a lot of information on alcoholism, drug abuse, chat channels, and text material on ways to tackle the addiction itself is widely available but no web based system seems to be available that is handling client records and retrieve the information in this way.

Regarding the prescription part of the program, it might be skipped since there is already Icelandic software for that purpose. This software called ePref is a standalone application where doctors can prescribe medication safely. This software is developed by a company called Doc.

4 Design

A web based system as a design for this project not only fulfils the needs for the organisation but also makes it portable and easy to set up anywhere. As seen in section 2.1 the organisation is geographically dispersed. This fact makes it possible to create a closed system by the use of Virtual Private Network (VPN) and assigning the Media Access Control (MAC) addresses of only those computers allowed to use the system. But let us take a closer and more detailed look at the system design.

4.1 Security design

The security issues for the Jfactor system are mandatory, yet most of them will not be implemented until the system will be installed at Samhjálp headquarters. Avoiding using the Secure Socket Layer (SSL) is desirable since it slows the system down because it is a computer power intensive feature. But if health authorities demand it there will not be a problem to implement that feature. There are ways of compressing the data packets, hence increasing significantly the efficiency of the SSL technology. This security design as shown below may sound like a paranoid and hysterical cry, but it is my belief that as many as possible of the issues mentioned in this chapter should be implemented (without compromising performance) for securing such sensitive personal information.

4.1.1 Virtual Private Network (VPN)

By using VPN the system will be totally closed. It is my recommendation to use Cisco VPN routers which are safer than the Windows software version. However, the Windows version could be used to start with.

The VPN works in the following way:

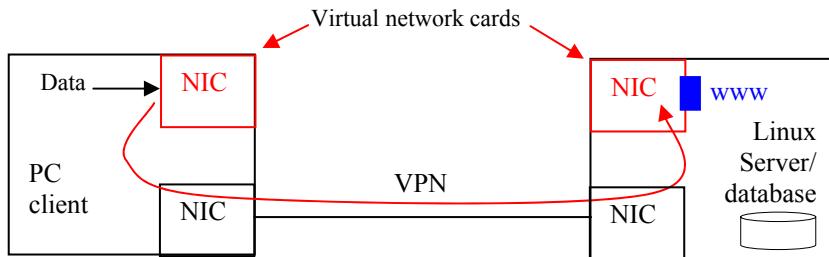


Figure 1: Virtual Private Network (VPN) in action.

VPN creates a virtual network card in memory and all traffic is directed through this ‘card’ where it is encrypted. Then the traffic is routed to the real network card, and encapsulated with another encryption layer. Encryption level is configurable. The algorithm probably used for encryption is the 3DES, 168-bit algorithm. It encrypts each packet three times before sending it. On the other end the packet is decrypted accordingly.

4.1.2 Transmission Control Protocol (TCP) wrappers

The TCP Wrapper program in the Linux operating system is one of the most widely spread security tool in existence. It works like a firewall in that it accepts or rejects network connections based on IP addresses and service types (Wreski 1999). This feature will be implemented when the system is installed at the Samhjálp’s headquarters.

4.1.3 Internet Protocol (IP) tables

An IP table is a program in Linux that acts as a firewall. There we can make choices on how incoming or outgoing packets are handled based on origin, destination, user, and packet or protocol type. Are they accepted, dropped, forwarded or logged? The only IP addresses that are registered in the IP tables will be granted access to the system (Andreasson 2001-2003). These addresses are the one associated with the Samhjálp’s computers. Using IP tables with TCP wrappers will give us a twofold firewall.

4.1.4 Firewall unit

One option is to add a hardware firewall as the third firewall to ensure even better the security.

4.1.5 Intrusion Detection system (IDS)

The Linux community has created many excellent open source software features. One of them is a program that monitors all packets that are sent to the server and detects all intrusion attempts and shuts them off. Such intrusion detection will be implemented when the system will be set up.

4.1.6 Authentication

When members of staff access the system they first have to log into the VPN network. This is only the first step. The next step is to log into the Jfactor system it self. When opening the browser and accessing the

Jfactor's login page, the user is asked for both user name and a password. When accepted the session will start a timing of every idle time period. If such period exceeds, say ten minutes, the session will close and user has to login again. When logging in the authentication servlet will forward the username and password to the classes handling the staff and if it gets a positive response it will request for the appropriate data for the user and pass it back to the user in HTML format.



Figure 2: The login window to the Jfactor system.

This is necessary since each staff member has its own responsibilities and will only access the data that is on need to know basis. If however the username and/or password can not be found in the database the user is notified and given two more tries to log successfully into the system. If these three tries fail the user will be directed to Samhjálp's homepage and administrator is notified about which MAC address has tried to access the system with wrong information.

4.2 Interface design

The interface is designed to meet the standard ergonomics guidelines regarding user – computer interface design. The graphical design is intended to be comfortable in daily use as in its look and feel. Maintaining a smooth dialogue is also very important such that the work flow will not be interrupted by a badly designed layout. The dialogue helps the user to concentrate on the client such that the gathered information is as precise as possible and that the staff member does not have to search for the next thing he needs to fill in or the button to press. An example of 'explicit codes' used in the pages is the tabs seen in the header below. The flow is from the left to right and top to bottom. This design is true both for the tabs at the top of the interface as well as the content and form pages accessed by the staff member. An example of this is the counsellor and treatment manager pages where they insert the client data. These tabs are in the same order as the treatment process. First is the admission (Innlögn) secondly the first interview (Viðtal 1) then the second interview and then the progress interviews. Finally is the discharge tab (Útskrift) where the client is registered out of the treatment. This is shown in the figure below. Implicit codes in the header are the

Jfactor title which will take the user when clicked to his homepage and the Samhjálp logo which will take the user to the Samhjálp's homepage.



Figure 3: The top part of the treatment manager's interface.

The ergonomics guidelines used are the following:

- Let the users feel in control by providing them with simple and clear cut user interfaces
- Avoid information overloading.
- Using grants (roles) and log-on features gives the system a sense of “power” to the user.
- Processes are performed in left to right direction to aid logical progress of the processes.
- Use of explicit and implicit coding to aid the user’s understanding of what he can do on each page.
- Clear – easy to read fonts are used to prevent misreading.
- Colours used are mild to prevent distraction from the actual content of the interface.
- Task analysis used to decide the build up of the process steps.

4.3 Application design

There are many ways to design an application such as this one. However, considering the size, type and purpose of this application, I chose the popular three tiered approach in designing this system.

The reason for choosing this design is simply to separate the functionality and make it easier to make changes to the program. This web based database system is using three layers.

First to mention is the presentation layer which connects to a browser used by the client to access and work on the system. Secondly, it is the Domain layer which contains the business logic and thirdly it is the database layer that handles the connections to the database itself. This means that any number of users can access and use the system, given they are granted access. So Jfactor is a very extendable system. A clear separation between layers has been made such that each layer is not dependent on more than the layer below.

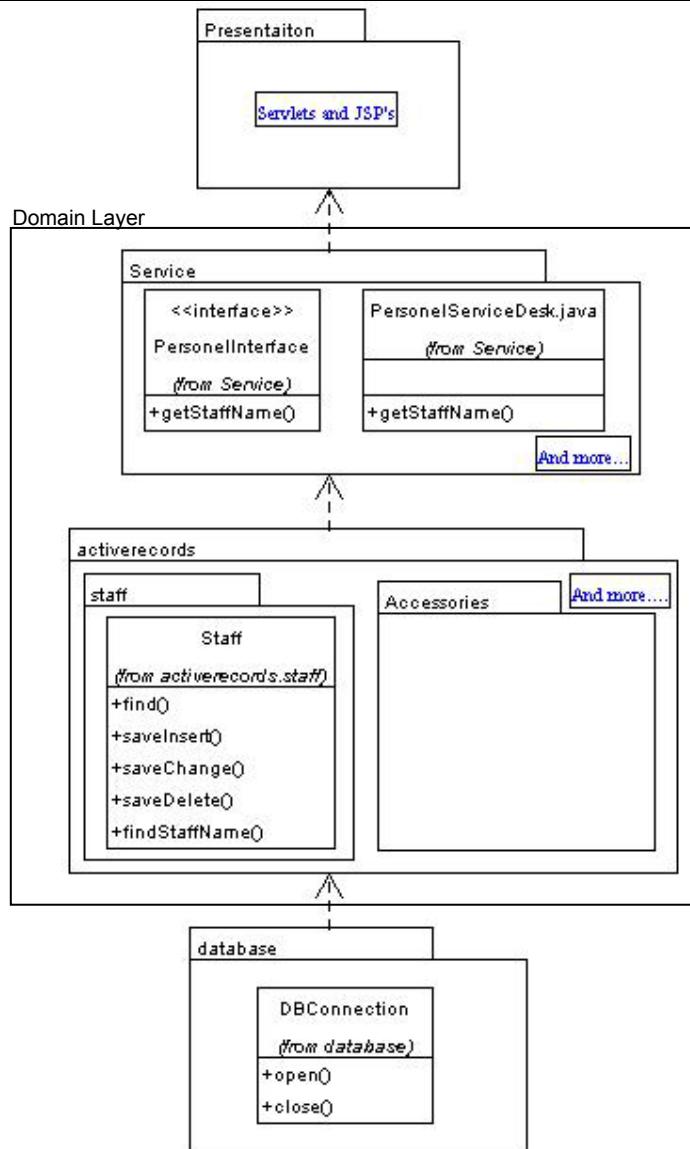


Figure 4: Package diagram of the Jfactor application.

4.3.1 Presentation layer (webfiles directory)

This layer contains all the servlets, JSP's, HTML's needed to receive the HTTP requests and transfer the data downwards to the service layer. As well as creating the pages needed to be created and sent to the user. All requests use the POST method such that any information entered or received by the browser will not be seen in the browsers address bar.

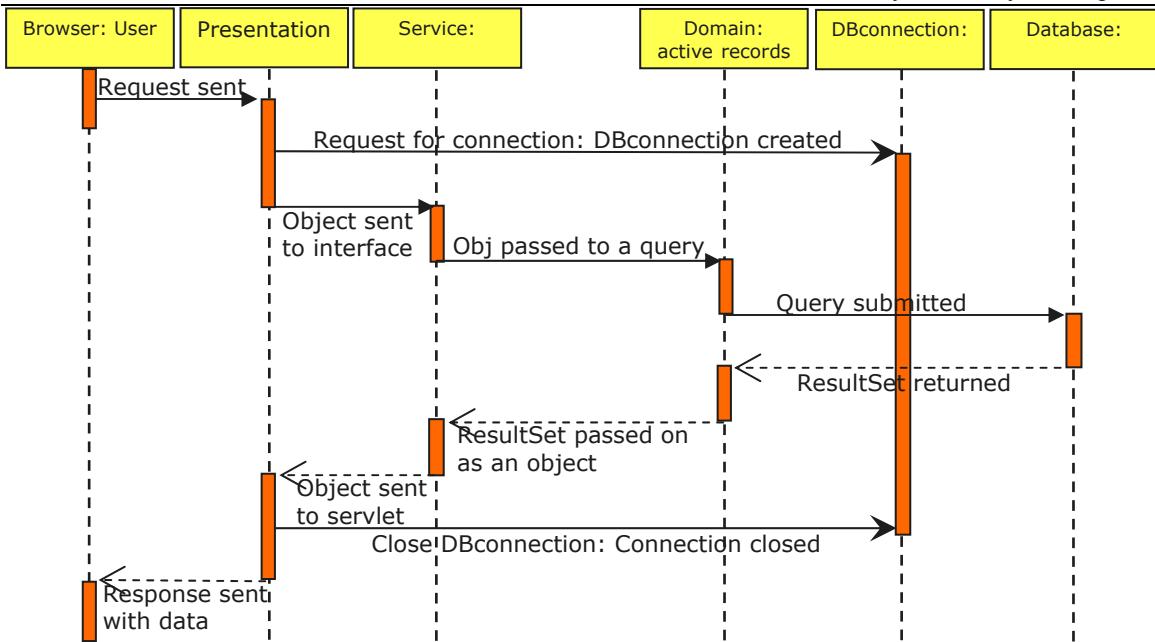


Figure 5: Sequence diagram of the data flow in Jfactor from browser to database and back.

4.3.2 Domain layer

The domain layer consists of two main packages. First it is the service layer and secondly it is the active records layer which contains several sub-packages that interact with the various parts of the system.

4.3.2.1 Service layer

The service layer contains interfaces to the service desk classes. These service desk classes contain the interactions needed to access the active records in the domain layer. The service layer acts as a receptionist in a company, directing the data in the right direction. The interfaces separate the domain layer from the presentation layer.

4.3.2.2 Active Records

Active Record is the Data Source Architectural Pattern used in the domain layer (Fowler 2003).

The classes are named similar names as the database tables they represent. These classes handle all the operation needed to insert, update, delete and query the database, returning a class object which is redirected towards the service layer. So in general the active records have similarities to Transaction Scripts which is so to speak a ‘do it all’ class. Active records, however, have restricted functionality in the way needed to handle all database activity close to the database itself. This makes it easier to manage applications of medium size while the Transaction Scripts become hard to handle when the program’s size grows significantly. See section 5.6.3 for detailed structure of the Active Record package.

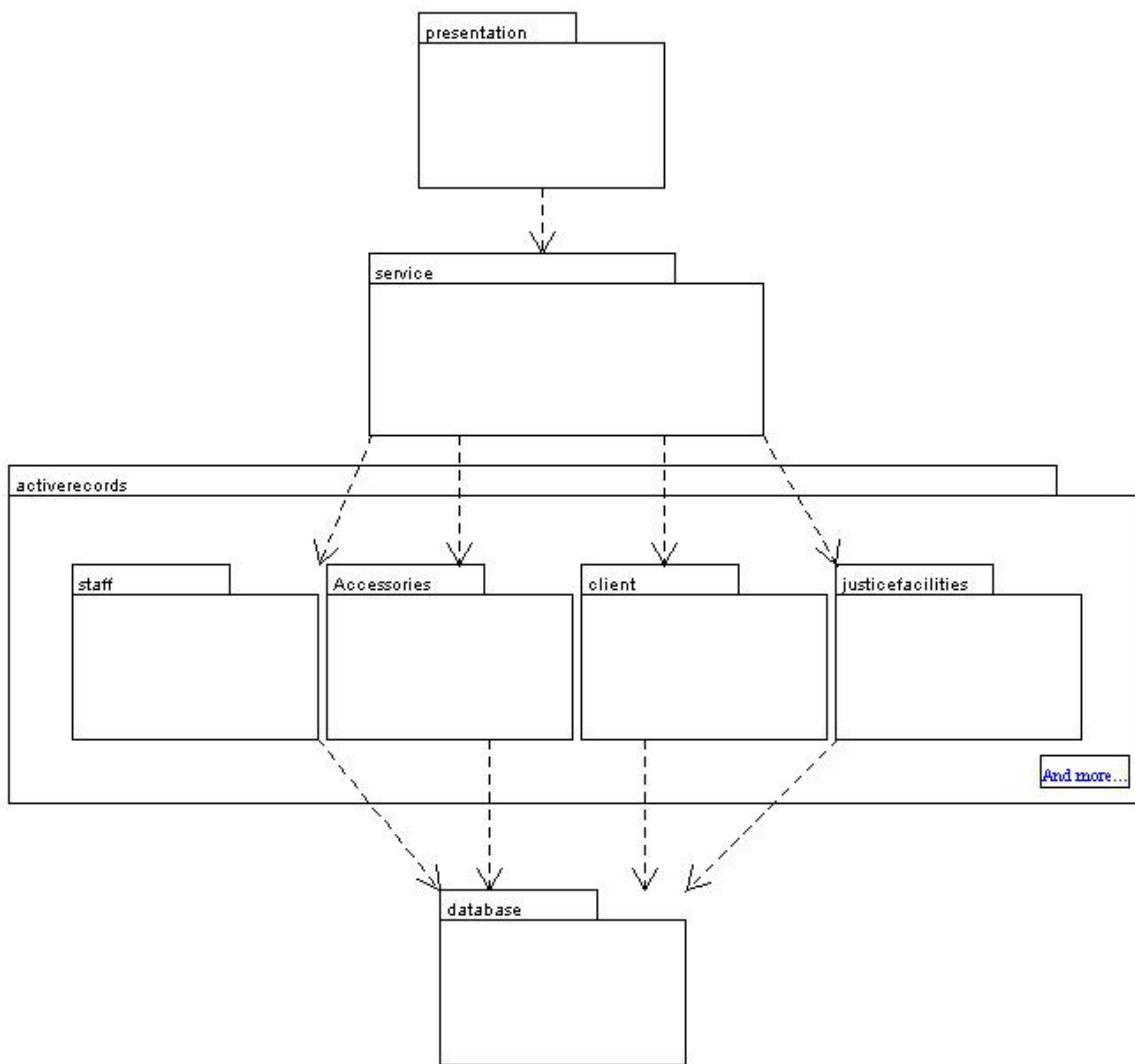


Figure 6: Dependencies between sub-packages and the layer above and below.

The Object – Relational Behavioural Pattern called Unit of Work (Fowler 2003) will handle the communication with the database where necessary. All created objects are immediately registered at the Unit of Work class. If an object changes in any way, it is registered dirty i.e. it is made sure that it will be saved again. The Unit of work will then make sure the data is stored and committed in right order into the database. This feature will only be implemented where necessary and remains to be implemented.

4.3.3 Database layer

This layer handles all connections to the database. A standard connection method used to create the connection has already been implemented. Connection pooling, however, will enable multiple connections hence allowing seamless workflow from many users at a time. The ‘pooling’ term means that many

connections can be kept alive, or reserved, then distributed at will to the requests and returned to the pool when they have completed their task. This provides easy configuration of the connections allowing the administrator to control their number and therefore guarantee sufficient resources for accessing the database.

The connection pooling feature remains to be implemented.

4.4 Database design

The main part of the database design was done in November 2003. The fine tuning of the database design was made during the programming of the system.

This database, today, has 34 tables that contain the necessary information for HLK to be able to register all relevant information about staff, clients, phone calls, and visitations to HLK. The central point in the database is of course – the client. He is having the treatment and the information needed cover all his life as an addict. Information on family situations, financial, health and housing condition, active court cases, jail time and former treatments and addiction need to be entered to be able to give the client the help he so desperately needs.

The database is designed in BCNF (Boys Codd Normal Form). As seen on the diagram on next page, the client table is the largest one containing a lot of information, but also foreign keys to tables that contain other relevant information like lists of relatives, court cases, diseases, ingestion periods etc.

During the programming period I recognised some flaws in the database design. One of them had to do with the client table. I had to retract the financial information into a separate table and create a new table for the client's reasons for starting to drink, to dope and why they went to the first treatment.

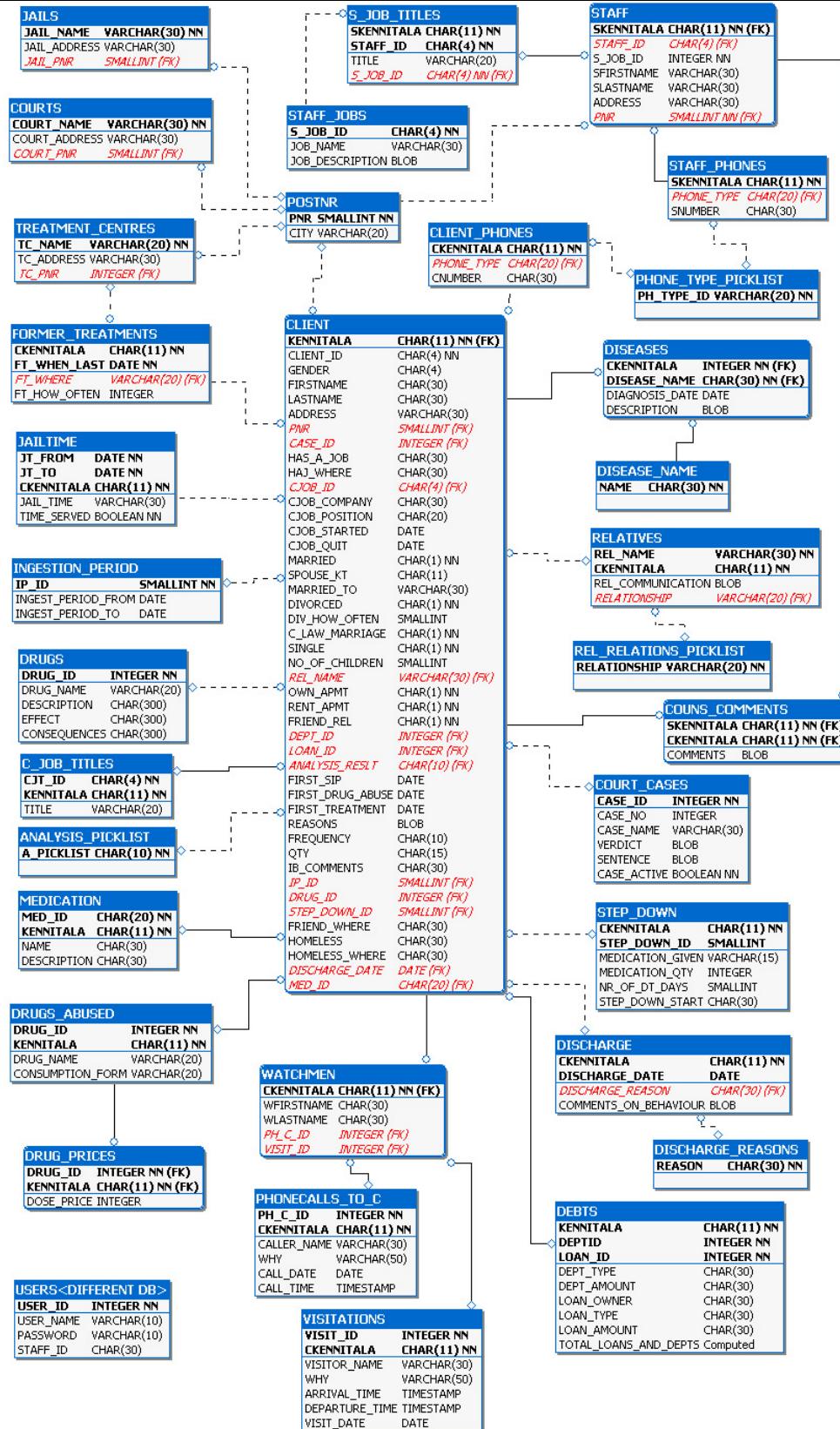


Figure 7: The current database design for the Jfactor application.

5 Implementation

This system will, when fully developed, enable the users to see the client's history, if any, and ease counselling tasks in many ways such as information gathering. Clients will not be asked repetitively about the same things. Repetitive input of data will be eliminated and all parties that are involved in the client's treatment will be able to input and view data regarding the client and his behaviour. This will provide an extensive history on the client were it is possible to do pattern seeking and data mining in order to improve the treatment for every client. The completed system will also provide the tools for the administrator to monitor counsellors. If an unusually high percentage of clients assigned to one particular counsellor are falling repeatedly then the administrator should be able to see that and provide the necessary help and backup for the counsellor, again increasing the quality of the treatment.

5.1 Resources

Let us consider the resources for this project. Operating system used is Windows XP professional but this system will be set up for Linux using the RedHat 9.0 operating system. Programming language used in creating the system is Java 1.4.1_05 and jsdk2.1. Software packages used are IntelliJ IDEA for Java programming, Firebird or Apache Tomcat server and Interbase 6.0 or MySQL 4.01 database using a JDBC driver and IBWorks 1.2 for database design. Macromedia Dreamweaver 5.0 and Fireworks 5.0 will be used for interface design and creation. The reason for mentioning the open source software Firebird/Interbase 6.0 server and database was that I wanted to try to use some other technology than I was used to. This choice, however, was a wrong one as explained in the next section about problems encountered.

Hardware used in this project is standard PC laptop Dell 600m owned by the author. Notes from the first and second year will also be used. Unit testing will be done using jUnit. Other testing like white and black box testing will be done by if time allows. And finally, if time allows, some regression testing will be conducted to see how much stress the system can take.

The reasons for using the technology mentioned above are the following:

- HTML and JSP are very common ways of sending and receiving data in web based applications.
- JSP's and servlets handle web transactions very well.
- If I had known about the power of Java and XML together at the beginning of this project I would have probably used that technology. However, it should not be so hard to change Jfactor in that manner to totally separate the data from the application in the future. Jfactor could that way stay in the front row regarding technical design and implementation.
- Using Java is an obvious choice since Jfactor is a web based application. No other programming language is as well suited for web application development due to its widespread distribution and popularity except maybe the .net technology that is gaining a lot of popularity among developers.

- Using MySQL was also an easy choice. For a non-profit organisation it is not a feasible option to buy an expensive database as Oracle or MS SQL server. MySQL has grown to be a very stable, safe, popular and free open source database for web based applications.
- Apache Tomcat is the top of the line open source web server that has gained popularity among web developers around the world. It is safe, very stable and flexible.
- The Linux operating system is one of the most stable operating systems available. That decision therefore speaks for itself.
- The Macromedia web development tools are very powerful and provide great look for the system. I could not think of any better way to design the interfaces than to use that software package.

Next, let us look at the problems encountered during this project.

5.2 Problems encountered

The problems encountered during the implementation of the system were the following:

- Connection difficulties occurred when trying to set up the database connection class and the ant build script. I had a hard time debugging which resulted in getting help from one of my teachers, Gauti Reynisson. Connection got finally established. The reason was mainly that the Firebird server could not find a path to a log file it was requesting. Solution was to create this log folder and file.
- The foreign key syntax in Interbase is only working on some tables but not all. I am using the same syntax structure in the tables and I cannot see why it is not working on one table when it is working in another with appropriate references to the right table and column. At the time of this writing (19/01/04) I have not found a solution to this problem.
- Interbase does not support the Boolean data type and recommends using char(1) instead. I found a solution in the Interbase knowledgebase but I could not get it working.
- The decision was made to use the open source database MySQL version 4.0.1 for the time being. So far (23/01/04) everything is going ok except programming is going slower than anticipated.
- Since the Interbase database was giving me a hard time I decided to eliminate some more risk factors and decided to use Apache Tomcat server instead of the Firebird server. That turned out to be a good decision in terms of smoother development.
- Enormous time went in the programming part of the project, hence giving me hard time achieving my goals. I had to re-schedule and decrease my goal many times because of that.

5.3 Technology used

Understanding the technology used in this project is vital so this section will provide the fundamental information on how these technologies work individually and together. The diagram below gives a brief idea how these technologies work.

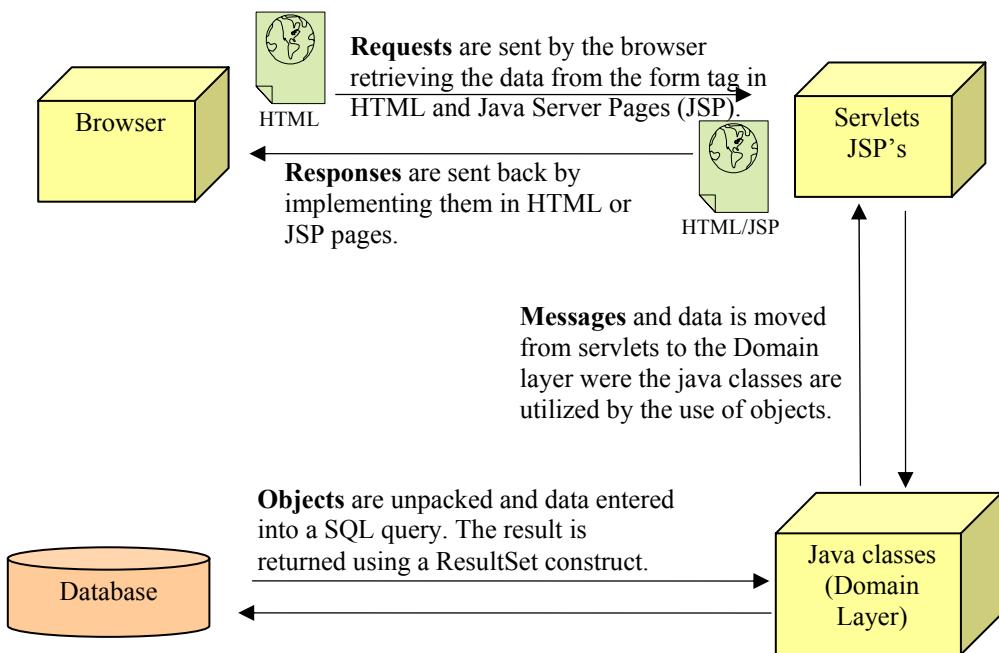


Figure 8: A diagram showing how different technologies flow together.

5.3.1 Hyper Text Transfer Protocol (HTTP)

The HTTP protocol is a very simple and stateless protocol. If for example a browser makes a request, the web server will respond and complete the transaction. An HTTP method is called when the client sends its request. This method will tell the server what type of an action it wants to react to. An example of a HTTP request looks like this:

```
GET/index.htm HTTP/1.1
```

Here we see an example of the GET method. It is used to ask for the index.htm file using the HTTP version 1.1. Now, after this request has been sent the client can, if so desired, send more information about itself using request headers. These headers can be in the form of browser information or what content types are understood by the browser. And then of course the Universal Resource Identifier (URI) of the page being requested is sent to the server. The request and response headers are several and will not be discussed further regarding this project. Further information is widely available on the Internet in many excellent books.

The GET method has already been mentioned. But there are two methods that a browser can use to send and receive requests and responses. These are the GET and the POST methods. The difference between those methods lies in the fact that the GET method is inserted into the HTTP request meaning that the information can be bookmarked such that it can be seen and stored in the browser. This is a bad thing when dealing with sensitive information like this project is about. So the GET method should never be used when

shopping on the Internet or working on a database through a browser. The POST method, on the other hand, passes the information over the socket connection as a part of the HTTP request. This results in the fact that the user can not see in the browsers address bar what information is being sent. This is a good thing when personal and even sensitive information is being handled though a web browser. The user then does not have to worry about the information being caught by the history and cache features in the browser. The Uniform Resource Locator (URL) remains the same in the address bar.

To handle the POST requests we can either use the following code fragment that dispatches all the POST requests to the doGet() method or simply use the doPost() method by itself.

```
public void doPost(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {
    doGet(req, res);
}
```

Now in a HTML page we need also a statement that confirms the use of the POST method.

```
<FORM METHOD=POST ACTION="servlet/kennitala.class">
```

This will make sure that this action will use the POST method. In Jfactor all communication between the browser and server will be done by the use of the POST method.

5.3.2 Hyper Text Mark-up Language (HTML)

This language revolutionized the World Wide Web. Suddenly the Internet became an attractive source of information. This language consists of tags embraced with < and > and HTML requires most of them to have both a start tag like <BOLD> and end tag </BOLD> containing a slash “/” as the first character after the tag opens. Some tags are solo tags like
 meaning there is no need for an end tag.

Ordinary HTML pages containing the necessary forms will be used to gather the required information and then send it using the POST method to the server which returns a JSP page with the required information. For detailed information about HTML (*see section 9- HTML Tutorial*).

5.3.3 Java Server Pages (JSP)

This technology was invented by Sun Technologies, the company that created the Java programming language. A JSP page is an ordinary HTML page except for embedded JSP elements stored between the HTML segments. These code fragments are used to gather information or retrieve data from the database

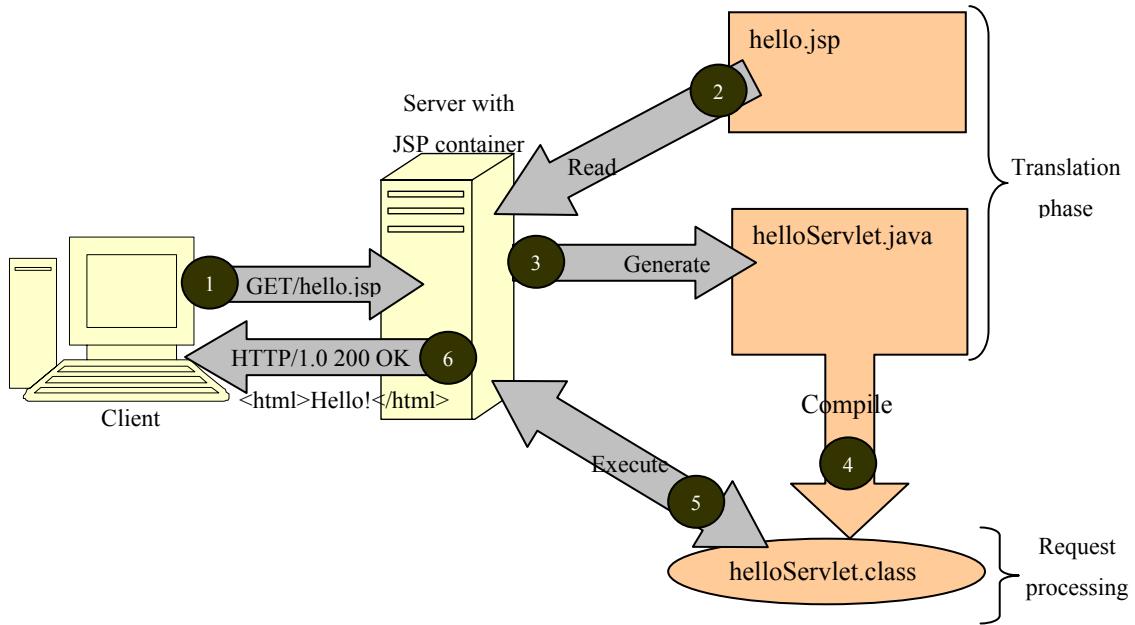


Figure 9: The JSP processing and translation structure (Bergsten, H.).

and display it in the browser. It is well worth noting that *JSP elements are case sensitive*. The advantage of this technology is that a JSP page is sort of another way of writing servlets. The JSP container skips all the HTML tags and creates and compiles a servlet from the JSP elements embedded within the page. The only difference is the translation phase. So when the JSP page is called for the first time, the server needs to compile it (4). Then retrieve the data (5) it is asked for and return it to the user (6). This can take several seconds. But when the server has compiled and created the servlet the JSP page will show up immediately until the server is shut down (steps 1,5 and 6). There is also the possibility to make the server pre-compile all the JSP pages such that the user will not have to wait when the page is called for the first time after booting the server.

JSP technology has been criticised of breaking the Single Responsibility Principle, since both HTML and Java code is mixed up within the same file, i.e. there are two reasons to change the code. However, the JSP technology is a simplified and in many ways easier way of making servlets since the JSP container at the server takes care of reading the JSP page, create the servlet for you, compile it and then execute it. To many this is an attractive feature for less experienced programmers and web site developers. So the main reason for choosing this technology is to gain understanding on how it works and how it can work together with other technologies like HTML.

Let us take a look at the elements that make up the JSP page and the way navigation is performed.

5.3.3.1 JSP Elements

There are three main types of elements that are used in creating JSP pages: directive, action and scripting elements. Let us consider each one of them.

5.3.3.1.1 Directive elements

Directive elements are used for specifying information about the page itself. That means information that is persistent between requests like a name of an error page to be called, buffering information and if session tracking is required or not. The directive elements are three:

Element	Description
<%@ page ... %>	Defines page dependant attributes like mentioned above
<%@ include ... %>	Includes a file during the translation phase
<%@ taglib ... %>	Declares a tag library, containing actions – custom or commercial

An excellent example of using the page element is the following directive:

```
<%@ page contentType="text/html" %>
```

The meaning of this construct is telling the browser what type of content it is getting received, in this case the attributes text and html. The JSP container sends this information as a response header called contentType such that browser will know how to handle the data.

5.3.3.1.2 Action elements

Action elements make the page dynamic i.e. the content changes depending on the input parameter values. There are three main categories of action elements. They are the standard, custom and JSP Standard Tag Library.

All action elements have the form <prefix:action_element_name>. So when used with HTML an example will look like this:

```
<prefix:action_element_name attribute1="value1" attribute2="value2">
    action_body
</prefix:action_element_name>
```

For example, the *standard* actions have the prefix <jsp:action_element>. However, in a *custom* action library we can create our own actions using the whole java and java servlet API's.

The prefix is set using the before mentioned directive element

```
<%@ taglib prefix="unak" uri="..." %>
```

and then this prefix is used in the code.

```
<unak:attribute name="value" ... >
```

The *JSP Standard Tag Library (JSTL)* is a set of five separate libraries that enable the developer to do many of the most common tasks done by JSP applications. These libraries are:

- Core – used for conditional processing and importing external data.
- XML processing – XML data processed using transformation
- Internationalization and formatting – Format and parse localized information
- Relational database access (SQL) – Accessing and manipulating data from relational databases.
- Functions – A generic Expression Language functions.

5.3.3.1.3 Scripting elements

The scripting elements are used to embed pure Java code into the JSP pages. This method will be used in this project. There are three types of scripting tags.

Element	Description
<%! ... %>	Used to declare variables and methods
<% ... %>	Used to embed the scripting code
<%=...%>	This element has two purposes. First to embed expressions and secondly it is used to display attribute values at request time.

The primary drawback of using scripting elements is the tendency to put a lot of Java code into the JSP pages, hence, making it hard to maintain the pages. This can be avoided by creating custom tag libraries. However, this project will not demonstrate the use of tag libraries, but only the use of scripting tags where requests are embedded into objects which are sent down to the domain layer for further processing. So to summarise this discussion on JSP we can picture the JSP pages as an easier way to create servlets. We will discuss servlets a little later but first let us look at the control mechanism provided in JSP.

5.3.3.2 Navigation and Control

To be able to navigate and control where each page leads, it is essential that some kind of a structure is established in the application. JSP pages have special tags to handle this type of control. The

`<jsp:forward page="some.jsp"/>` tag is used to direct the requests to other pages.

`<jsp:param name="Some string" value="another string"/>` tag is nested within the `jsp:forward` tag when used to add parameters to the request. This is best shown by the figure below:

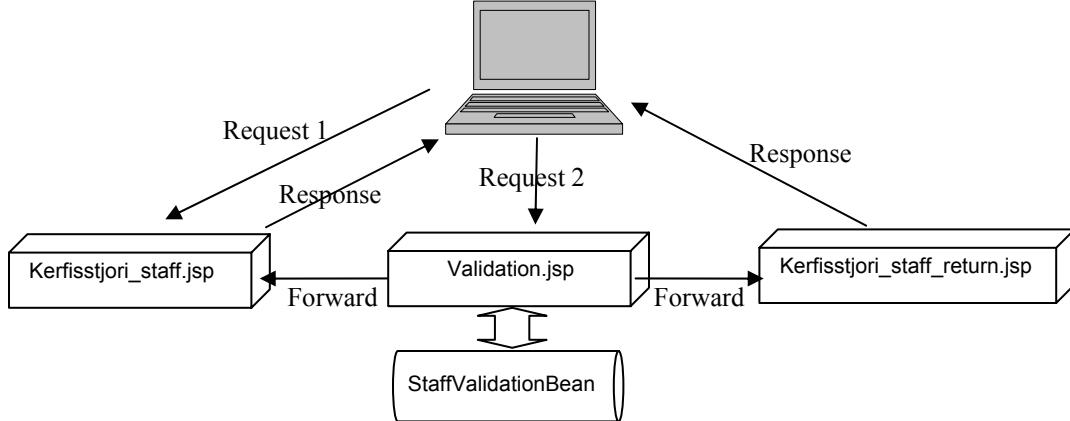


Figure 10: Validation structure for the kerfisstjori pages.

Let us explain this structure a little further. The user asks for the `Kerfisstjori_staff.jsp` page (Request 1). After filling out the form the user submits the request (Requests 2) which is directed to the `Validation.jsp`. The validation page is a page not visible but works in the background communicating with the Bean than handles the staff business logic. If all user input is valid the request is forwarded to the `Kerfisstjori_staff_return.jsp` page and returns the response to the user. Else it is sent back to the `Kerfisstjori_staff.jsp` page with notes on what fields are incorrect. This structure called the Model-View-Controller (MVC) where the bean is the model, different return pages are the controller and the `Kerfisstjori_staff.jsp` is the View.

To control which section of the application is getting each request, a scope parameter belonging to the `useBean` construct, is used. We can choose between page scope were the request or response is only for an individual page and a request scope where the requests and responses can be forwarded between pages. Further we can choose the session scope were where requests and responses are bound within one user and finally the application scope where requests and responses of all users of the application are involved. (Bergsten 2004) This control mechanism works for simple applications. There are more advanced methods available but they will not be discussed here. This feature has not been implemented.

5.3.4 Java Servlets

A servlet is a java class that can be added to extend the server's functionality. Many servers today have implemented this technology and have added a servlet container into their server structure. The real power of servlets lies, among other things, in the portability. Since they are written in Java they can be moved between most common servers used today. Servlets are therefore platform independent. Another powerful thing about servlets is that they can utilise all of the Java API. So tasks like database connections, data compression, object serialisation and internationalisation to name a few will not stop the developer from achieving his goal. When the servlet has been loaded it will remain in the server's memory as a single object instance ready to be used anytime while the server is up and running. This persistence keeps the memory usage low and eliminates the overhead regarding object creation. Servlets maintain their state and they can easily hold on to database connections and other resources local or remote. Servlet code is simple and clean. And servlets are object oriented. One more powerful feature of servlets is the safety of servlets. They inherit the strong type safety and the automatic garbage collection as well as the flexibility provided by the Java programming language. The before mentioned reasoning made servlets an easy choice to pick in this project. (Source: *Java Servlet Programming* by Jason Hunter).

We talk about a servlet lifecycle. This means that a servlet is a persistent object from it's creation until it has been destroyed. This lifecycle involves three steps.

1. Creation and initialisation of the servlet.
2. Service period were the servlet handles zero to many requests from the user.
3. Destruction of the servlet and garbage collection.

For more information on servlets I refer to the J. Hunter's book *Java Servlet Programming*. Plenty of resources are available on the Internet.

5.3.5 Java Classes

The Java programming language was created by Sun technologies. Java is an object oriented language which is designed to work well with web applications. It can be used to develop standalone applications but its major performance drawback is the ‘middle man’ called Java Virtual Machine (JVM). All Java applications run on top of the JVM and which enables the application to use the Java’s strongest fortress, the portability. JVM has now been developed for almost any personal computer or server type available today, making Java applications the most portable software packets there is.

Any application created using the Java programming language is put together using elements called classes. Each class contains a feature or even only a part of a feature of the whole application. Together they connect and are (usually) joined together into one file (.jar file) that stands as a beacon or a gateway for accessing the features that have been implemented. In web applications the classes are stored on the server usually as .war files which are accessed using a standard Java enabled browser.

Each class is built using predefined constructs. These constructs are primitives, constructors and methods that are used to send and receive messages (data) between methods and classes to achieve the desired results. The Java programming language is not within the scope of this report so any detailed information on how to program can be found at www.java.sun.com along with tons of material found in books and on the internet.

5.3.6 SQL - Database Sublanguage

In 1979 IBM had constructed a new database query language called the Structured English Query Language (SEQUEL). After further development it was shortened to SQL.

“SQL is a database sublanguage that is used for accessing relational databases. The letters don't stand for anything!. A database sublanguage is one that is used in association with some other language for the purpose of accessing a database.”(Jim Melton)

This project's SQL usage conforms to the ANSI SQL-99 standard.

This data sublanguage is the most popular tool worldwide in creating and manipulating data and database structures in relational databases and is used with the Java programming language in this project to access the database.

In this project the SQL queries are located in classes that contain the phrase `<nameOfSomePackage>SQLStatements.java` in every sub-package of the active records package in the system. This makes it easy to maintain the queries since they are all located in one place.

The database is created using an Ant build script. From there the database itself is created along with the tables and integrity constraints. See Appendix D – Code listing for the SQL code for the database.

5.4 Work and dataflow summary of the application

Now we have very briefly touched on the technology used in this project. So let us give an example of an insert of a data item to the database and the retrieval of that data back to the browser.

5.4.1 Data insert

To simplify things let us focus on one field, a staff member's first name, and ignore other relevant data that would be in the same request in the real world. When the user has typed the name into the field on the HTML page and presses the 'Submit' button the browser creates a request. This request is of the type (POST) so that the content will not appear in the address bar of the browser. When the request is received by the appropriate servlet (or JSP) it unpacks the request, retrieves the data, creates a Java (staff) object and forwards it to the service layer by passing it to the `PersonnelInterface.java`¹. The interface contains an `addStaff()` method which not only transfers the data object but also a database connection request. While database connection is being established the data object will continue its journey to the database. It will call the `addStaff()` method in the `PersonnelServiceDesk.java` also located in the service layer. This `addStaff()` method, still containing the database connection request, will then call and forward the staff object and the connection to the `saveInsert()` method in the Staff class. The `saveInsert()` method prepares a statement by unpacking the staff object containing the staff members first name and place it into the appropriate variable that will be mapped to the SQL string which then inserts the data into the database using the established connection it got with the staff object. This method concludes by calling the `executeUpdate()` and `commit()` methods to permanently store the data in the database. Committing data will be explained in detail in section 5.6.3.6.

5.4.2 Data retrieval

Again starting at the client, the user types in the data for the query e.g. a Kennitala, and sends it to the server as a request. The servlet (or JSP) unpacks the request and rewraps it into an object and calls the `PersonnelInterface.java`. That sends the object along with a connection parameter to the `PersonnelServiceDesk.java` which contains a method called `getStaffName()`. This method calls a `find()` method in the Staff class which unpacks the staff object, triggers the SQL query and retrieves

¹ The trace will be through the .java source files even though it is done through the .class files in the real system.

the staff members name. This method then repacks the retrieved data into a new staff object using the method `newStaff()` and returns it to `getStaffName()` method in the `PersonelServiceDesk.java` class. There the first name is retrieved from the staff object using `getFirstName()` method. The servlet can use this method to retrieve the first name, place it into a response header and send it back to the browser where it is displayed.

This simple example shows clearly how the data flows from the browser to the database and back. The same method is used with all staff and client data. The interface separates clearly between the presentation layer and the domain layer. The separation between the domain layer and the database layer is more blurry but will do for now. That will be solved later by the use of the Data Mapper pattern (Fowler 2003) which acts as an interface between the domain and the database layers.

Next let us look at some of the tasks encountered during the implementation of this project.

5.5 Interface implementation

In this section I will discuss the interface elements implemented in this project, why they are implemented the way they are and how they work.

5.5.1 Preventing caching and storing of data

Since Jfactor is a web-based system and works with highly sensitive personal information it is paramount that any of the information handled is not cached or stored at the local computers, but only stored and accessed at the server. To prevent caching and storing of the data the following META data must be embedded within the head tags of the page.

```
<head>
    <meta http-equiv="Content-Type" content="text/html;">
    <META http-equiv=Content-Language content=is>
    <META http-equiv=pragma content=no-cache>
    <META http-equiv=Cache-Control content=must-revalidate>
    <META http-equiv=Cache-Control content=no-cache>
    <META http-equiv=Cache-Control content=no-store>
    <META http-equiv=expires content=-1>
</head>
```

In fact this content of the head tags (the META data) MUST appear in ALL pages called or created by Jfactor. Let us examine this META data line by line.

```
<META http-equiv="Content-Type" content="text/html;">
<META http-equiv=Content-Language content=is>.
```

These lines tell the browser what type of content it should expect and what character set to use when displaying the content. In this case a textual content in the form of HTML and the language used is Icelandic.

Next five lines have to do with caching and storing.

```
<META http-equiv=pragma content=no-cache>
```

This line using no-cache directive generally works in older browsers, before Microsoft Internet Explorer(IE) 5.0 and should be used for those that are using an old browser in their computer.

```
<META http-equiv=Cache-Control content=must-revalidate>
```

This line demands the browser to get a new copy of the page at the server so it will not even consider looking into the cache folder.

```
<META http-equiv=Cache-Control content=no-cache>
```

This line does the same thing as the pragma line, but works on some other browsers than IE.

```
<META http-equiv=Cache-Control content=no-store>
```

This directive can be best described by quoting directly to the RFC 2068 about the HTTP protocol.

"The purpose of the no-store directive is to prevent the inadvertent release or retention of sensitive information (for example, on backup tapes). The no-store directive applies to the entire message, and may be sent either in a response or in a request. If sent in a request, a cache MUST NOT store any part of either this request or any response to it. If sent in a response, a cache MUST NOT store any part of either this response or the request that elicited it. This directive applies to both non-shared and shared caches. "MUST NOT store" in this context means that the cache MUST NOT intentionally store the information in non-volatile storage, and MUST make a best-effort attempt to remove the information from volatile storage as promptly as possible after forwarding it." (Fielding et al. 1997)

```
<META http-equiv=expires content=-1>
```

By setting expires content to -1 makes the browser forget instantly that this data ever existed. It is expired and therefore deleted from memory and will never be cached or stored.

If Jfactor is used with Netscape browser then this following JavaScript snippet must be inserted into the BODY tag of every page.

```
onLoad="if ('Navigator' == navigator.appName)  
document.forms[0].reset();"
```

5.5.2 Login page

This authentication page (*see image of the login window in section 4.1.6*) is the first page to appear when Jfactor is accessed. The user has three tries to enter the username and password. The forth try will direct him to Samhjálp's homepage and notify the administrator or any other authority within the organisation. The login processes are as follows:

- User enters information and clicks the button or presses enter.
- Browser creates a POST request and sends the data to the login servlet.
- The servlet verifies the data against the database and if there is not a match the servlet returns a new login page indicating that the user has made an error and has only two tries left. This continues until user is either closed out (used all three tries) or accepted.
- If accepted the servlet redirects the user to his private homepage according to his grants and login information e.g. if user is registered as a counsellor he will get the counsellor homepage and the list of all clients registered to him.

5.5.3 Adding staff members

Since the first thing to do before entering clients into the system is to enter the staff members, let us next take a look at the system's administrator's interface.

The screenshot shows a web-based administrative interface for Jfactor. At the top, there is a navigation bar with links: Heim, Starfsmenn (highlighted in yellow), Heimildir, Póstnúmer, Database, and Data mining. On the right side of the header, there is a logo for 'Samhjálp'. Below the header, the main content area has a title 'Bæta við starfsmanni'. The form contains the following fields:

- Kennitala: 2305662349, Starfsmanna númer: 234
- Nafn: Valdimar, Júlíusson
- Heimili: Frábærulind 45
- Staður: 200, Kópavogur
- Starfsheiti: Ráðgjafi
- Netfang: valdimar@samhjalp.is
- Símitæki: Heimasími, Númer: 564-4444
- Ráðningarár: Karl (radio button selected), Kona (radio button)
- Ráðningarár dags: 12.04.2002
- Notendanafn: valdi23
- Lykilord: (redacted)
- Staðfesta lykilord: (redacted)
- Aðgangs heimild: Ráðgjafi

At the bottom of the form, there are two buttons: 'Vista starfsmann' and 'Hreinsa'. A footer bar at the bottom of the screen displays the text 'Starfsmenn Samhjálpar'.

Figure 11: Administrators interface. Adding new staff member and login information.

This page above allows the administrator to create new staff members and assign to them a user name, password and access rights to the system. The list of current staff members appears at the bottom of the screen. This page above is not ready for use. However, a page called jsptestpage.htm is ready. As seen on the name of this file it is just to demonstrate the technology used in this project. (*See project CD for viewing the shown functionality of the system.*)

5.6 Application implementation

This part of the dissertation will contain a description of the work already done, what classes and methods have been written.

5.6.1 Presentation layer (webfiles directory)

All servlets, JSP and HTML pages are stored in the JSP container in the server. The requests enter this layer and the responses are sent back to the user. Hence, all input and output must enter this ‘door’ to the system. The servlets being implemented will forward the requests to the layer below – the Service layer. This layer is the entrance into the application containing the interfaces necessary to access the methods needed.

The HTML pages ready for use are: `kerfisstjori_staff.htm`

There is only one JSP page ready for use. That page is the `kerfisstjori_staff.jsp`. This page proves the functionality of the technology chosen for this project. The rest is just a tedious job of connecting the rest of the pages to the interfaces already created.

5.6.1.1 Checking the social security number (SSN or Kennitala)

To ensure that no SSN is the same the procedure of Modulus 11 is used in calculating them: I will now describe the method used by the Icelandic government here below. The same method is used either when calculating SSN for individuals or companies.

A mans birthday is e.g. 12. January 1960. The first six digits are therefore 120160 i.e. day, month and year shortened to two digits. The next two digits are randomly chosen e.g. 120160-33. Next is the check sum number, or security number calculated by multiplying the number sequence from right to left with the numbers 2 to 7 as shown in the table below:

1	2	0	1	6	0	3	3
x	x	x	x	x	x	x	x
3	2	7	6	5	4	3	2
3	4	0	6	30	0	9	6

The sub-totals are then added together and in this case the sum is 58. Next we use the Modulus 11 on the sum giving us the remainder of 3. Finally the remainder is subtracted from 11, $11 - 3 = 8$. The check sum or the security number is therefore 8 and the kennitala is 120160-338. If the security number turns out to be 11 then it must be set to 0.

The last (10th) digit in the kennitala is 8, 9 or 0 signifying the century the person was born in and is not checked. (Hagstofa Íslands 2003)

When implementing this into the system I created a servlet called KennitalaCheck. This servlet takes the kennitala from the incoming request and splits and converts the string into single digit integer variables. It then performs the calculation as described above and returns the security number. This number is then compared to the 9th digit in the kennitala. If there is a match the servlet will forward the request to continue to process the rest of the data associated with this particular request. If not, the user is notified.

This check will be done whenever a kennitala is entered into a request i.e. during login, staff administration and client registration. The class is just about finished but is not implemented.

5.6.1.2 Staff JSP's

For time's sake it will not be possible to create more than one JSP page that connects to the domain layer. This page is the system administrator's 'Add Staff Member' page where the administrator will be able to add new staff members to the database and assign to him access rights and authentication codes. This page is implementing both the AccessoriesInterface and the PersonnelInterface.

5.6.2 Service layer

The service layer is containing the interfaces and the service desk classes to access the domain layer. The following files are useable:

- AccessoriesInterface and AccessoriesServiceDesk that contain the methods to modify the data in staff and clients phone and mail information tables in the database.
- PersonnelInterface and PersonnelServiceDesk that contain methods to modify the data in staff, staff jobs and staff job titles tables in the database.
- JusticeInterface and JusticeServiceDesk are partially usable and contain methods that affect the data in the jail table. They will also contain methods that affect the jail time, courts and court cases tables.
- ClientInterface and ClientServiceDesk are partially usable and contain methods that affect the data in the client table.

When this is written it is not known if all interfaces can be changed to sending objects as parameters instead of sending individual attribute values as parameters. Some methods have long parameter lists which is both considered a bad practice and a "code smell" (Fowler 2003).

The reason for keeping the interfaces and the service desk classes in the same package is debateable and probably not the best practice, but will do for now.

5.6.3 Domain layer (Active Records)

This section will describe the content and functionality of each class within each package. This will give and overview of the applications structure and aid the understanding of the domain layer.

5.6.3.1 Staff package

This package keeps all the classes and methods to handle all interactions with the staff related tables in the database.

5.6.3.1.1 StaffPerson.java

This class is a ‘person’ class containing getters and setters describing the information needed for the staff member.

5.6.3.1.2 Staff.java

This class extends the before mentioned StaffPerson class. The staff class contains the following methods:

- *getStaffMembers()* returns a collection with all members of staff. Calls the *findStaff()* method.
- *findStaff()* locates and returns an object with all members of staff.
- *find()* takes kennitala as a parameter and returns the associated name of the staff member.
- *newStaff()* takes in a ResultSet as a parameter and creates an staff object with the requested information and returns it to the *find()* method.
- *saveInsert()* inserts the data from the staff object into the database.
- *saveChange()* updates the database using the data from the staff object.
- *saveDelete()* takes kennitala as a parameter and deletes the appropriate staff member from the database.
- *findStaffId()* returns the staff id of a staff member.

5.6.3.1.3 StaffJobTitles.java

The class contains getters and setters as well as methods to interact with the StaffJobTitles table in the database. The getters and setters will not be listed here.

- *getStaffJobTitles()* returns a collection of the job titles of the staff. This can than be used as a value list in the user interface.
- *find()* returns a StaffJobTitles object which contains the job title for a given job id.
- *newSJT()* is called by the *find()* method and returns a StaffJobTitles object.
- *saveInsert()* inserts the job title data into the StaffJobTitles table in the database.
- *saveChange()* updates the job title data in the StaffJobTitles table.
- *saveDelete()* deletes the job title selected.
-

5.6.3.1.4 StaffJobs.java

The class contains getters and setters as well as methods to interact with the StaffJobs table in the database.

The getters and setters will not be listed here.

- *getStaffJobs()* returns a collection of the jobs that staff members do at Samjhálp.

-
- *find()* returns a StaffJobs object according to the job id given as a parameter. Calls *newStaffJob()*.
 - *newStaffJob()* returns a StaffJobs object to the *find()* method above.
 - *saveInsert()* inserts the data from a StaffJobs object into the database
 - *saveChange()* updates the database with the information from the given StaffJobs object.
 - *saveDelete()* deletes the job from the database by the given job id.

5.6.3.1.5 *StaffSQLStatements.java*

This class contains all the SQL statements for this package. This makes the maintenance of the queries easy. No search is needed since all the queries are in the same place.

5.6.3.2 *Accessories package*

This package will contain all the methods needed to interact with the phone tables for both clients and staff members as well as the postnr table and the phone_type_picklist table.

5.6.3.2.1 *MailInfo.java*

This is a ‘person’ class describing the post number and cities in Iceland, thus containing getters and setters for that task.

5.6.3.2.2 *PhoneInfo.java*

This is a ‘person’ class describing the phone types and numbers. It also contains kennitala to link many phones to one individual, thus containing the necessary getters and setters for that task.

5.6.3.2.3 *PostnrOps.java*

This class extends the MailInfo class. It contains the methods needed to interact with the table containing the post number and city information. These methods are:

- *getPnrs()* returns a collection of post numbers and cities from the postnr table.
- *findPnr()* returns a PostnrOps object containing the requested post number. Calls *newMailInfo()*.
- *findCity()* returns a PostnrOps object containing the requested city. Calls *newMailInfo()*.
- *newMailInfo()* returns an PostnrOps object back to the *findPnr()* and *findCity()* methods.
- *saveInsert()* saves the data from a PostnrOps object into the database.
- *saveCityChange()* updates a change to a city name in the postnr table.
- *savePnrChange()* updates a change to a post number in the postnr table.

5.6.3.2.4 *ClientPhones.java*

This class extends the PhoneInfo class and its primary purpose is to handle the recording of all phones and phone types that are registered to the client.

- *getPhones()* returns a collection of all the phone numbers, types and kennitala id's in the table.

-
- *findClientPhoneInfo()* takes kennitala as a parameter. Returns a ClientPhones object that contains all the information based on the given kennitala.
 - *saveInsert()* inserts the data from a ClientPhones object into the client_phones table
 - *saveChange()* updates the database using the data from a given ClientPhones object.
 - *saveDelete()* deletes the given row in the client_phones table based on the given kennitala.

5.6.3.2.5 *StaffPhones.java*

This class extends the PhoneInfo class and its primary purpose is to handle the recording of all phones and phone types that are registered to the staff member.

- *getPhones()* returns a collection of all the phone numbers, types and kennitala id's in the table.
- *findStaffPhoneInfo()* takes kennitala as a parameter. Returns a StaffPhones object that contains all the information based on the given kennitala.
- *saveInsert()* inserts the data from a StaffPhones object into the staff_phones table
- *saveChange()* updates the database using the data from a given StaffPhones object.
- *saveDelete()* deletes the given row in the staff_phones table based on the given kennitala.

5.6.3.2.6 *PhoneTypePicklist.java*

This class interacts with the phone_type_picklist table in the database. It contains the necessary getters and setters which will not be spelled out here. The methods in this class are the following:

- *getPicklist()* returns a collection of all the pick list items.
- *find()* takes a pick list item as a parameter and returns an PhoneTypePicklist object containing the requested pick list item. Calls *newPicklistItem()*.
- *newPicklistItem()* returns an PhoneTypePicklist object to the *find()* method.
- *saveInsert()* uses the passed PhoneTypePicklist object to save the pick list data into the database table.
- *saveChange()* updates the database table by using the given PhoneTypePicklist object.
- *saveDelete()* deletes the chosen pick list item.

5.6.3.2.7 *AccessoriesSQLStatements.java*

This class contains all the SQL statements for this package. Hence, only one class per package containing all the queries for easy maintenance.

5.6.3.3 *Client package*

This package is at the moment mostly dysfunctional. However, the main structure of the package will be the same as in other sub-packages in the active records package.

5.6.3.3.1 *ClientPerson.java*

This is a person class describing the nature of the information needed for registering the client, his addictive history as well as everything else mentioned before. Hence, this class has many getters and setters which will not be described here any further.

5.6.3.3.2 *Client.java*

This class extends the ClientPerson.java class. It contains the methods needed to interact with the appropriate database table, the client table. These methods are the following:

- *getClients()* returns a collection from an Client object. Calls the *findClients()* method.
- *findClients()* selects all clients from the client table and returns it as an Client object.
- *find()* takes a kennitala as a parameter and returns a Client object containing all the information related to that kennitala.
- *newClient()* picks up all the information from the client table and creates an Client object and passes it back to the *find()* method.
- *saveInsert()* saves the information content of the given Client object into the client database table.
- *saveChange()* updates the client table using the contents of the given Client object.
- *saveDelete()* deletes the client based on the given kennitala.

5.6.3.3.3 *ClientSQLStatements.java*

This class contains all the SQL statements and queries needed to access the client table.

5.6.3.4 *Justice facilities package*

This package handles the data at the tables that have to do with courts, active court cases, jails and jail times and periods.

5.6.3.4.1 *Courts.java*

This class has the methods needed to interact with the courts table. It has the getter and setter methods to describe the necessary information regarding the courts. These methods are:

- *getCourts()* returns a collection of all the courts in the table using a Courts object.
- *find()* returns a Courts object with the selected court name. Calls the *newCourt()* method.
- *newCourt()* returns a Courts object and takes in a ResultSet object as a parameter.
- *saveInsert()* inserts the court name from the given Courts object into the database.
- *saveChange()* updates the court name using the given Courts object.
- *saveDelete()* deletes the requested court name from the database.

5.6.3.4.2 *CourtCases.java*

This class is empty at the time of this writing.

5.6.3.4.3 *Jails.java*

This class has the methods needed to interact with the jails table. It has the getter and setter methods to describe the necessary information regarding the jails. These methods are:

- *getJails()* returns a collection of all the jails in the table using a Jails object.
- *find()* returns a Jails object with the selected court name. Calls the *newJail()* method.
- *newJail()* returns a Jails object and takes in a ResultSet object as a parameter.
- *saveInsert()* inserts the jail name from the given Jails object into the database.
- *saveChange()* updates the jail name using the given Jails object.
- *saveDelete()* deletes the requested jail name from the database.

5.6.3.4.4 *JailTime.java*

This class is empty at the time of this writing.

5.6.3.4.5 *JusticesQLStatements.java*

This class contains the SQL statements used to handle the interaction from the courts and jail tables to query the database.

5.6.3.5 Other classes

Other classes that have not yet been assigned to any package are the following.

5.6.3.5.1 *UnitOfWork.java*

This code is taken from Martin Fowler's book *Patterns of Enterprise Application Architecture*. It needs to be changed and adjusted to this application before being used.

5.6.3.5.2 *DomainObject.java*

This code is taken from Martin Fowler's book *Patterns of Enterprise Application Architecture*. It needs to be changed and adjusted to this application before being used.

5.6.3.5.3 *ItemNotFoundException.java*

This class is an exception class that is called from the methods in the active records. It has not been implemented in all places so null pointer errors and other errors alike will be popping up.

5.6.3.5.4 *JfactorTest.java*

This class contains the tests already created for the active records. Many more are needed.

5.6.3.5.5 *KennitalaCheck.java*

This servlet has not been tested but it contains the code needed to check the kennitala for correctness. This class is located in the Presentation layer.

5.6.3.6 Committing data to database

There are several ways to commit the data to the database. By committing it is meant that information in a transaction is permanently stored in the database i.e. it takes another transaction to modify that change. The MySQL function `setAutoCommit()` is set to false. That means that MySQL will not commit any transactions. There are three reasons for doing this. Firstly the jUnit tests are set up in such a way that after each test the database will perform a rollback i.e. abort the transaction and leave the database untouched. Secondly it is better to be able to control the commits from within the application. Especially when it comes to more complicated commit procedures where e.g. foreign key constraints will force the application to store the data in certain order. This is where the before mentioned Unit of Work pattern comes in. Lastly the commits are not made active in the active records because of the jUnit tests. However, when the system will be well on the way in the development process the commits will be activated for full system functionality testing.

5.6.4 Database layer

The database layer is rather thin at the moment. Only one class is located in the database package. This class handles the database connections the methods used there are the following:

- `openConnection()` returns a Connection object with the driver and login information for interacting with the MySQL database.
- `close()` is a static method that shuts the connection down if it is active.

For the moment the application is only using a simple connection to access the database. However, connection pooling will be the real thing to use. It will enable the administrator to control the number of available connections. Hence, it will make it easy to extend greatly the number of users accessing and using the Jfactor system.

5.7 Database implementation

The database will contain all the necessary information for the staff at HLK to be able to view, analyse and perform data mining on the client's information. The decision was made to program first the pieces that were needed to handle the insertion and modification of the data to and from the database. But first let us take a closer look at the database design.

5.7.1 Database connection

The first task in programming this Jfactor system was to establish a simple connection to the database. Using the open source database MySQL a connection was established in no time using a JDBC driver.

5.7.2 SQL queries

When an active record receives a request it calls the appropriate method that contains an SQL statement which then acts on the database. If it is an INSERT or UPDATE request then data is entered into the SQL

string and executeUpdate() is called (and Unit of Work is then notified – if applicable). If the request is a SELECT query then the appropriate SQL query is called and the result set is returned as a Collection. Since no change is made to the database there is no need to use an insert method or call *commit()*.

5.7.3 Database tables

All tables used in this MySQL database are of type ‘InnoDB’. These tables provide MySQL with an ACID compliant storage engine with commit, rollback, and crash recovery capabilities. All the tables should also be conforming to the Boyce Codd Normal Form (BCNF) normalization standard.

The SQL code for the database tables can be seen in Appendix D – Code listing.

5.7.4 Grants

Grants have not been implemented in this project, but will be before first release. This security feature will enable the system administrator to control the user’s access to the database and therefore restricting them in accessing tables that are not to their concern.

5.7.5 Users table

This table will store all users, their passwords along with their social security numbers (kennitala). The application will then verify the login information given in the request to the user table.

6 Testing

Testing is one of the major phases needed to create good quality software. However, due to time constraints the amount of testing that was originally intended has not been archived. The testing already done is depicted in the following sections. This evaluation chapter then ends with a conclusion on this project.

6.1 Unit testing

During the programming phase, unit tests were created to test each part of the domain layer application. Tests for all classes have not been created. Only 20 tests are there but need to be created when the future of this project is known. These tests were run through the integrated part of IntelliJ called jUnit. The tests were designed in order to capture most of the possible errors that could occur for each unit of the application. However, more tests need to be created to fully verify the code quality, like data integrity checks and foreign key enforcement. The tests already created check the insertion, update and deletion of the tests. The tests can be viewed in Appendix D.

6.2 Other types of testing performed

Other testing that has been done is to install a WAR file of this project onto a Tomcat server (other than the server on my laptop) and actually access it from the internet and see it working.

This turned out to be successful and the already created JSP page reacted as planned. The project was also tested on four types of browsers. Internet Explorer 6.0, Mozilla Firefox 0.8, Mozilla Firebird 1.5 and Opera 7.23. All these Java enabled browsers work fine with the application and display the content correctly.

6.3 Viewing the JSP page

To see the JSP working in this application I refer to the README file on the software CD in the back of this dissertation for information on setup and execution of the page.

7 Evaluation

In this section the project will be evaluated from a broad perspective. The quality and the future of the software will be discussed, what needs to be done to complete it and the possibilities for marketing the final product.

7.1 Project's quality

As can be seen earlier the testing phase was performed on the domain layer using unit testing. Other testing, apart from what was stated in the section above, has not been done since the implementation is needless to say very poor in quantity even though it is working. I have been fighting to get things to work and I have achieved that. I am simply short on time. As the program is today (14th April 2004) some data can be sent to the database and received successfully although it is not properly laid out in terms of look and feel. So in that sense I have proven the technology in this project.

If we look at the project as a whole, it is my opinion that I have laid a solid foundation for the future development of Jfactor. The first part of the database is ready for use, and the overall system structure is mostly completed even though a lot of coding still remains to be done.

Using JSP is a good option for this system. It will take me a while longer to master the JSP technology and further implement the Standard Tag Libraries for page navigation and using Enterprise JavaBeans for validation and other tasks. However, it is my conviction that another technology is even better for this type of software. I think that the final system should be directed towards implementing XML for total separation of program and data. XML is now a world wide standard in representing data. It will most likely continue to be that way for years to come. Since the presentation layer has not been developed to any extent it will not be a great problem to switch over to XML, Java servlets, JDOM and any other applicable technology that will create a fast and secure system.

So, can the current version be considered to be a good system? Since there is so much left to do I can not say this is a good system, but it has an excellent possibility to become a very good system in the near future.

7.2 Project's future

The fact is that Samhjálp does not have any funds to put into the continued development of Jfactor. I have talked to them about this situation and they suggested that they would help me in holding a series of presentations on Jfactor's possibilities to the government authorities that in some way touch this area. Audience from the ministries of health, social services and department of justice would be the audience along with staff in all the major addiction rehabilitation centres. If these presentations are successful then the funding should not be a problem. Alternatively there is the possibility to have a presentation for investors to create a software company around this idea.

There is no doubt in my mind that this software will work when done. And not only that, it will work well and be a standard package for treatment centres both in Iceland and abroad.

There is a lot that needs to be done to complete this software. Apart from finishing this first part of the project there are also the following modules that need to be done:

- Administrator's interface embedding tools for statistical display of the information in the database in both numbers and graphs.
- Treatment manager's interface with tools to handle monthly calculation for the government agencies.
- Watchmen interface for registering phone calls and visitations.
- Doctor's interface for recording health information.
- Prescription interface (might be skipped).
- Connection pooling implementation for controlling and extending the number of connections.
- Session cookies need to be implemented to let the system close the user's access if user has not done anything in 10 minutes.
- Setup and test the system with all security measures for performance and safety.
- Implementation of data mining software.
- Implementation of an international database containing statistical and concrete information showing the situation in each country where Jfactor is in use. Example of such information is current drug prices and a percentage of each drug is being used.

It is my belief that this dissertation shows beyond the shadow of doubt that Jfactor has an excellent opportunity to prosper and grow. When we look at the national news these days (April 2004) we see criticism on the lack of evaluation of the result from spending money in addiction therapies. The demand for concrete statistical figures to justify the means has surfaced. So it is no doubt in my mind that this system will one day succeed if funds will be granted soon for continued development, hence, making Jfactor truly the addict's hope.

7.3 Conclusion

This project has been a deep fountain of learning. Even though the amount of working code is far from the original goal I aimed for. However, it is my firm belief that this project can grow to be an excellent business opportunity by providing rehabilitation centres with a tool that could improve addiction treatments significantly in the future. To me this project has given me a chance to exercise the knowledge in software analysis, design and creation of software applications, learned in this University. This project has also taught me important lessons in applying learning methods, organising projects like this one, as well as teaching me a lot about self discipline and control and last but not least – personal limitations. I have also have come to know, very well my strengths and weaknesses in the field of computer science. That will, however, empower me to learn more where I am lacking knowledge, and to utilise and improve the skills already there.

I want to thank my supervisor, Mark O'Brien, for believing in me and pulling me up when I faced a dead end, and other faculty members of the computer science department in the University of Akureyri for excellent teaching and help.

8 References

Bergsten, Hans. 2003, *Java Server Pages*. 3rd Ed. Sebastopol, California: O'Reilly & Associates, Inc. USA.

Fielding R. et al. 1997, *HyperText Transfer Protocol –HTTP/1.1*. RFC 2068. Available from:
URL: <http://www.w3.org/Protocols/rfc2068/rfc2068> [Accessed in March 2004]

Hagstofa Íslands. 2003, *Hagstofan – Kennitölur*. Available from:
URL: <http://www.hagstofa.is/template38.asp?PageID=808> [Accessed in September 2003]

Hunter, Jason. 2001, *Java Servlet Programming*. 2nd Ed. Sebastopol, California: O'Reilly & Associates, Inc. USA

Wreski, Dave. *Using TCP Wrappers*. 1999. Available from:
URL: http://www.linux-mag.com/1999-10/security_03.html [Accessed in February 2004]

9 Bibliography

Andreasson, Oskar. *Iptables Tutorial 1.1.19.* 2001-2003 Available from: URL:

http://www.linuxsecurity.com/resource_files/firewalls/IPTables-Tutorial/iptables-tutorial.html

[Accessed February 2004]

Bergsten, Hans. 2003, *Java Server Pages*. 3rd Ed. Sebastopol, California: O'Reilly & Associates, Inc. USA.

Fowler, Martin. 2003. *Patterns of Enterprise Application Architecture*. 3rd printing. USA. Addison-Wesley.

Hall, Marty. Brown, Larry. 2001. *Core Web Programming*. 2nd ed. USA. Sun Microsystems Press.

Hunter, Jason. 2001, *Java Servlet Programming*. 2nd Ed. Sebastopol, California: O'Reilly & Associates, Inc. USA

HTML Tutorial. 2004, Available from: URL: <http://www.w3schools.com/html/> [Accessed in March 2004]

Jim Melton. *SQL-92*. 2000-2004. Barry & Associates. Available from:

URL: <http://www.service-architecture.com/database/articles/sql-92.html> [Accessed in April, 2004]

Teen Challenge's Proven Answer to the Drug Problem. 2000-2003. Available from:

URL: http://www.teenchallenge.com/tcreview.html#Jesus_Factor [Accessed in November 2003]

10 Index

A

Accessories package, 35
Action elements, 24
Active Records, 15, 33
addictive, 2, 6
administrator, 8, 19
administrators, 6
analysis, 2
Analysis, 8
Apache Tomcat, 19, 20
Authentication, 11

B

background, 9
BCNF, 17, 40

C

Client package, 36
Committing data, 39
Connection pooling, 16
control, 26
counsellors, 6, 19

D

data mining, 8, 19, 39
Database connection, 39
Database design, 17
Database layer, 9, 16, 39
Database tables, 40
Dependencies, 5, 16
Directive elements, 24
Doc, 10
doctor, 6, 8
Domain layer, 9, 13, 15, 33

E

eMR, 10
ePref, 10

F

finances, 6
Firebird server, 20
Firewall, 11

G

Grants, 40

H

health, 10
Hlaðgerðarkot (HLK), 6
HTML, 9, 12, 14, 19, 22, 23, 24,
28, 30, 32, 44

I

Implementation, 19
information, 2, 6, 8, 19
IntelliJ, 19, 40
Interbase, 19, 20
interface design, 12, 19
Intrusion Detection, 11
IP table, 11

J

Java and XML, 19
Java Classes, 27
Java Server Pages, 22, 43, 44
Java Servlets, 26
JDBC driver, 9, 19, 39
Jfactor, 2, 9, 13

JSP, 5, 9, 14, 19, 22, 23, 24, 25,
28, 32, 33, 41
JSP container, 23, 32
jUnit, 19, 39, 40
Justice facilities package, 37

K

kennitala, 22, 32, 33, 34, 35, 36,
37, 40

L

Linux, 11, 19, 20
login, 5, 12, 31, 33, 39, 40

M

M-18, 7
M-20, 7
Macromedia, 19, 20
Media Access Control, 10
medication, 6, 7, 8, 10
Model-View-Controller, 26
MS SQL server, 20
MySQL, 19, 20, 39, 40

N

Navigation, 25

O

Oracle, 20

P

packet, 11
pharmacist, 6, 8
POST, 14, 21, 22, 28, 31
Presentation layer, 14, 32

protocol, 11, 21, 30	Sequence diagram, 5, 15	Unit testing, 40
R	Service layer, 15, 32, 33	Universal Resource Identifier, 21
rehabilitation, 2, 6, 8, 43	Single Responsibility Principle, 23	Users table, 40
request, 12, 21, 25, 28, 30, 31, 33, 39, 40	SQL, 20, 25, 27, 28, 35, 36, 37, 38, 39, 40, 44	V
requirements, 8	staff, 2, 6, 8	virtual network, 11
Resources, 19	Staff package, 34	Virtual Private Network, 5, 10, 11
response, 12, 21, 24, 29, 30	Sun Technologies, 22	VPN network, 11
S	T	W
SÁÁ, 9	TCP Wrapper, 11	watchmen, 6, 9
SAGA, 10	traffic, 11	Watchmen, 7
Samhjálp, 2, 6, 7, 9, 10, 11, 12, 13, 31, 34	treatment managers, 6, 8	webfiles directory, 14, 32
scope, 26, 27	U	World Wide Web, 22
Scripting elements, 25	Uniform Resource Locator, 22	
Secure Socket Layer, 10	Unit of Work, 16, 39, 40	

11 Glossary

HLK – Hlaðgerðarkot	MVC – Model View Controller
HTML – Hyper Text Markup Language	NIC – Network Interface Card
HTTP – Hyper Text Transfer Protocol	PC – Personal Computer
IDS – Intrusion Detection System	RSS – Reykjavík Social Services
IP – Internet Protocol	SÁÁ – National Center of Addiction Medicine
JSP – Java Server Pages	SEQUEL – Structured English Query Language
JSTL – Java Standard Tag Library	SQL – Structured Query Language
JVM – Java Virtual Machine	SSN – Social Security Number (Kennitala)
M-18 – Miklabraut 18, Reykjavík	TCP – Transmission Control Protocol
M-20 – Miklabraut 20, Reykjavík	URI – Universal Resource Indicator
MAC – Media Access Control	URL – Uniform Resource Locator
META – A prefix meaning one level of description higher	VPN – Virtual Private Network
	XML – Extensible Markup Language

Appendix A

Software Requirements Specification

Software Requirements Specification

for

Jfactor

Version 0.7 approved

Prepared by Yngvi Rafn Yngvason

University of Akureyri

19 March 2004

Table of Contents

1.	Introduction	52
1.1	Purpose	52
1.2	Document Conventions	52
1.3	Intended Audience and Reading Suggestions	52
1.4	Project Scope	52
1.5	References	53
2.	Overall Description	54
2.1	Product Perspective	54
2.2	Product Features	54
2.3	User Classes and Characteristics	55
2.4	Operating Environment	55
2.5	Design and Implementation Constraints	55
2.6	User Documentation	56
2.7	Assumptions and Dependencies	56
3.	System Features.....	56
3.1	Secure Login	56
3.1.1	Description and Priority	56
3.1.2	Stimulus/Response Sequences	57
3.1.3	Functional Requirements	59
3.2	Web user interface	61
3.2.1	Description and priority	61
3.2.2	Stimulus/Response sequences	61
3.2.3	Functional Requirements	61
3.3	Web page generation	61
3.3.1	Description and priority	61
3.3.2	Stimulus/Response sequences	61
3.3.3	Functional Requirements	62
3.4	Automatic lookup	63
3.4.1	Description and priority	63
3.4.2	Stimulus/Response sequences	63
3.4.3	Functional Requirements	64
3.5	Link generation	65
3.5.1	Description and priority	65
3.5.2	Stimulus/Response sequences	65
3.5.3	Functional Requirements	65
3.6	Structured forms for recording data	65
3.6.1	Description and priority	65
3.6.2	Stimulus/Response sequences	65

3.6.3	Functional Requirements	65
4.	Database design.....	66
4.1	Database schemas	66
4.2	Key constraints	66
4.3	Views	66
4.4	Database security	66
5.	External Interface Requirements	67
5.1	User Interfaces	67
5.2	Hardware Interfaces	67
5.3	Software Interfaces	67
5.4	Communications Interfaces	67
5.4.1	VPN (Virtual Private Network)	67
5.4.2	HTTPS (Optional)	67
5.4.3	JDBC/ODBC	67
5.4.4	Servlets	68
5.4.5	Web browser	68
6.	Other Non-functional Requirements	68
6.1	Performance Requirements	68
6.1.1	Login	68
6.1.2	Encryption	68
6.1.3	Browser – database communication	68
6.2	Security Requirements	68
6.2.1	Secure Socket Layer (SSL) (OPTIONAL)	68
6.2.2	Caching	69
6.2.3	IP tables - firewall	69
6.2.4	Encryption	69
6.2.5	Virtual Private Network (VPN)	69
6.2.6	TCP wrappers	69
6.3	Software Quality Attributes	70
6.3.1	Testing	70
6.3.2	Portability	70
6.3.3	Usability	70
6.4	Style guide	70
6.4.1	Templates	70
6.4.2	Fonts and sizes	71
6.4.3	Buttons	71
6.4.4	Icons	71
6.4.5	Logo	71
6.4.6	Sidebar	71
6.4.7	Images	71

Revision History

Name	Date	Reason For Changes	Version
Yngvi Rafn Yngvason	27.11.2003	This is the first version of this SRS	0.1
Yngvi Rafn Yngvason	10.01.2004	Updates during implementation	0.3
Yngvi Rafn Yngvason	15.02.2004	Updates during implementation	0.5
Yngvi Rafn Yngvason	21.03.2004	Updates during implementation	0.7

Introduction

Purpose

The purpose of this document is to enumerate the requirements needed for building a computer system, called Jfactor, for Samhjálp – treatment centre. The document describes the requirements for all components of the Jfactor system. The SRS document should directly lead to a feasible design for implementing the system.

Document Conventions

Fonts used in this document are as follows:

- Main headings Times 18 pt. Bold
- Sub headings – Times New Roman 14 pt. Bold
- Plain text – Times New Roman 12 pt. Normal
- Use case heading – Times New Roman 10 pt. Bold
- Use case body – Times New Roman 10 pt. Normal
- Every requirement will have its own priority.

Intended Audience and Reading Suggestions

This document is intended for the developers, testers and teachers and to help documentation writers of the Jfactor project. Section one and two should be read by all members of the intended audience. System designers should refer to sections three and four for a detailed description of the system's functional requirements.

Project Scope

The name of the project “Jfactor” comes from a research made by the United States government where they discovered the “Jesus factor”. This research was on all treatment centres in the US. The results of this research was that the Christian treatment centres gave the far best results in their treatments or 65-80%, opposite to the government run programs that gave normally 25 to 30% results (*see chapter 0*). This however has not been implemented before into a system where the user can quickly see black on white information like the percentages of sober individuals in a given time period i.e. choosing time periods from 1-10 years, all clients registered to each counsellor, perform data mining on the database gaining valuable information about patterns and show various statistical information of overall progress of the treatment.

This project is designed to meet the needs of treatment centres for alcoholics and other individuals with any other addictive behaviour. Since Samhjálp is a geographically distributed organisation, this system is built upon existent web technology using a browser, web server and a database, -i.e. 3 tier architecture. The

database will be containing highly sensitive personal data and will need to have high security standard, equivalent to an Internet banking system.

The objectives are to record the client's alcohol and drug abusive history and his progress while in treatment and after being discharged back into normal life or his additional steps towards that goal. Specific information will be recorded such as behaviour, habitation, financial, work and family situations. Active court cases, health and abuse history, drug types and prices are just few of many other things that will be recorded into the database.

The purpose of this is to be able to pull out information on abusive behaviour and possibly be able to spot patterns that can help Samhjálp to improve the treatment significantly. This system will when completed, also be able to spot problems that counsellors might encounter. This will help Samhjálp to help the counsellors to maximise their success as counsellors. This system will also be able to display information on current drug prices and display textually and graphically different pieces of information.

Future implementation will be an international database or data warehouse where numerical information from the local databases, untraceable to individuals, will be stored giving the administrators plenty of ammunition in discussions and debates on this issue in public media giving the treatment centres power to present their successes, failures and discoveries.

So the scope for version 0.7 of this SRS will include the following:

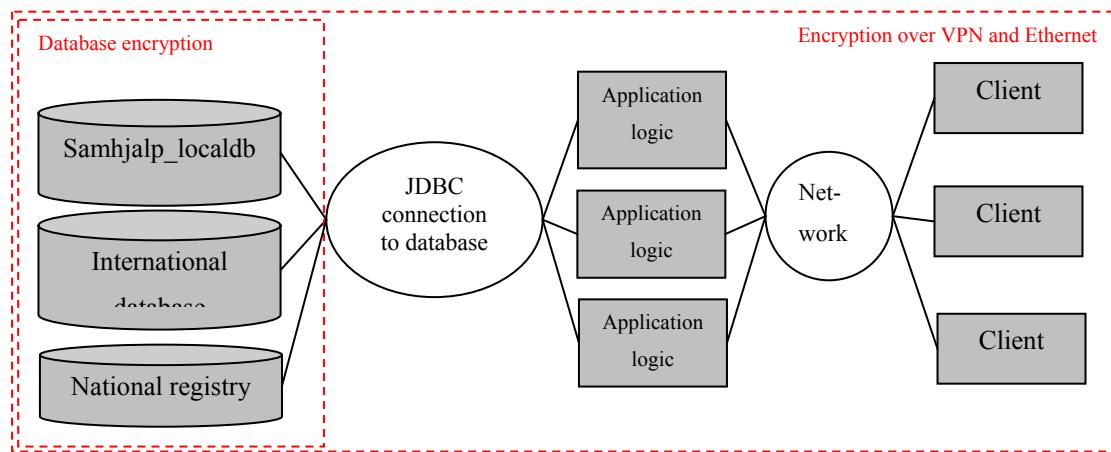
1. Design of the database
2. Implementation of the database tables regarding clients at Hlaðgerðarkot treatment centre and all potential users of the system.
3. Connection Pooling
4. Servlets for insertion update and delete of information in the database.
5. Servlets for generating html pages
6. JSP pages for submitting queries
7. JSP pages for submitting insert and update
8. Security implementation of the system i.e. usernames, passwords and internal security of the system.
9. Design of the user interfaces for treatment managers, counsellors and the doctor.

References

Jesus factor article: http://www.teenchallenge.com/tcreview.html#Jesus_Factor

Kennitala verification formula: <http://www.hagstofa.is/template38.asp?PageID=808>

Overall Description



Product Perspective

This software system is a new and self contained product. There is no other system to the author's knowledge that is specialised in alcoholic and drug abusive treatments like this project. There is no other larger system that will be linked to this system at the time of this writing. There will, however, be a connection to two other databases in the final version (not, however, in this project). The first is an International data warehouse where statistical untraceable data will be stored from the local databases and the other containing the national registry. The third database might be implemented and that is a database containing a list of all drugs that are allowed to use in Iceland.

Product Features

The major features of the system in this version are the following:

- Secure user's login and password window where the users input will be encrypted and sent using secure socket layer (SSL). (*See section 0 below*)
- Web interface views tailored for each user for administrators, treatment managers, counsellors etc. (*See section 0 below*)
- System will generate web pages with links to all relevant places of interest for each user. (*See section 0 below*)
- Automatic lookup for name and address when kennitala has been entered into the system will be implemented. (*See section 0 below*)
- Link creation on individual items e.g. all former treatments of a client. Counsellor can view a detailed history of each client. (*See section 0 below*)
- Structured forms for consistent input and easy data manipulation which will help make use of data mining. (*See section 0 below*)

User Classes and Characteristics

Jfactor will have several primary user classes, the administrators, treatment managers, counsellors, doctor, pharmacist, and the watchmen. Members of this user class will be adults and will have basic computer literacy skills (including use of a keyboard and mouse). There are no strict educational requirements for this user class outside of a basic understanding of using an Internet browser.

Operating Environment

The system server will run on a PC using the Linux RedHat 9.0 operating system. The database system will be the InterBase 6.0 system which is an open source database from Borland. The access to the database will be controlled by Apache Tomcat server. The Java web application stored on the Tomcat server will be written by the author of this SRS. The system will be accessed by any Java enabled web browser through ADSL internet connection. Connection size needs to be from 256 kb or greater.

Design and Implementation Constraints

The Jfactor system will contain highly sensitive personal data so security is an important issue. All users must log in with a username and password (Optional!!!) and all activity must be done using 128 bit encryption through the secure socket layer (SSL)). Encrypted Virtual Private Network (VPN) will be used to secure the connection between the client and server plus encrypting the Ethernet itself. IP tables in the Linux OS will be used as a firewall and a packet filter. Only chosen computers will be allowed to connect to the system. TCP wrappers will be used to encapsulate further the sent and received packets closing out all other traffic. An Intrusion Detection System called Snort will be used to capture any intrusions to the system. The Linux OS provides extra security by using built in 32 bit encryption of passwords so the Linux system never knows any password entered, only the encrypted 32 bit string. The hard drive storing the data will also be encrypted. A session will be created for each user that logs into the system. This session will automatically log the user out if there is no activity going on for 15 minutes. One security issue is left to be mentioned which are actions to prevent the caching and storing of all temporary files during the session period.

There are no hardware or memory constraints unless the database enforces itself regarding caching and transaction processing. This system will not interfere with those constraints.

The programming language used in this project is Java, developed by Sun Microsystems™. The 1.4 version of the API will be used and coding will be done according to Java code conventions. The application will consist of java servlets and java server pager (jsp). It will be designed using object oriented methods and patterns to avoid code smell that make maintenance of the system harder than it should be. The system will

be programmed using IDEA IntelliJ. The Apache Tomcat server and the MySQL 4.0.1 database will handle the data transactions.

The communication protocol TCP/IP will be used from browser to server. There will be a JDBC/ ODBC driver that will handle the communication between the server and the database.

User Documentation

User manuals will be created for the system administrator. A detailed dissertation will also be provided containing all relevant information on this project.

Assumptions and Dependencies

In this project I will make the following assumptions regarding hardware and software packages.

- Computer will be new and with at least 512MB of RAM.
- RedHat 9.0, Linux operating system will be installed on the server.
- Apache server can work seamlessly together either in Windows XP pro or Linux environment.
- Snort (Intrusion detection package) works well
- VPN will have no negative effect on the communication between client and server.
- MySQL 4.0.1 is sufficient for this project, knowing that MySQL works well with Java.

System Features

The system features mentioned in this chapter will demonstrate the functionality of version 1.0 of this project. The features will be written in a logical order, i.e. starting when the user sits down at the computer and opens the system. Use cases are used as a tool to highlight the most important parts, functionalities and features of the system. Number of use cases will increase as this project is being developed.

Secure Login

Secure login allows only legal users to enter and use this system.

Description and Priority

The secure login is a MEDIUM priority part of the system.

When the user enters the address of the login page the system establishes the Secure Socket Layer protocol and then opens the login page. The login page will be written in JavaScript code. When the Submit button is pressed the system will validate the information and, if legal, feed the user the view page he is intended to use.

Component rating (1-10):

Benefit	10
Penalty	9
Cost	3
Risk	5

Stimulus/Response Sequences

In this section are two use cases that show the response sequences for the secure login procedure.

In the first one the user enters the URL and the system sets up the SSL and redirects him to the login page.

The second use case handles the procedure of logging into the site.

Use Case: 1 User enters the URL that directs him to the secure site.

CHARACTERISTIC INFORMATION

Goal in Context: To access the login page for entering the system.

Scope: Jfactor

Level: Primary task

Preconditions: User has a connection to the Internet.

Success End Condition: Successful retrieval of login page.

Failed End Condition: User needs to try again.

Primary Actor: Login servlet in the Jfactor system

Trigger: Enter button pressed

MAIN SUCCESS SCENARIO

1. User enters the URL into the address field of the browser
 2. User presses "Enter"
 3. URL requests for SSL
 4. URL triggers the login servlet which creates the login html page
 5. System returns the login page.
 6. User can start the login process (*see Use Case 2*)
-

EXTENSIONS

- 3.1 Server responds by checking the certificate and then accepting the request.
-

SUB-VARIATIONS

None

RELATED INFORMATION (optional)

Priority: HIGH

Performance Target: < 2 seconds

Frequency: 10 times per day per user

Superordinate Use Case: None

Subordinate Use Cases: Use Case 2

Channel to primary actor: Interactive

Secondary Actors: SSL and Server

Channel to Secondary Actors: HTTP

OPEN ISSUES (optional)

Use of JavaScript or JSP

SCHEDULE

Due Date: TBD

Use Case: 2 User logs into the system

CHARACTERISTIC INFORMATION

Goal in Context: To log into the system through the SSL

Scope: Jfactor

Level: Primary task

Preconditions: User has entered the sites address and the browser has requested an SSL connection. Server has approved.

Success End Condition: Successful login.

Failed End Condition: User needs to try again.

Primary Actor: User

Trigger: Submit button pressed

MAIN SUCCESS SCENARIO

1. User enters username in a text field
 2. User enters password in a password field
 3. User presses "Enter" or clicks the "Submit" button
 4. Information encrypted by the SSL
 5. System decrypts information
 6. System queries the database using the received information for verification
 7. If information is accepted then acknowledge and redirect user to his view.
 8. User has entered the system and can start working.
-

EXTENSIONS

- 4.1 Encrypted information sent to the system.

7.1 If access is not granted then clear username and password fields and ask user to try again and showing the user the number of tries he has left before redirection.

SUB-VARIATIONS

7.2 If three attempts to connect are not successful then redirect user to Samhjálp's homepage with a message why he is redirected.

RELATED INFORMATION (optional)

Priority: HIGH

Performance Target: 2 seconds

Frequency: 10 times per day per user

Superordinate Use Case: Use Case 1

Subordinate Use Cases: Use Case 3

Channel to primary actor: Interactive

Secondary Actors: SSL, Server and database

Channel to Secondary Actors: TBD

OPEN ISSUES (optional)

Use of JavaScript or JSP

SCHEDULE

Due Date: TBD

Functional Requirements

To make secure login work, the needed elements are:

REQ-1: An Internet connection – must be active when a login attempt is made.

REQ-2: Java enabled browser – the browser will issue an error message if java is not enabled.

REQ-3: Optional! – SSL enabled server – will not allow access and not order the login page to be sent unless the certificate is accepted and browser and server agree on the SSL protocol. The server decrypts the username and password when received from the browser.

REQ-4: A login servlet – which both generates the html and handles the login information.

A comparative method will compare the login data against the database. If username and/or password are not valid the login form is cleared and user asked to try again, noting the number of attempts left before the user is redirected. When, however the login data is accepted the system will redirect the user to his/her view.

Web user interface

Description and priority

Priority is MEDIUM.

Stimulus/Response sequences

TBD

Functional Requirements

REQ-1: Every page must have a ‘natural’ timeline flow from left to right and from top to bottom of the page.

REQ-2: The Jfactor logo in the top left corner must always bring the user to his homepage.

Web page generation

This part of the system will generate the dynamic content the user needs when entering his home area. This part of the system must have the ability to generate pages according to ergonomics design guidelines providing the data that the user needs such that each page will not be overloaded by the volume of information, thus making it hard for the user to grasp the relevant information.

In this version of Jfactor there will only be page generation for the counsellor and if time allows, the treatment manager.

Description and priority

Priority is HIGH.

Stimulus/Response sequences

Use Case: 5 Counselor’s Home Page generation

CHARACTERISTIC INFORMATION

Goal in Context: To log into the system through the SSL

Scope: Jfactor

Level: Primary task

Preconditions: User has entered the sites address and the browser has requested an SSL connection. Server has approved.

Success End Condition: Successful login.

Failed End Condition: User needs to try again.

Primary Actor: User

Trigger: Submit button pressed

MAIN SUCCESS SCENARIO

1. Servlet gets the acknowledgement of accepted user and is passed the username
 2. Servlet passes request to domain layer
 3. Class queries the db for the counselor view.
 4. All clients located that are registered to the user.
 5. Data passed back up to servlet.
 6. Servlet constructs the web page and adds the data to it and closes the file.
 7. Servlet returns the page to be displayed.
-

EXTENSIONS

None

SUB-VARIATIONS

None

RELATED INFORMATION (optional)

Priority: HIGH

Performance Target: < 2 second

Frequency: 8 times per day per user

Superordinate Use Case: Use Case 2

Subordinate Use Cases: TBD

Channel to primary actor:

Secondary Actors: database layer

Channel to Secondary Actors: SQL

OPEN ISSUES (optional)

Use of Java and JSP

SCHEDULE

Due Date: TBD

Use Case: 6 Client selected

TBD

Use Case: 7 Clients history

TBD

Functional Requirements

REQ-1: The system must generate a page for starting a new treatment

REQ-2: The system must generate a search page.

REQ-3: The system must generate a list of client's history, including former treatments, abuse history and current situation of the client.

REQ-4: The system must generate pages that aid insertion, deletion and updating of information.

REQ-5: Patterns required in this system are Active Records, Unit of Work and Observer.

Automatic lookup

The lookup feature will be concerned with getting identity information from the national registry and return it to the user.

Description and priority

Priority is MEDIUM. Samhjálp will need to buy a copy of the national registry to make this possible. The information that will be looked up based on kennitala is:

- Full name
- Address
- Postcode
- City
- Marital status
- Name of spouse
- Children (optional, could help counsellor to reach hardened addicts)

Stimulus/Response sequences

Use Case: 3 User enters kennitala for automatic lookup

CHARACTERISTIC INFORMATION

Goal in Context: To perform automatic lookup in the national registry

Scope: Jfactor

Level: Primary task

Preconditions: User has been granted access into the system and is located at his home page.

Success End Condition: Successful lookup

Failed End Condition: User needs to type information in manually.

Primary Actor: User

Trigger: Submit button pressed

MAIN SUCCESS SCENARIO

1. User enters kennitala in a text field
2. User presses "Enter" or clicks the "Submit" button

3. Kennitala is verified using the verification formula (*see section 0 for a link to the formula*)
 4. A Lookup class will search for the data in the database and return it to user.
 5. On commit, all information including the added information from user will be stored in the database.
-

EXTENSIONS

- 3.1 If kennitala is incorrectly typed in an alert window pops up and asks the user to retype it.
 - 3.2 If correct the kennitala is sent to the server.
 - 4.1 If server cannot find the information, lookup the national registry
-

SUB-VARIATIONS

- 4.2 If for some reason the national registry is down, enter the information manually.
-

RELATED INFORMATION (optional)

Priority: MEDIUM

Performance Target: < 1 seconds

Frequency: 10 times per day per user

Superordinate Use Case: Use Case 2

Subordinate Use Cases: Use Case 4

Channel to primary actor: Interactive

Secondary Actors: Server and database

Channel to Secondary Actors: TBD

OPEN ISSUES (optional)

Use of JavaScript or JSP for the kennitala verification, Java servlets for the retrieval of data

SCHEDULE

Due Date: TBD

Functional Requirements

Assuming the requirements from section 0 as a standard for the whole system then other requirements are:

REQ-1: A validation class or script to verify the typed kennitala.

REQ-2: Lookup class that will check for the data at the database, if unsuccessful lookup the national registry and return the data to user.

REQ-3: Unsuccessful lookup must direct the user to a manual typing of information.

REQ-4: Information must then be successfully entered into the database on commit.

Link generation

The generation code must be able to receive strings that are to be converted into links.

Description and priority

Priority is HIGH. Example of use is when the user searches for a specific client he will get a collection of former treatments (if available). Another example is when a counsellor enters the system and accesses his home area he will see all the names of all clients that are registered to him as links.

Stimulus/Response sequences

TBD

Functional Requirements

TBD

Structured forms for recording data

These forms will ease the use of the system. Make the insertion of data easy and painless for the staff.

These forms are focused on the first three interviews of a client when arriving at HLK.

Description and priority

Priority is HIGH.

Stimulus/Response sequences

TBD

Functional Requirements

TBD

Database design

Database schemas

The data dictionary will contain the following schemas:

Key constraints can be seen in the schema above. But if in doubt all constraints will be displayed in a table in the next section.

Key constraints

Views

The views needed for this system are for the following users:

1. counsellors
2. treatment managers
3. doctor
4. pharmacist
5. administrators and finally
6. watchmen

These views will be shown here below. But in this version of Jfactor, only the counsellors and treatment managers' views will be implemented into a web user interface since these staff members input most of the relevant data regarding the clients coming for treatment.

The implemented views look like this: (TBD)

Database security

Each user will only be granted access to the data he needs. Data that is not relevant to his job must be closed for him. All tables must be normalised into Boyce Codd Normal Form (BCNF) to ensure referential integrity.

InterBase provides all necessary features available for database security including recovery, concurrency and integrity.

External Interface Requirements

User Interfaces

The user interfaces must be designed to contain all relevant information in a logical workflow order for each user. This is crucial especially for the treatment manager, counsellor, doctor and pharmacist roles. The watchmen and administrators will use views that are more general in use. The workflow must be pinned down using standard ergonomics techniques like task analysis and HCI.

Hardware Interfaces

The communication protocol used in this project will be TCP/IP over ADSL Internet connection. The computers will use Ethernet connections with a router to enable solid connections. The Linux OS works on TCP/IP so there will not be a problem using this protocol.

Software Interfaces

Let us follow one request from client to server and back.

The request is in http protocol format. It then is encapsulated using a TCP wrapper and protocol and passed to the VPN and encrypted in the virtual network interface card. It then is forwarded to the real NIC where it is encrypted again. The packet is now sent over the network using IP protocol and received at the server, checked if it is allowed, sniffed at and decrypted first on the real NIC then on the virtual one. Then the TCP wrapper is checked, and removed if valid. The original http request is now processed and returns the page with the data. Now the same process starts again with the encapsulation, TCP wrapping and encryption sending and decryption. The page is then displayed in the browser.

Communications Interfaces

The communications between the system layers are as follows:

VPN (Virtual Private Network)

The communications between the user interface and the server will be achieved by using the built-in VPN network. This is a standard both in Windows and Linux operating systems. The VPN will allow configurable encryption on both the virtual network card as well as on the physical card. This will provide a secure closed network that will enable fast and reliable communication between client and server.

HTTPS (Optional)

Optional is the use of communications between the user interface and the server by using HTTPS communication protocol which is a protocol using the Secure Socket Layers (SSL). The SSL must provide 128 bit encryption on all transmissions between client and server.

JDBC/ODBC

This protocol must be used to connect the server to the database. The JDBC/ODBC drivers available are called InterClient and JayBird and either one must be used to access the InterBase database system.

Servlets

Java Servlets will be used to generate the dynamic html pages needed and perform the transactions to the database.

Web browser

Any Java enabled web browser should be able to use this system. Test should be performed on IExplorer, Mozilla and Opera which are all Java enabled browsers.

Other Non-functional Requirements

Performance Requirements

Performance must be high. Though this might include some trade offs the aim must be to make the system as fast as possible. There are at least three issues regarding performance in the system. To ensure good performance the Internet connection should be no less than 256Kb per second.

Login

The time it takes to achieve the login page must be less than 2 seconds. Assuming that username and password are correct the login time should be less than 3 seconds.

Encryption

Encryption used in this system will not degrade its performance by any noticeable measures.

Browser – database communication

This performance issue is dependent upon the Internet connection speed of the client. The faster the connection is the faster is the retrieval of the data into the browser. The database must be optimised for maximum output and SQL queries must be optimised for maximum speed.

Security Requirements

There are several security requirements for this system. These are listed below in priority order.

Secure Socket Layer (SSL) (OPTIONAL)

This system MUST at all times run in SSL mode using 128 bit encryption of every single bit that is moved between the browser and server. The system must ensure that the system is closed down if something happens to the SSL layer.

Caching

No information can be cached or stored in the client's browser. This is to ensure that nobody can access sensitive data without logging into the server.

IP tables - firewall

The system must be protected by a firewall where all packages are blocked except VPN packages that are destined for the Jfactor system and only packets that will come from certain IP addresses.

Encryption

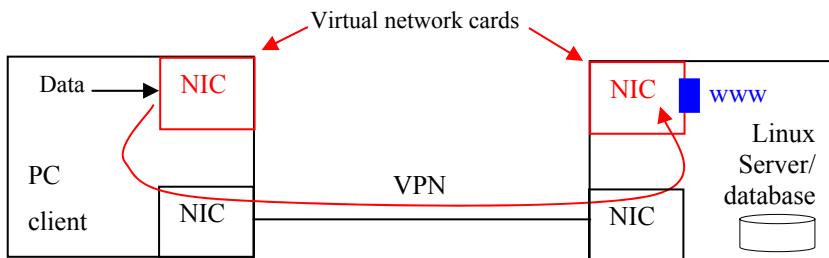
All username and password access into the Linux server must be encrypted using the built-in encryption feature of the Linux operating system.

Virtual Private Network (VPN)

Both Microsoft Windows and Linux operating systems provide encryption through VPN. This allows the encryption of both the virtual network card and the Ethernet hardware card in the computer. This would have a timeout option such that if a computer is not used for 15 minutes the VPN will close the system until the user re-enters the password. This must be implemented but will not be done until the system is set up at Samhjálp headquarters.

TCP wrappers

This security feature will encapsulate every VPN packet and rebuke every packet from computers not registered in the Linux IP tables. The TCP wrapper can look into packets and block them if they do not conform to the given rules. This security feature must be implemented but will not be done until the system is set up at Samhjálp headquarters.



Software Quality Attributes

The quality of the system is dependant on several factors. First to mention is the testing performed on the system. This has also to do with the portability and usability of the system. In this section, these aspects are viewed in detail.

Testing

This system must be thoroughly tested. Security, integrity and concurrency of the system must be unit tested, black and white box tested and if possible stress tested as well. While programming the jUnit system integrated into IntelliJ must be used to unit test the system. Black box testing must be done by sending data to server and expect the correct output. White box testing must include cyclic complexity measures.

Portability

Portability is an important factor for Jfactor. It must be possible to be able to run on any Java enabled browser on any machine. This is for future expansion of the system.

Usability

Usability is equally important for this software is scheduled to be internationalised. Since the system is written in Java, measures must be taken in the design of the system regarding the use of different languages. It must be easy to implement different locales so that user interfaces can be easily switched between languages.

Style guide

This style guide will contain all aspects needed to create the graphical user interfaces used for this system.

Templates

Width: 800 pixels

Height: 840 pixels

Background colour: White

Columns:

Fonts and sizes

Field titles: Verdana 10pt. Normal

Normal text: Verdana, 9pt. Normal, Black

Tabs: Arial, 8pt. Bold

Links: Verdana, 9pt. Normal,

Buttons

Hreinsa – Cancel:

Staðfesta – Submit:

Icons

TBD

Logo

Two logos will be used. First the Samhjálp logo which will be located in the top right corner of the title bar.

The letters must be coloured with #666666 gray. The dove should be white, and a white transparent shadow lifting the logo from the background.

Secondly the Jfactor logo which looks like this on the toolbar used in every page of the application:



This logo must be used as is no changes allowed without consulting the author.

The subtitle varies in regards to the user and uses the following font:

Subtitle: Verdana Italic CAPS, 14pt. Normal.

Sidebar

The sidebar, seen on the right of this page, can and will change in height but will not be allowed to change on the width. Fonts used are Verdana 10pt Bold in the title and Verdana 10pt. Normal in the body.

Ymislegt nýtsamt
Félagssbj ónustan
Meðferðar stofn anir
Lögfræðingar
Fangelsi
Landlæknir
Tryggingastofnun
Lögreglan
Doktor.is
Símaskrá.is
Pjóskrá
Íslandsbanki
Landsbankinn
Búnaðarbankinn
SÁÁ

Images

Width: TBD

Height: TBD

Resolution: 72 pixels p/inch.

Format: JPG

Appendix A: Glossary

TBD – To Be Determined

Appendix B: Issues List

- Finish sketching out the use cases
- Look at hardware and software interfaces
- Work on the style guide.
- Server / database connection, does it work? Needs to be done ASAP.
- Familiarise myself on TCP wrappers.

Appendix C: Kennitala verification formula

DOB							Random numbers		Verification number		Century	
1	8	0	4	6	5	5	5	3	7	9		
3	2	7	6	5	4		3	2				
3	16	0	24	30	20	15	6		114	Total		
											MOD 4 remainder	
11-4=											7	Verification no.

The kennitala is structured by ten numbers that are structured in the following way:

Two for day(1-2), two for month(3-4), two for year(5-6). The next two numbers (7-8) are randomly chosen.

The next number (9) is the verification number and the last (10) number denotes the century a person is born in. 9 for 1900-1999, 0 for 2000-2099.

The verification number is calculated in the following manner:

1. Write the first eight numbers of the kennitala into a table.
2. Write the numbers 2-7 in reverse order from right to left.
3. Multiply each column and add those sums together.
4. Use **modulus 11** on that sum.
5. Subtract the remainder from 11 and you have found the verification number.

If a verification number comes out as 11 it should be changed to 0.

Appendix B

Analysis documents

CONTENTS

Overview	76
Samhjálp - The organisation.....	77
Resources	77
1. Financial	77
2. Human	77
3. Groups	78
4. Real-estates.....	78
5. Computers	78
6. Musical equipment	78
7. Spiritual	79
Activities	79
Client's arrival for treatment at HLK	79
1. Applications.....	79
2. Morning meetings.....	79
3. Applicant's arrival	79
4. Step down	79
Counsellor's tasks after step down	80
5. Counsellor's checklist.....	80
6. Second interview – Analysis/Diagnosis	80
7. Third interview – Treatment goals.....	80
8. Progress interviews.....	81
Treatment manager's task.....	81
1. Admission	81
2. Discharge	81
3. Prescriptions form	82
4. Medical allowances certificate	82
5. Confirmation certificate	82
6. Lay days report at HLK.....	82
7. Applications and handling.....	83
8. Inmate report	83
9. Monthly reports.....	83
Doctor's tasks	83
Watchman's tasks	83
Pharmacist's tasks	84
Other processes.....	84

1.	Overall and Office – processes.....	84
2.	Support shelter - processes.....	85
3.	M-18 – processes.....	85
4.	M-20 – processes.....	86

Goals 86

1 Overview

This system, when finished, will be big. I'm thinking at least 2 to 3 years to complete it.

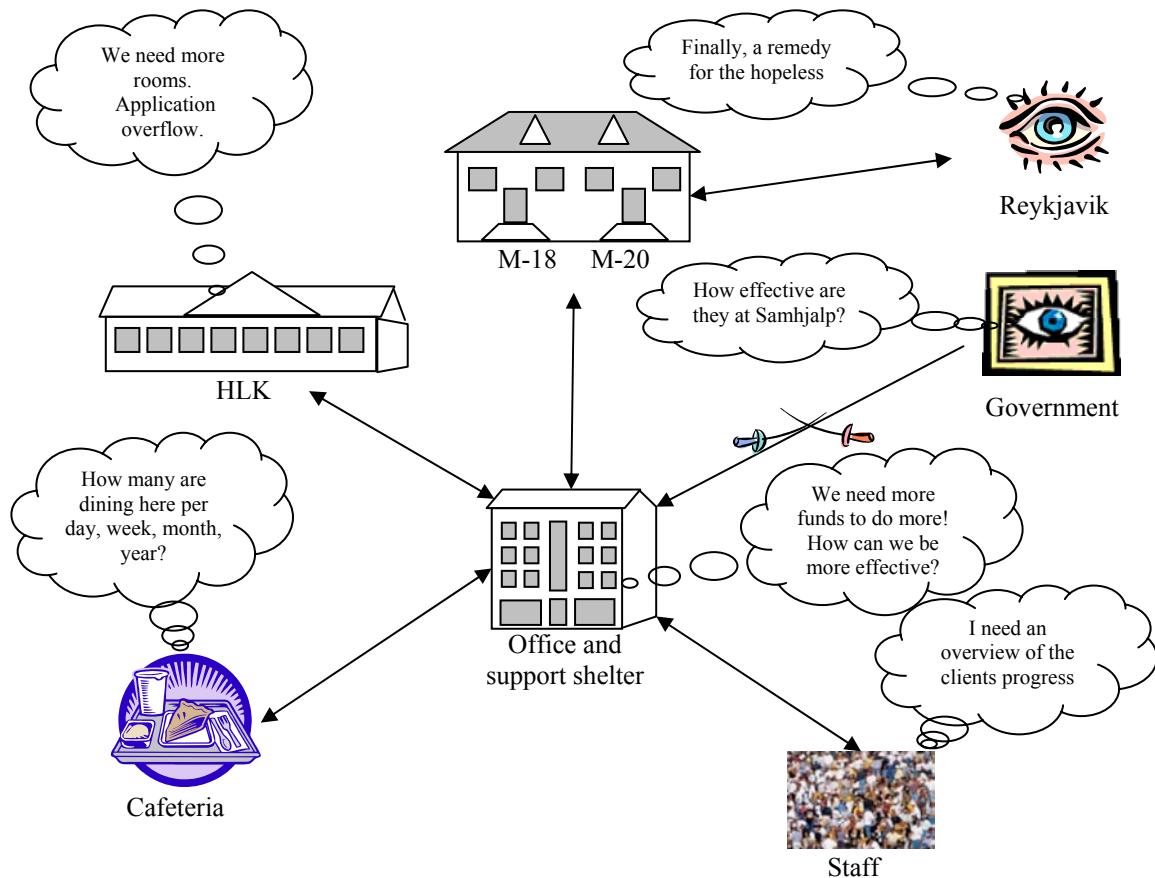
The system needs to span most of Samhjálp's work, including the treatment processes at Hlaðgerðarkot, which from now on will be abbreviated HLK, Stoðbýli, which is a support shelter for clients, M-18 and M-20. M-20 is a support shelter for clients with twofold diagnosis i.e. alcoholism and mental problems, and M-18 is for clients that come from any treatment centre and are willing to continue in a rehab program(M-18). This system needs also to provide the office with tools to create inside information like success of both staff and clients; and outside information like prices of drugs, types of drugs, quantity, including general statistical information about clients that have been treated at Samhjálp.

Other interesting tools for the office to have access and control over are: Job descriptions, job contracts, files on staff, staff interviews, meeting (church) attendance, AA attendance, also the attendance at the women's group, Dorkas (women at Samhjálp) meetings, Ungsam (young people at Samhálp) meetings and celebration meetings.

Now, let's take a closer look at the organisation.

2 Samhjálp - The organisation

3 Rich picture of Samhjálp



4 Resources

Samhjálp's resources are various. Here below is a listing of the most common ones.

Financial

- Government funds
- Support members of Samhjálp

Human

Presently there are 27 staff members and two contractors. Some of the staff members are part time.

- Counsellors (4 will be 5) – follow clients both through and beyond the treatment.
- Administrator (1) – handles resources and keeps track of the welfare of both clients and staff. They optimize the staff resources and organise every bit of the treatments for maximum efficiency and result.

- Treatment managers (2) – maintain daily function of the treatment centre.
- Office people – manage resources.
- Watchmen – keep track of all activity coming to or going from the treatment centre.
- Kitchen staff – make food for staff at HLK and the cafeteria.
- Doctor (contractor) – examines the clients and gets them into stable condition. Provides the clients with all necessary medication to maintain the clients' health.
- Pharmacist (contractor) – is located at a pharmacy. Provides the treatment centre with medication according to the doctor's prescriptions.
- Personal experience of staff – most of Samhjálp's staff have personal experience of addiction of various kind. Either them selves or close relatives.
- Volunteers – not on payroll but do all sorts of tasks not relevant to the system.

Groups

- Marita - preventive action against addiction. Supported by the police and social services.
- UngSam - focuses on the young individuals that come for treatment at Samhjálp.
- Dorkas - Group of women that have monthly fellowship

Real-estates

- HLK - treatment centre
- Hverfisgata 42 – Samhjálp's offices and support shelter. All individuals that stay in this support shelter come from HLK after going through treatment.
- Hverfisgata 44 - Cafeteria (food distribution)
- Miklabraut 18 (M-18) - Support home for individuals that have gone through a treatment in other treatment centres than Samhjálp. It is owned by Reykjavík social services but run by Samhjálp. Reykjavík social services choose the individuals in agreement with Samhjálp's administrators.
- Miklabraut 20 (M-20) - Support home for individuals with twofold diagnosis - i.e. alcoholism and mental disorders. Owned by Reykjavík social services but run by Samhjálp. Reykjavík social services choose the individuals in agreement with psychiatric doctors, counsellors and Samhjálp's administrators.
- More to come...

Computers

- Five computers at the offices (Hverfisgata 42)
- Five computers at HLK
- One at M-18
- One at M-20

Musical equipment

- Piano
- Mixer console
- Microphones cords
- Multi core sound cable
- Organ
- Speakers
- Monitors

Spiritual

- Meetings – open for everyone.
- Fellowship – positive talks and strengthening of friendship bonds.
- Lectures / Bible studies – for clients building up learning regarding their disease.
- Prayer – individual / public prayer meetings.

5 Activities

6 Client's arrival for treatment at HLK

Applications

- a. Clients must call every day till they are admitted for treatment. This is done to ensure only those who really want treatment will have most chance of being admitted. This is also needed since only limited number of beds (30) are available and paid for by the government
- b. An account of every call to HLK is held by the **watchman**. A watchman is a client who is trusted by the supervisors for tasks like taking phone calls and recording other activity like who come and go from the treatment centre. This is done to make any incidents traceable.

Morning meetings

Treatment supervisors and counsellors go over the list and decide which applications should be accepted if beds are free for use.

Some get a (+) or “Adó”, meaning that the client is finishing a treatment elsewhere, is in a hospital, in jail, or has any other valid reason. These clients do not need to call every day. Their applications are automatically renewed.

Applicant is notified when he/she is admitted.

Applicant's arrival

a. First interview

Treatment manager takes the first interview and fills out the application form, then takes a copy of it and passes it to the client's counsellor.

Step down

- a. According to doctors prescription and directions the **treatment manager** gives the new applicant a “knock-out dose” of drugs to help the client rest. This is usually a standard dose of drugs. ***ATH!!! What about Adó clients?***
- b. The drug prescription process for the step down procedure is as follows:
 - i. **Doctor** examines the client and decides the drug types and doses.
 - ii. Treatment managers get the prescriptions and fax them to the pharmacist.
 - iii. **Pharmacist** puts the drug doses in the client's drug container. Container is split into seven separate areas, one for each day of the week.
 - iv. **Treatment manager** receives the boxes and delivers each dose on right time.
- c. Step down takes 5-7 days on average but can exceed to 10-20 days. Counsellor takes part in the Step down procedure by monitoring the client and reporting any side effects or problems of the client's condition.

7 Counsellor's tasks after step down

Counsellor's checklist

- i. This checklist is an excellent tool for the counsellor to keep track of the progress of each of his client. (see Gátlisti Ráðgjafa). By using this tool he can follow diligently through all the steps without any fear of forgetting anything.

Second interview – Analysis/Diagnosis

- ii. The second interview is very important (Greiningarblað 2.viðtal). On this paper is written the current situation of the client in the following matters:
 - a. Habitation situation
 - b. Financial situation
 - c. Work situation
 - d. Family situation
 - e. Any active court cases
 - f. Type of addiction (or addictions)
 - g. Time of abuse period before treatment.
 - h. Abuse history
 - i. The first sip(app. date)
 - ii. Beginning of the drug abuse
 - iii. First treatment
 - iv. Former treatments
Where, how often, when last
 - v. Jail time
Where, how often, when last
 - i. Diagnosis summary
 - i. Stage 1 – six weeks of group therapy
 - ii. Challenge – 3-18 months long term treatment in three steps
(see Third interview – Treatment goals)
 - j. Counsellor's comments
- iii. The analysis paper as it is today, demands a lot of writing i.e. it is impossible to draw from it statistical information or other information with any meaning for the whole. So every part, from 1 to 8e needs to be reconsidered and structured into options that can be selected by the counsellor. Only the Counsellor's comments should remain a text area. Other areas should be either check boxes or number fields.
Part 9 - Diagnosis summary – remains a selective option, either stage 1 or challenge.

Third interview – Treatment goals

- i. This paper (Markmið meðferðar) is filled out by the client's counsellor after the diagnosis has been established and confirmed.
- ii. Treatment time after diagnosis:
 1. Stage 1 – six weeks of group therapy
 2. Challenge – 3-18 months long term treatment in three steps.
 - i. Step 1: Six weeks group therapy
 - ii. Step 2: Six weeks where all courses need to be completed and six weeks of group therapy.
 - iii. Step 3: 4-12 weeks, were occupational skills are trained and group work is decreased proportionally to actual work.
- iii. Counsellor then fills out the following parts on behalf of the client:
 3. Living conditions
 - iv. What changes do I need to make?
 - v. Ways to achieve goal (with reference dates). 1,2 or 3 steps.

4. Finance
 - vi. What is the real condition of my finances?
 - vii. Ways to achieve goal (with reference dates). 1,2 or 3 steps
5. Occupational matters
 - viii. What changes do I need to make?
 - ix. Ways to achieve goal (with reference dates). 1,2 or 3 steps
6. Family matters
 - x. What changes do I need to make?
 - xi. Ways to achieve goal (with reference dates). 1,2 or 3 steps
7. Counsellors comments

Progress interviews

- i. Are in four stages. Progress interviews – once a week. Minimum 5 interviews in six weeks of treatment.
- ii. Long term treatment (Challenge). Interview minimum once a week.
- iii. Create an application with the client for a stay at the Support shelter in concordance with the support shelter manager.
Discharge plan created.
- iv. Meeting with the counsellor, support shelter manager and the client.

8 Treatment manager's task

Admission

When client arrives, the treatment manager fills out a Admission form. This form is numbered starting at 1 in the beginning of a new year. The information entered on this form is as follows:

- Admission date
- Kennitala
- Full name
- Postcode
- Address
- City
- Job title
- Closest relative name
- Closest relative phone number
- Clients phone number
- Marital status
- Beneficiary at Tryggingastofnun ríkisins
- Invalidity % or Senior citizen
- Treatment history
- Period of intoxication
- What drugs? – to evaluate the step down process.
- Other diseases (other than alcoholism) / Drugs / Comments

Discharge

The treatment manager comes in when a client leaves the treatment centre and fills out the following information.

Discharge can be for four different reasons.

- With consultation = Consultation with a counsellor

- To Samhjálp's support shelter = Clients goes directly to Hverfisgata 42
- Without consultation = Leave treatment by themselves (Walk out).
- Dismissal = Removed from treatment for the cause of intoxication while in treatment, or for violently breaking the rules.
- Discharge date
- Counsellors comments – behavioural comments on the client and reason for leaving the treatment. (ATH!!! stöðlun)
- Other things needed here is to know where the clients are going from HLK. Are they going back to:
 - The street?
 - M-18?
 - Rented apartment?
 - Home?
 - Some relative, if so then whom?

Prescriptions form

The treatment manager receives the prescriptions and faxes to the pharmacist. This form is currently created in Microsoft Excel and the information inserted there is the following:

- Client's name
- Kennitala
- Date
- Drug name and type used for step down if that is the case
- Additional drug listing that client must take while in treatment.

This part of the system will however not be implemented at this time.

Medical allowances certificate

The information needed on this certificate is the following:

- Name of client
- Kennitala
- Address
- Admission date or the date were the doctor examines the client
- Period of days till discharge

Confirmation certificate

A confirmation certificate is needed for most clients to send to various parties regarding the client's treatment. This is a simple standard statement that will be available for printing from the system.

Lay days report at HLK

This is a monthly report which lists all clients that are presently in treatment. Every day that a client stays for treatment is marked. Information entered on this report is as follows:

- Admission date
- Kennitala
- Name
- X entered for each day a client is in treatment
- Marked if the client is a invalid or a senior citizen.

- Discharge date

Applications and handling

The treatment manager takes care of recording all applications that come during the day. Recently all applications were taken and put on a special monthly list – i.e. how many individuals are calling in their applications, not how often. For example if an individual starts calling in, in the beginning of the month, then stops and starts again at the end of the month, he is only counted once.

Inmate report

This is a file over all clients currently in treatment. Information entered is as follows:

- What group is he in
- Challenge route or not
- What jobs or roles are they doing at the treatment
- Medication if any
- Admission date – it happens if a client gets a leave of absence or needs to go to a hospital that he is discharged and then re-entered when he comes back
- Second Admission date
- Counsellor's initials

Monthly reports

In this report there is some calculation involved. Information like number of males and females in treatment per month, number of applications, Admissions, individuals and lay days need to be calculated and printed out for the government officials.

That will give an overview if the treatment centre is in debit or credit. HLK is allowed to have 30 beds used per month.

9 Doctor's tasks

The doctor examines the client when he has arrived and fills out the following information:

- Types of drug abuse
- Current health status
- Health history
- Prescriptions
 - Drug id
 - Drug name
 - Drug type
 - Drug dose
- Interviews if necessary

This part of the system will however not be implemented at this time.

10 Watchman's tasks

The watchman records every phone call and every visit to HLK. The information entered by him is as follows:

- Phonecalls (Caller's name, person requested, time of call),
- Visits (Visitor's name, person requested, time of arrival, time of departure).

This part of the system will however not be implemented at this time.

11 Pharmacist's tasks

The pharmacist must access the prescriptions assigned by the doctor, for him to be able to generate the suitable drug doses for the clients. Fields accessed by the pharmacist are therefore:

- Prescriptions
 - Doctor's id
 - Doctors name
 - Drug id
 - Drug name
 - Drug type
 - Drug dose

This part of the system will however not be implemented at this time.

12 Other processes

There are many other processes that must be considered to complete the whole system. However, in this project I will not consider them in detail, just mention them briefly.

Overall and Office – processes

These processes have not yet been analysed in detail. However, the office managers will access the clients and many (but not necessarily all) of the information on the clients. They are more interested in the general information capability of the system. Information like graphs and statistics pulled out from the database.

This functionality will not be implemented at this time. The needed information will be already in the database so all that is needed will be the views and the software to display it properly.

However, ideas on the possible usage of the system regarding the administration are as follows:

- Basic information hand-in to the government
- Who are not finishing the treatment
- Progress measurement
- Support for counsellors
 - How is each counsellor doing?
 - How are their clients doing?
- Age distribution of clients
- What narcotics are currently in use?
 - Wine
 - Cannabis
 - Morphine
 - Tobacco
 - Etc.
- Current and past prices of narcotics
- Fewer admissions = longer time in treatment
- Where are clients going after treatment?

- What is in common between clients – statistics

Other information needed and which is more related to the office management are as follows:

- Inner information on counsellors and clients
- Outer information like what narcotics are currently in use and the quantity?
- Job descriptions
- Job contracts
- Staff interviews
- Meeting attendance
- AA attendance
- Listing on who are participating in:
 - Dorkas
 - UngSam
 - Women's group

Support shelter - processes

The information that will be provided here are the names of the clients and their progress at the shelter. The relevant information can be derived from the database except the progress data that will be entered during the client's stay at the shelter.

M-18 – processes

These processes will be similar to the HLK's 2nd and 3rd interviews since the clients are not coming from HLK but from other treatment centres. However, there is no step down process involved since the clients are coming fresh from the treatment centre to the shelter. The admission steps are as follows:

The clients is introduced to the shelter. Date:

Analysis\Diagnosis paper filled out and these are the fields filled out with date:

- a. What doctor do you have?
- b. Do you have a psychiatrist\specialist?
- c. What medication are you using?
 - Name of medication
 - Size of dose
 - Used for how long?
- d. Do you receive any assistance?
- e. Are you interested in utilising Christianity in your life?
- f. Who is your social worker?
- g. Any agreement on social counselling?
- h. Habitation situation
- i. Financial situation
- j. Work situation
- k. Family situation
- l. Any active court cases
- m. When did you last abuse alcohol\drugs?
- n. Type of addiction (or addictions)
- o. Time of abuse period before treatment.
- p. Abuse history
 - The first sip(app. date)
 - Beginning of the drug abuse
 - First treatment

- Former treatments
Where, how often, when last
 - Jail time
Where, how often, when last
- q. Counsellor's comments

Work plan created. Date: _____

Objectives set. Date: _____

Progress interviews are then used to follow up on the client in his rehabilitation. Fields in the table for the progress interviews are:

Interview nr.

Date

Starts at

Ends at

Signature

Just like the analysis paper at HLK, it demands a lot of writing i.e. it is impossible to draw from it statistical information or other information with any meaning for the whole. So every part, from 1 to 5e needs to be reconsidered and structured into options that can be selected by the M-18's administrator. Only the Counsellor's comments should remain a text area. Other areas should be either check boxes or number fields.

These processes will not be implemented into the final year project, but will be in the near future.

M-20 – processes

These processes will be similar to the M-18's since the clients are not coming from HLK. These clients have twofold diagnosis, i.e. alcoholism and some type of mental disorder(s). However, I will not do any more analysis on M-20 in this project. That will be implemented sometime in the future.

13 Goals

- To return people into society as healthy, sober and strong productive individuals.
- To build up an organisation that will produce the best results in fighting addiction of any kind.
- To build up good communication between government, health organisations and other treatment centres.
- To fight and conquer any form of addiction.

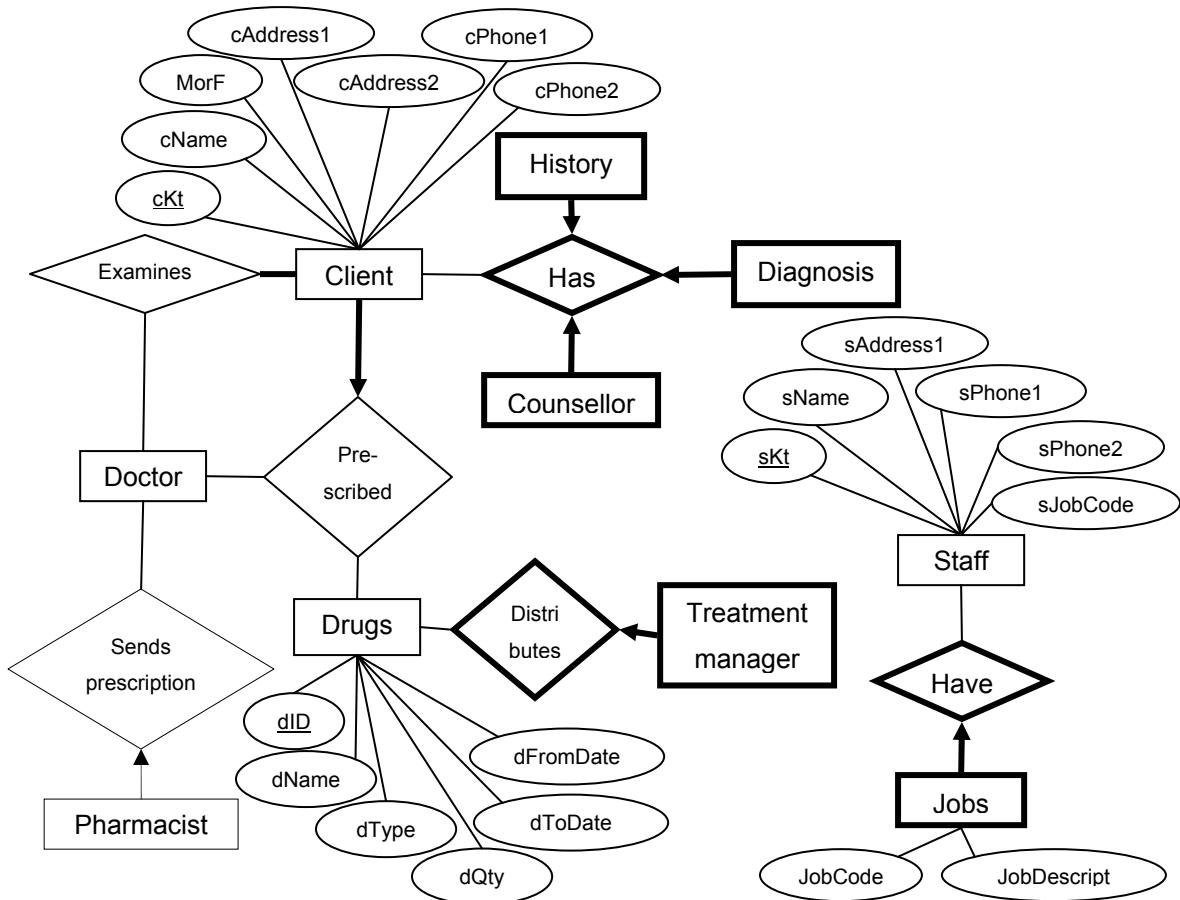
Appendix C

System Design

In this document are ideas and sketches of the system design for Samhjálp.

Database design

ER diagram – client and staff



This ER diagram was one of the first to be drawn in this project but has been deprecated. The database design has changed a lot since then and the newest version is in the Appendix A.

1. Database possibilities

The database used in this project is InterBase 6.0 from Borland. This is an open source product that was cancelled by Borland but picked up by the IBPhoenix group which has been working on its development including a server called Firebird. Below is a list of features that are contained in InterBase 6.0.

INTEGRITY

- Declarative Primary Key.
- Declarative Foreign Key.
- Cascade Declarative Referential Integrity.
- Domain and column-level Check constraints.
- Trigger procedures with the following features:
 - Unlimited triggers per record change.
 - Invoked before or after record insertion, deletion, or update.
 - Multiple triggers per action, optionally ordered.
 - Forward-chaining (cascading triggers).

CONCURRENCY CONTROL

- Optimistic locking.
- Data isolation levels: read consistency, read committed, and cursor stability.
- Shared, and protected lock types for explicit table-level locking.

AVAILABILITY

- Online backups.
- Immediate recovery after failure.

DISTRIBUTED DATABASE

- Simultaneously connected databases - limited only by hardware.
- Automatic distributed transaction processing via two-phase commit.

DATATYPES

- Character (fixed/variable length): up to 64Kb per field.
- Integer (8, 16, 32 & 64 bit).
- Floating point: single and double precision.
- Date & time: Jan. 1, 100 to Dec. 11, 5491.
- Date, Time, and Timestamp.
- Fully Year 2000 compliant.
- Multidimensional arrays; up to 16 dimensions per column.
- BLOB: unlimited size.
- Import and export of ASCII fixed-length data.
- BLOB filters for compressing or translating BLOB field data.

STANDARDS

- ANSI SQL-92 entry-level conformant.
- ODBC rev. 2.0 (16-bit).
- ODBC rev. 3.0 (32-bit).
- UNICODE compliant.

DATABASE CAPACITY

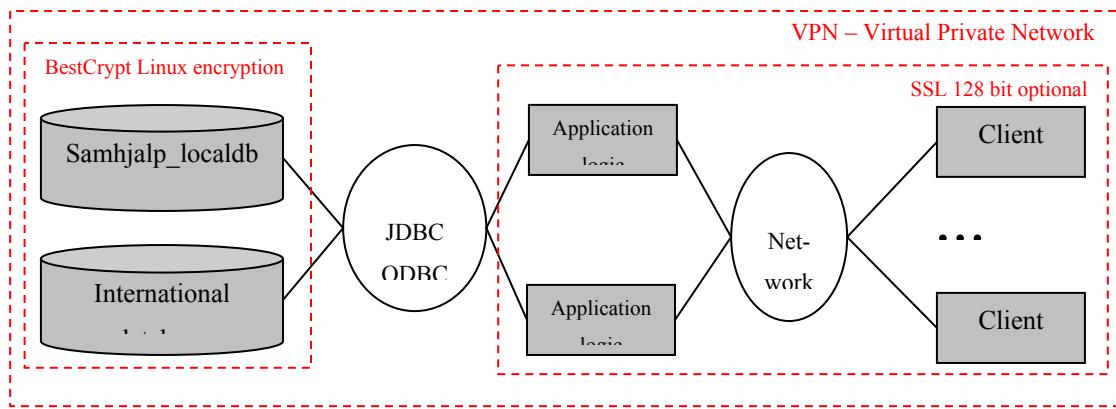
- Maximum number of rows per table: approximately 2 billion.
- Maximum size of a table: limited by system resources.
- Maximum number of databases per system: limited by system resources.
- Maximum number of active users possible per system: limited by system resources.
- Maximum number of tables per database: 64Kb.
- Maximum row size (excluding BLOB): 64Kb.

2. Design ideas

System security issues

Standard login and password submission should be implemented. The 128 bit secure https protocol must be implemented. Every user must have well specified admission rights and see only the fields needed for him to do his job.

The system must use the three-tier architecture.



This will provide the system with easy maintainability and high security.

Searching for clients

When a user enters the system, he can search for clients by name or kennitala. If it is chosen to search by name, the user enters the name in the name field. While the name is being typed the system should be able to come up with suggestions collected from the database to speed up the search. If search is done by kennitala the system should come up with suggestions collected from the database. This however might be a problem since this is a web based application and could raise questions about personal information safety.

When a specific client has been found the system needs to collect all the information concerning the client and make it available in an easy to use way. For example a list of links containing information on all previous treatments in Hlaðgerðarkot and other treatments could be shown.

Layout ideas

Use similar layout as www.netbanki.is. Sample screenshots are provided here below. The layout design will be created in Macromedia Fireworks and HTML implementation in Macromedia Dreamweaver.



The login window into the system

The screenshot shows a web-based application for managing client admissions. At the top, there's a header with the Jfactor logo and the text "UMSJÓNARMADURINN". Below the header, a navigation bar includes links for "Heim", "Innlögn" (selected), "Viðtal 1", "Viðtal 2", "Viðtal 3", and "Útskrift". On the right side of the header, there's a "Sömhjálp" icon.

The main form area is titled "Ymislegt nytíamt" (General Information). It contains several input fields and dropdown menus:

- Kennitala:** A text input field.
- Nafn:** A text input field.
- Heimili:** A text input field.
- Póstr/Staður:** A text input field.
- Sími skj. st.:** A text input field.
- Atvinnna:** A text input field. To its right is a question: "Er skj ólstæðingur í vinnu?" (Is it an involuntary placement?).
- Nafn settingja:** A text input field.
- Símanúmer settingja:** A text input field.
- Hjúskaparstaða:** A text input field.
- Öryrki:** A dropdown menu with options "Já", "Nei", and "Örorku %".
- Elliflífeyrisþegi:** A dropdown menu with options "Já" and "Nei".
- Neyslutími fyrir innlögn:** A text input field. To its right is a question: "Hvaða vímugjáfar" (What kind of leave) followed by a text input field.
- Meðferðarsaga:** A section with three columns: "Hvar" (Where), "Hversu oft" (How often), and "Hvenær síðast" (Last time). It contains a list of locations with checkboxes:
 - Hlaðgerðarkot
 - Vogur
 - Staðarfell
 - Vík
 - Landspítali 33A
 - Gunnarsholt
 - Byrgið
 - Unglingameðferð/Geðdeild
 - Geðdeildir
 - Annað
- Aðarir sjúkdómar:** A text input field.
- Lýf við þeim sjúkdómum:** A text input field.
- Athugasemdir:** A large text input area.
- Nafn rádgjafa:** A text input field.
- Hreinsa:** A button.
- Staðfesta:** A button.

On the right side of the form, there are three input fields with labels: "Númer innlagnar" (Number of admissions), "Fjöldi umsókna fyrir rinnlögn" (Number of applications for admission), and "Dagsetning innlagnar" (Admission date).

This window is for the Treatment manager where he inserts information about a new client during the admission process.

Appendix D

Code listing

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
project\Jfactor\build.xml

```

1  <project name="SamhjalpRehabSystem" default="test" basedir=".">
2      <property name="buildDir" value="build"/>
3
4      <path id="project.build.classpath">
5          <fileset dir="lib">
6              <include name="**/*.jar"/>
7          </fileset>
8      </path>
9
10     <target name="createDatabase">
11         <sql driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost/mysql"
12           userid="root" password="">
13             CREATE DATABASE samhjalpdb;
14             commit;
15             grant all on samhjalpdb.* to yngvi
16         </sql>
17     </target>
18
19     <target name="dropDatabase">
20         <sql driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost/mysql"
21           userid="root" password="">
22             DROP DATABASE samhjalpdb;
23             commit;
24         </sql>
25     </target>
26
27     <target name="createDatabaseTables">
28         <sql driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost/samhjalpdb"
29           userid="yngvi" password="">
30             CREATE TABLE client
31             (
32                 kennitala CHAR(11) NOT NULL,
33                 client_id CHAR(4) NOT NULL,
34                 firstname VARCHAR(30),
35                 lastname VARCHAR(30),
36                 pnr SMALLINT REFERENCES postnr(pnr),
37                 address VARCHAR(30),
38                 phone_id VARCHAR(20),
39                 case_id INTEGER REFERENCES court_cases(case_id),
40                 has_a_job BOOL,
41                 haj_where CHAR(30),
42                 cjob_id CHAR(4) REFERENCES c_job_titles(cjt_id),
43                 cjob_company CHAR(30),
44                 cjob_position CHAR(20),
45                 cjob_started DATE,
46                 cjob_quit DATE,
47                 married BOOL,
48                 spouse_kt CHAR(11),
49                 married_to VARCHAR(30),
50                 divorced BOOL,
51                 div_how_often SMALLINT,
52                 c_law_marriage BOOL,
53                 single BOOL,
54                 no_of_children SMALLINT,
55                 rel_name VARCHAR(30) REFERENCES relatives(rel_name),
56                 own_apmt BOOL,
57                 rent_apmt BOOL,
58                 friend_rel_apmt BOOL,
59                 fr_where CHAR(30),
60                 homeless BOOL,
61                 hl_where CHAR(30),
62                 dept_id CHAR(10),
63                 dept_type CHAR(30),
64                 loan_id INTEGER,
65                 loan_owner VARCHAR(30),
66                 dept_amount DOUBLE PRECISION,
67                 analysis_result CHAR(20) REFERENCES analysis_picklist(a_picklist),
68                 first_sip DATE,
69                 first_drug_abuse DATE,
70             )
71         </sql>
72     </target>
73
74     <target name="copyDatabaseTables">
75         <sql driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost/samhjalpdb"
76           userid="yngvi" password="">
77             INSERT INTO client
78             SELECT *
79             FROM client
80             WHERE client_id = 1
81         </sql>
82     </target>
83
84     <target name="insertData">
85         <sql driver="com.mysql.jdbc.Driver" url="jdbc:mysql://localhost/samhjalpdb"
86           userid="yngvi" password="">
87             INSERT INTO client
88             SELECT *
89             FROM client
90             WHERE client_id = 1
91         </sql>
92     </target>
93
94 
```

```

69      first_treatment DATE,
70      reasons TEXT, <!--THE REASONS FIELD NEEDS TO BE ANALYSED BETTER FOR DATA
MINING REASONS-->
71      frequency INTEGER,
72      frequency_dwm CHAR(10),
73      qty INTEGER,
74      qty_dwm CHAR(10),
75      comments TEXT,
76      ip_id INTEGER REFERENCES ingestion_period(ip_id),
77      drug_id CHAR(10) REFERENCES drugs(drug_id),
78      step_down_id INTEGER REFERENCES step_down(step_down_id),
79      discharge_date DATE REFERENCES discharge(discharge_date),
80      PRIMARY KEY(kennitala)
81      )TYPE = MyISAM;
82
83      CREATE TABLE POSTNR
84      ( PNR SMALLINT NOT NULL,
85      CITY CHAR(20),
86      PRIMARY KEY(PNR)
87      )TYPE = InnoDB;
88
89      CREATE TABLE phone_type_picklist
90      ( ph_type CHAR(20) NOT NULL,
91      PRIMARY KEY(ph_type)
92      )TYPE = InnoDB;
93
94      CREATE TABLE relatives
95      ( rel_name VARCHAR(30) NOT NULL,
96      ckennitala CHAR(11) NOT NULL,
97      rel_communication TEXT,
98      relationship VARCHAR(20) REFERENCES rel_relations_picklist(relationship),
99      PRIMARY KEY(rel_name, ckennitala)
100     )TYPE = MyISAM;
101
102      CREATE TABLE rel_relations_picklist
103      ( relationship VARCHAR(20) NOT NULL,
104      PRIMARY KEY(relationship)
105      )TYPE = InnoDB;
106
107      CREATE TABLE c_job_titles
108      ( cjt_id CHAR(4) NOT NULL,
109      title VARCHAR(20),
110      c_job_id CHAR(4) REFERENCES client_jobs(c_job_id),
111      PRIMARY KEY(cjt_id)
112      )TYPE = InnoDB;
113
114      CREATE TABLE client_jobs
115      ( c_job_id CHAR(4) NOT NULL,
116      job_name VARCHAR(30),
117      job_description TEXT,
118      PRIMARY KEY(c_job_id)
119      )TYPE = InnoDB;
120
121      CREATE TABLE former_treatments
122      ( kennitala CHAR(11) NOT NULL,
123      ft_when_last DATE NOT NULL,
124      ft_where VARCHAR(20),
125      ft_hwo_often SMALLINT,
126      PRIMARY KEY(kennitala, ft_when_last)
127      )TYPE = InnoDB;
128
129      CREATE TABLE jailtime
130      ( jt_from DATE NOT NULL,
131      jt_to DATE NOT NULL,
132      kennitala CHAR(11) NOT NULL REFERENCES client(kennitala),
133      jail_time VARCHAR(30),
134      time_served TINYINT(1),
135      PRIMARY KEY(jt_from, jt_to, kennitala)
136      )TYPE = InnoDB;
137
138      CREATE TABLE court_cases
139      ( case_id INTEGER NOT NULL,
140      case_no INTEGER,
141      case_name VARCHAR(50),
142      verdict TEXT,
143      sentence TEXT,
144      case_active TINYINT(1),

```

```

145 PRIMARY KEY(case_id)
146 )TYPE = InnoDB;
147
148 CREATE TABLE ingestion_period
149 ( ip_id INTEGER NOT NULL,
150 ingest_period_from DATE,
151 ingest_period_to DATE,
152 PRIMARY KEY(ip_id)
153 )TYPE = InnoDB;
154 <!--Edited 11.02.04 -->
155 CREATE TABLE medication
156 ( id CHAR(10) NOT NULL,
157 name VARCHAR(20),
158 description TEXT,
159 PRIMARY KEY(id)
160 )TYPE = InnoDB;
161 <!--Created 11.02.04 -->
162 CREATE TABLE drugs
163 ( drug_id INTEGER NOT NULL,
164 drug_name VARCHAR(20),
165 description TEXT,
166 effect TEXT,
167 consequences TEXT,
168 )TYPE = InnoDB;
169 <!--Created 11.02.04 -->
170 CREATE TABLE drugs_abused
171 ( drug_id INTEGER NOT NULL,
172 kennitala CHAR(11) REFERENCES client(kennitala),
173 drug_name VARCHAR(20),
174 consumption_form VARCHAR(20),
175 primary key(drug_id, kennitala)
176 )TYPE = InnoDB;
177 <!--Created 11.02.04 -->
178 CREATE TABLE drug_prices
179 ( drug_id INTEGER NOT NULL REFERENCES drugs(drug_id),
180 kennitala CHAR(11) REFERENCES client(kennitala),
181 dose_prise INTEGER,
182 primary key(drug_id, kennitala)
183 )TYPE = InnoDB;
184
185 CREATE TABLE watchmen
186 ( kennitala CHAR(11) NOT NULL,
187 wffirstname CHAR(30),
188 wlastname CHAR(30),
189 ph_c_id INTEGER REFERENCES phonecalls_to_client(ph_c_id),
190 visit_id INTEGER REFERENCES visitations(visit_id),
191 PRIMARY KEY(kennitala)
192 )TYPE = InnoDB;
193
194 CREATE TABLE visitations
195 ( visit_id INTEGER NOT NULL,
196 kennitala CHAR(11) NOT NULL,
197 visitor_name VARCHAR(30),
198 why VARCHAR(200),
199 v_date DATE,
200 arrival_time TIMESTAMP,
201 departure_time TIMESTAMP,
202 PRIMARY KEY(visit_id, kennitala)
203 )TYPE = InnoDB;
204
205 CREATE TABLE phonecalls_to_client
206 ( ph_c_id INTEGER NOT NULL,
207 kennitala CHAR(11) NOT NULL,
208 caller_name VARCHAR(30),
209 why VARCHAR(200),
210 call_date DATE,
211 call_time TIMESTAMP,
212 PRIMARY KEY(ph_c_id, kennitala)
213 )TYPE = InnoDB;
214
215 CREATE TABLE step_down
216 ( kennitala CHAR(11) NOT NULL,
217 step_down_id INTEGER NOT NULL,
218 medication_given VARCHAR(30),
219 medication_qty INTEGER,
220 nr_of_dt_days INTEGER,
221 step_down_start TIMESTAMP,

```

```

222 PRIMARY KEY(kennitala, step_down_id)
223 )TYPE = InnoDB;
224
225 CREATE TABLE counsellors_comments
226 ( skennitala CHAR(11) NOT NULL REFERENCES STAFF(kennitala),
227 ckennitala CHAR(11) NOT NULL REFERENCES CLIENT(kennitala),
228 comments BLOB,
229 PRIMARY KEY(skennitala, ckennitala)
230 )TYPE = InnoDB;
231
232 CREATE TABLE staff
233 ( kennitala CHAR(11) NOT NULL,
234 staff_id CHAR(4) REFERENCES s_job_titles(staff_id),
235 s_job_id INTEGER,
236 firstname VARCHAR(30),
237 lastname VARCHAR(30),
238 pnr SMALLINT NOT NULL REFERENCES postnr(pnr),
239 address VARCHAR(30),
240 PRIMARY KEY(kennitala)
241 )TYPE = InnoDB;
242
243 CREATE TABLE s_job_titles
244 ( kennitala CHAR(11) NOT NULL,
245 staff_id CHAR(4) NOT NULL REFERENCES staff_jobs(s_job_id),
246 title VARCHAR(20),
247 PRIMARY KEY(kennitala, staff_id)
248 )TYPE = InnoDB;
249
250 CREATE TABLE staff_jobs
251 ( s_job_id CHAR(4) NOT NULL,
252 job_name VARCHAR(30),
253 job_description TEXT,
254 PRIMARY KEY(s_job_id)
255 )TYPE = InnoDB;
256
257 CREATE TABLE staff_phones
258 ( kennitala CHAR(11) NOT NULL,
259 phone_type CHAR(20) REFERENCES phone_type_picklist(ph_type),
260 number CHAR(20),
261 PRIMARY KEY(kennitala)
262 )TYPE = InnoDB;
263
264 CREATE TABLE client_phones
265 ( kennitala CHAR(11) NOT NULL,
266 phone_type CHAR(20) REFERENCES phone_type_picklist(ph_type),
267 number CHAR(30),
268 PRIMARY KEY(kennitala)
269 )TYPE = InnoDB;
270
271 CREATE TABLE jails
272 ( jail_name VARCHAR(30) NOT NULL,
273 jail_address VARCHAR(30),
274 jail_pnr SMALLINT REFERENCES postnr(pnr),
275 PRIMARY KEY(jail_name)
276 )TYPE = InnoDB;
277
278 CREATE TABLE courts
279 ( court_name VARCHAR(30) NOT NULL,
280 court_address VARCHAR(30),
281 court_pnr SMALLINT REFERENCES postnr(pnr),
282 PRIMARY KEY(court_name)
283 )TYPE = InnoDB;
284
285 CREATE TABLE treatment_centres
286 ( tc_name VARCHAR(20) NOT NULL,
287 tc_address VARCHAR(30),
288 tc_pnr SMALLINT REFERENCES postnr(pnr),
289 PRIMARY KEY(tc_name)
290 )TYPE = InnoDB;
291
292 CREATE TABLE diseases
293 ( kennitala CHAR(11) NOT NULL REFERENCES client(kennitala),
294 disease_name CHAR(30) NOT NULL REFERENCES disease_name_piclist(name),
295 diagnosis_date DATE,
296 description TEXT,
297 PRIMARY KEY(kennitala, disease_name)
298 )TYPE = InnoDB;

```

```

299
300     CREATE TABLE disease_name_picklist
301     (   name CHAR(30) NOT NULL,
302         PRIMARY KEY(name)
303     ) TYPE = InnoDB;
304
305     CREATE TABLE discharge
306     (   kennitala CHAR(11) NOT NULL,
307         discharge_date DATE NOT NULL,
308         discharge_reason VARCHAR(30) REFERENCES discharge_reason(reason),
309         comments_on_behaviour TEXT,
310         PRIMARY KEY(kennitala, discharge_date)
311     ) TYPE = InnoDB;
312
313     CREATE TABLE discharge_reason
314     (   reason VARCHAR(30) NOT NULL,
315         PRIMARY KEY(reason)
316     ) TYPE = InnoDB;
317
318     CREATE TABLE analysis_picklist
319     (   a_picklist CHAR(10) NOT NULL,
320         PRIMARY KEY(a_picklist)
321     ) TYPE = InnoDB;
322
323     CREATE TABLE client_reasons
324     (   kennitala CHAR(11) NOT NULL REFERENCES client(kennitala),
325         first_sip DATE,
326         first_sip_why VARCHAR(20),
327         first_drug_use DATE,
328         first_drug_use_why VARCHAR(20),
329         first_treatment DATE,
330         first_treatment_why VARCHAR(20),
331         PRIMARY KEY(kennitala)
332     ) TYPE = InnoDB;
333
334     CREATE TABLE reasons_picklist
335     (   picklist VARCHAR(20) NOT NULL,
336         PRIMARY KEY(picklist)
337     ) TYPE = InnoDB;
338
339     </sql>
340 </target>
341
342     <target name="init">
343         <delete dir="${buildDir}" />
344         <mkdir dir="${buildDir}" />
345     </target>
346
347     <target name="clearDatabase"
348 depends="dropDatabase,createDatabase,createDatabaseTables">
349     </target>
349
350     <target name="compile" depends="init">
351         <javac srcdir="src" destdir="${buildDir}" deprecated="off" debug="on">
352             <classpath refid="project.build.classpath"/>
353         </javac>
354     </target>
355
356     <target name="createWarFile" depends="compile">
357         <war destfile="jfactor.war" webxml="web.xml"
358             update="false">
359             <fileset dir="webfiles"/>
360             <lib dir="lib"/>
361             <classes dir="build"/>
362         </war>
363     </target>
364
365     <target name="createZipFile">
366         <zip destfile="YngviRafnYngvason.zip" includes="Jfactor/lib" basedir="."
367             update="false" excludes="build/">
368             <zipfileset dir="src" prefix="Yngvi/src"/>
369             <zipfileset dir="lib" prefix="Yngvi/lib"/>
370             <zipfileset dir=".." includes="LibraryProject.ipr"
371 fullpath="Yngvi/Jfactor.ipr"/>
371             <zipfileset dir=".." includes="LibraryProject.iws"
372 fullpath="Yngvi/Jfactor.iws"/>
372             <zipfileset dir=".." includes="README.txt" fullpath="Yngvi/README.txt"/>

```

```
373      <zipfileset dir=". " includes="build.xml" fullpath="Yngvi/build.xml"/>
374  </zip>
375 </target>
376
377 <target name="jUnit">
378   <junit printsummary="yes" showoutput="yes">
379     <test name="src.is.unak.library.test.SamhjalpTest"/>
380     <classpath>
381       <pathelement location="${buildDir}"/>
382       <pathelement path="${java.class.path}"/>
383     </classpath>
384     <formatter type="plain" usefile="false"/>
385   </junit>
386 </target>
387
388 <target name="test"
depends="createDatabase,createDatabaseTables,compile,jUnit,dropDatabase">
389   </target>
390
391 </project>
```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
project\Jfactor\src\is\unak\webapp\test\JfactorTest.java
```

```

1  package is.unak.webapp.test;
2
3  import junit.framework.TestCase;
4  import java.sql.Date;
5  import java.sql.Connection;
6  import is.unak.webapp.activerecords.accessories.PostnrOps;
7  import is.unak.webapp.activerecords.ItemNotFoundException;
8  import is.unak.webapp.activerecords.staff.StaffJobs;
9  import is.unak.webapp.activerecords.staff.Staff;
10 import is.unak.webapp.service.PersonnelServiceDesk;
11 import is.unak.webapp.service.ClientServiceDesk;
12 import is.unak.webapp.service.AccessoriesServiceDesk;
13 import is.unak.webapp.database.DatabaseConnection;
14
15 public class JfactorTest extends TestCase {
16
17     PersonnelServiceDesk pServDesk;
18     ClientServiceDesk cServDesk;
19     AccessoriesServiceDesk aServDesk;
20     Date today;
21     Connection conn;
22
23     public JfactorTest(String string) {
24         super(string);
25     }
26
27     //This method sets up the services and the database connection
28     protected void setUp() throws Exception {
29         pServDesk = new PersonnelServiceDesk();
30         cServDesk = new ClientServiceDesk();
31         aServDesk = new AccessoriesServiceDesk();
32         DatabaseConnection db = new DatabaseConnection();
33         conn = db.openConnection();
34     }
35
36     //This method performs a rollback such that any data that is inserted into the
37     //database is never committed
38     //but cleared or aborted so that the database remains untouched after the test has
39     //completed.
40     protected void tearDown() throws Exception {
41         super.tearDown();
42         conn.rollback();
43         conn.close();
44     }
45
46     //The first three tests affect the staff table
47     public void testAddStaffMember() throws Exception {
48         Staff staff = new Staff();
49         staff.setKennitala("180465-5379");
50         staff.setStaffId(1);
51         staff.setJobId(33);
52         staff.setFirstName("Yngvi");
53         staff.setLastName("Yngvason");
54         staff.setPnr(600);
55         staff.setAddress("Munkabverárstræti 1");
56         staff.saveInsert(conn);
57         assertEquals("DB should initially be empty.", 0,
58                     pServDesk.getStaffMembers(conn).size());
59         pServDesk.addStaff(conn, staff);
60         assertEquals("The system should have added 1 staff member", 1,
61                     pServDesk.getStaffMembers(conn).size());
62     }
63
64     public void testChangeStaffInfo() throws Exception {
65         Staff staff = new Staff();
66         staff.setKennitala("180465-5379");
67         staff.setStaffId(1);
68         staff.setJobId(33);
69         staff.setFirstName("Yngvi");
70     }

```

```

66         staff.setLastName("Yngvason");
67         staff.setPnr(600);
68         staff.setAddress("Munkapverárstræti 1");
69         staff.saveInsert(conn);
70         pServDesk.addStaff(conn, staff);
71         pServDesk.changeStaffMember(conn, "123456-7890", 2, 4, "Eydis", "Rut", 603,
72         "Bondabaer Eyjafirdi");
73         assertEquals("Eydis", pServDesk.getStaffName(conn, "123456-7890"));
74     }
75
76     public void testDeleteStaffMember() throws Exception {
77         Staff staff = new Staff();
78         staff.setKennitala("123456-7890");
79         staff.setStaffId(1);
80         staff.setJobId(33);
81         staff.setFirstName("Haukur");
82         staff.setLastName("Hauksson");
83         staff.setPnr(600);
84         staff.setAddress("Skogarhlid 34");
85         staff.saveInsert(conn);
86         pServDesk.addStaff(conn, staff);
87         Staff staff2 = new Staff();
88         staff2.setKennitala("180465-5379");
89         staff2.setStaffId(1);
90         staff2.setJobId(33);
91         staff2.setFirstName("Yngvi");
92         staff2.setLastName("Yngvason");
93         staff2.setPnr(600);
94         staff2.setAddress("Munkapverárstræti 1");
95         staff2.saveInsert(conn);
96         pServDesk.addStaff(conn, staff2);
97         pServDesk.deleteStaffMember(conn, "180465-5379");
98         assertEquals("One row should have been deleted.", 1,
99         pServDesk.getStaffMembers(conn).size());
100    }
101
102    //Next three tests affect the postnr table. Method names speak for themselves.
103    public void testAddPnrAndCity() throws Exception {
104        PostnrOps pops = new PostnrOps();
105        pops.setPnr(101);
106        pops.setCity("Reykjavík");
107        pops.saveInsert(conn);
108        assertEquals("The system should have added one row", 1,
109        aServDesk.getPnrAndCity(conn).size());
110    }
111
112    public void testChangeCity(PostnrOps p) throws Exception {
113        aServDesk.addPnrAndCity(conn, p);
114        aServDesk.changeCity(conn, 103, "Reykjavík");
115        assertEquals("Reykjavík", aServDesk.getCityName(conn, 103));
116    }
117
118    public void testChangePnr(PostnrOps p) throws Exception {
119        aServDesk.addPnrAndCity(conn, p);
120        aServDesk.changePnr(conn, 603, "Akureyri");
121        assertEquals(603, aServDesk.getPostnumber(conn, "Akureyri"));
122    }
123
124    //Next three methods affect the staf_phone table.
125    public void testAddStaffPhone() throws ItemNotFoundException {
126        assertEquals("DB should initially be empty.", 0,
127        aServDesk.getStaffPhones(conn).size());
128        aServDesk.addStaffPhone(conn, "180465-5379", "GSM", "820-2825");
129        assertEquals("There should be one phone in the table", 1,
130        aServDesk.getStaffPhones(conn).size());
131    }
132
133    public void testChangeStaffPhoneType() throws ItemNotFoundException {
134        assertEquals("DB should initially be empty.", 0,
135        aServDesk.getStaffPhones(conn).size());
136        aServDesk.addStaffPhone(conn, "1804655379", "GSM", "820-2825");
137        aServDesk.changeStaffPhoneType(conn, "1804655379", "Heimasimi", "461-4363");
138        try {
139            assertEquals("Heimasimi", aServDesk.getStaffPhoneType(conn, "1804655379"));
140            assertEquals("461-4363", aServDesk.getStaffPhoneNumber(conn, "1804655379"));
141        } catch (ItemNotFoundException e) {
142            e.printStackTrace();
143        }
144    }

```

```

137         }
138     }
139
140     public void testDeleteStaffPhone() throws ItemNotFoundException {
141         assertEquals("DB should initially be empty.", 0,
aServDesk.getStaffPhones(conn).size());
142         aServDesk.addStaffPhone(conn, "1804655379", "GSM", "820-2825");
143         aServDesk.addStaffPhone(conn, "2804655379", "Heimasimi", "461-4363");
144         aServDesk.deleteStaffPhone(conn, "2804655379");
145         assertEquals("One row should have been deleted.", 1,
aServDesk.getStaffPhones(conn).size());
146     }
147
148     //Next three methods affect the client_phone table.
149     public void testAddClientPhone() throws Exception {
150         assertEquals("DB should initially be empty.", 0,
aServDesk.getStaffPhones(conn).size());
151         aServDesk.addStaffPhone(conn, "180465-5379", "GSM", "820-2825");
152         assertEquals("There should be one phone in the table", 1,
aServDesk.getStaffPhones(conn).size());
153     }
154
155     public void testChangeClientPhoneTypeAndNumber() throws ItemNotFoundException {
156         assertEquals("DB should initially be empty.", 0,
aServDesk.getStaffPhones(conn).size());
157         aServDesk.addStaffPhone(conn, "1804655379", "GSM", "820-2825");
158         aServDesk.changeStaffPhoneType(conn, "1804655379", "Heimasimi", "461-4363");
159         try {
160             assertEquals("Heimasimi", aServDesk.getStaffPhoneType(conn, "1804655379"));
161             assertEquals("461-4363", aServDesk.getStaffPhoneNumber(conn, "1804655379"));
162         } catch (ItemNotFoundException e) {
163             e.printStackTrace();
164         }
165     }
166
167     public void testDeleteClientPhone() throws ItemNotFoundException {
168         assertEquals("DB should initially be empty.", 0,
aServDesk.getClientPhones(conn).size());
169         aServDesk.addClientPhone(conn, "1804655379", "GSM", "820-2825");
170         aServDesk.addClientPhone(conn, "2804655379", "Heimasimi", "461-4363");
171         aServDesk.deleteClientPhone(conn, "2804655379");
172         assertEquals("One row should have been deleted.", 1,
aServDesk.getClientPhones(conn).size());
173     }
174
175     //Next three methods affect the phone_picklist table
176     public void testAddToPhonePicklist() throws ItemNotFoundException {
177         assertEquals("DB should initially be empty.", 0,
aServDesk.getPhonePicklist(conn).size());
178         aServDesk.addToPhonePicklist(conn, "GSM");
179         assertEquals("One row should have been added", 1,
aServDesk.getPhonePicklist(conn).size());
180     }
181
182     public void xtestChangePhonePicklist() throws ItemNotFoundException {
183         assertEquals("DB should initially be empty.", 0,
aServDesk.getPhonePicklist(conn).size());
184         aServDesk.addToPhonePicklist(conn, "GSM");
185         aServDesk.changePhonePicklist(conn, "Heimasimi");
186         try {
187             assertEquals("Heimasimi", aServDesk.getPicklistItem(conn, "Heimasimi"));
188             assertEquals("Row should be changed", 1,
aServDesk.getPhonePicklist(conn).size());
189         } catch (Exception e) {
190             e.printStackTrace();
191         }
192     }
193
194     public void testDeletePhonePicklist() throws ItemNotFoundException {
195         assertEquals("DB should initially be empty.", 0,
aServDesk.getStaffPhones(conn).size());
196         aServDesk.addToPhonePicklist(conn, "GSM");
197         aServDesk.deletePhonePicklist(conn, "GSM");
198         assertEquals("One row should have been deleted.", 0,
aServDesk.getPhonePicklist(conn).size());
199     }
200

```

```

201      //Next three methods affect the staff_job table
202      public void testAddStaffJob() throws ItemNotFoundException {
203          assertEquals("DB should initially be empty.", 0,
204              pServDesk.getStaffJobs(conn).size());
205          pServDesk.addStaffJob(conn, "R-33", "Radgjafi", "Counsellor handles clients
206          during and after treatment on a regular basis.");
207          assertEquals("One row should have been added", 1,
208              pServDesk.getStaffJobs(conn).size());
209      }
210
211      public void testChangeStaffJob() throws ItemNotFoundException {
212          //    assertEquals("DB should initially be empty.", 0,
213          pServDesk.getStaffJobs(conn).size());
214          pServDesk.addStaffJob(conn, "R-33", "Radgjafi", "Counsellor handles clients
215          during and after treatment on a regular basis.");
216          pServDesk.changeStaffJob(conn, "R-33", "Vaktmadur", "Watchmen record all visits
217          and phonecalls to HLK.");
218          try {
219              StaffJobs staffJob = pServDesk.getStaffJob(conn, "R-33");
220              assertNotNull(staffJob);
221              assertEquals("Vaktmadur", staffJob.getJobName());
222          }
223
224          public void testDeleteStaffJob() throws ItemNotFoundException {
225              assertEquals("DB should initially be empty.", 0,
226                  pServDesk.getStaffJobs(conn).size());
227              pServDesk.addStaffJob(conn, "R-33", "Radgjafi", "Counsellor handles clients
228              during and after treatment on a regular basis.");
229              pServDesk.deleteStaffJob(conn, "R-33");
230              assertEquals("One row should have been deleted.", 0,
231                  pServDesk.getStaffJobs(conn).size());
232
233          //Next three methods affect the staff_job table
234          public void testAddStaffJobTitle() throws ItemNotFoundException {
235              assertEquals("DB should initially be empty.", 0,
236                  pServDesk.getStaffJobTitles(conn).size());
237              pServDesk.addStaffJobTitle(conn, "1804655379", "R-33", "Radgjafi");
238              assertEquals("One row should have been added", 1,
239                  pServDesk.getStaffJobTitles(conn).size());
240
241          public void testChangeStaffJobTitle() throws ItemNotFoundException {
242              assertEquals("DB should initially be empty.", 0,
243                  pServDesk.getStaffJobTitles(conn).size());
244              pServDesk.addStaffJobTitle(conn, "1804655379", "R-33", "Radgjafi");
245              pServDesk.changeStaffJobTitle(conn, "1804655379", "R-33", "Umsjonarmadur");
246              try {
247                  assertEquals("Row should be changed", 1,
248                      pServDesk.getStaffJobTitles(conn).size());
249              } catch (Exception e) {
250                  e.printStackTrace();
251              }
252
253          public void testDeleteStaffJobTitle() throws ItemNotFoundException {
254              assertEquals("DB should initially be empty.", 0,
255                  pServDesk.getStaffJobTitles(conn).size());
256              pServDesk.addStaffJobTitle(conn, "1804655379", "R-33", "Radgjafi");
257              pServDesk.deleteStaffJobTitle(conn, "1804655379");
258              assertEquals("One row should have been deleted.", 0,
259                  pServDesk.getStaffJobTitles(conn).size());
260          }
261      }

```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
project\Jfactor\src\is\unak\webapp\service\AccessoriesInterface.java
```

```

1  package is.unak.webapp.service;
2
3  import is.unak.webapp.activerecords.ItemNotFoundException;
4  import is.unak.webapp.activerecords.accessories.PostnrOps;
5  import is.unak.webapp.activerecords.accessories.StaffPhones;
6  import is.unak.webapp.activerecords.accessories.ClientPhones;
7
8  import java.sql.Connection;
9  import java.util.Collection;
10
11 /**
12  * @author Created by Yngvi Rafn Yngvason
13  * IDE used: IntelliJ IDEA
14  * Date: 21.1.2004
15  * Time: 09:24:03
16  */
17 public interface AccessoriesInterface {
18
19  /**Methods that work on the staff phone properties in the staff_phone table. */
20  public void addStaffPhone(Connection conn, StaffPhones sp);
21  public void changeStaffPhoneType(Connection conn, StaffPhones sp);
22  public void changeStaffPhoneNumber(Connection conn, StaffPhones sp);
23  public Collection getStaffPhones(Connection conn);
24  public String getStaffPhoneType(Connection conn, String phoneType) throws
ItemNotFoundException;
25  public String getStaffPhoneNumber(Connection conn, String kennitala) throws
ItemNotFoundException;
26  public void deleteStaffPhone(Connection conn, String kennitala);
27
28  /**Methods that work on the staff phone properties in the client_phone table. */
29  public void addClientPhone(Connection conn, ClientPhones cp);
30  public void changeClientPhoneType(Connection conn, ClientPhones cp);
31  public void changeClientPhoneNumber(Connection conn, ClientPhones cp);
32  public Collection getClientPhones(Connection conn);
33  public String getClientPhoneType(Connection conn, String phoneType) throws
ItemNotFoundException;
34  public String getClientPhoneNumber(Connection conn, String kennitala) throws
ItemNotFoundException;
35  public void deleteClientPhone(Connection conn, String kennitala);
36
37  /**Methods that work on the postnumber and city properties in the postnr table. */
38  public void addPnrAndCity(Connection conn, PostnrOps pops);
39  public void changeCity(Connection conn, int pnr, String city);
40  public void changePnr(Connection conn, int pnr, String city);
41  public Collection getPnrAndCity(Connection conn);
42  public String getCityName(Connection conn, int pnr);
43  public int getPostnumber(Connection conn, String city);
44
45  /**Methods that work on the properties in the phone_type_picklist table. */
46  public void addToPhonePicklist(Connection conn, String phoneTypePicklist);
47  public Collection getPhonePicklist(Connection conn);
48  public String getPicklistItem(Connection conn, String pickListItem);
49  public void changePhonePicklist(Connection conn, String phoneTypePicklist);
50  public void deletePhonePicklist(Connection conn, String phoneTypePicklist);
51 }
```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
    project\Jfactor\src\is\unak\webapp\service\AccessoriesServiceDesk.java
```

```

1  package is.unak.webapp.service;
2
3  import is.unak.webapp.activerecords.accessories.*;
4  import is.unak.webapp.activerecords.ItemNotFoundException;
5  import java.util.Collection;
6  import java.sql.Connection;
7
8  /**
9   * @author Yngvi Rafn Yngvason
10  * Spring semester 2004.
11  */
12 public class AccessoriesServiceDesk implements AccessoriesInterface {
13
14 /**
15  * Methods for manipulating the staff_phones table.
16  * @param conn
17  * @param sp
18 */
19 public void addStaffPhone(Connection conn, StaffPhones sp) {
20     sp.saveInsert(conn);
21 }
22 /**
23 * @param conn Connection
24 * @param sp StaffPhones
25 */
26 public void changeStaffPhoneType(Connection conn, StaffPhones sp) {
27     sp.saveChange(conn);
28 }
29 /**
30 * @param conn Connection
31 * @param sp StaffPhones
32 */
33 public void changeStaffPhoneNumber(Connection conn, StaffPhones sp) {
34     sp.saveChange(conn);
35 }
36 /**
37 * @param conn Connection
38 * @param kennitala String
39 */
40 public void deleteStaffPhone(Connection conn, String kennitala) {
41     StaffPhones phone = new StaffPhones();
42     phone.setKennitala(kennitala);
43     phone.saveDelete(conn);
44 }
45 /**
46 * @param conn Connection
47 * @return Collection
48 */
49 public Collection getStaffPhones(Connection conn) {
50     Collection phone = StaffPhones.getPhones(conn);
51     return phone;
52 }
53
54 /**
55  * Methods for manipulating the client_phones table.
56  * @param conn Connection
57  * @param cp ClientPhones
58 */
59 public void addClientPhone(Connection conn, ClientPhones cp) {
60     try {
61         cp.saveInsert(conn);
62     } catch (ItemNotFoundException e) {
63         e.printStackTrace();
64     }
65 }
66 /**
67 * @param conn Connection
68 * @param cp ClientPhones
69 */
70 public void changeClientPhoneType(Connection conn, ClientPhones cp) {
```

```

71     try {
72         cp.saveChange(conn);
73     } catch (ItemNotFoundException e) {
74         e.printStackTrace();
75     }
76 }
77 /**
78 * @param conn Connection
79 * @param cp ClientPhones
80 */
81 public void changeClientPhoneNumber(Connection conn, ClientPhones cp) {
82     try {
83         cp.saveChange(conn);
84     } catch (ItemNotFoundException e) {
85         e.printStackTrace();
86     }
87 }
88 /**
89 * @param conn Connection
90 * @param kennitala String
91 */
92 public void deleteClientPhone(Connection conn, String kennitala) {
93     ClientPhones phone = new ClientPhones();
94     phone.setKennitala(kennitala);
95     try {
96         phone.saveDelete(conn);
97     } catch (ItemNotFoundException e) {
98         e.printStackTrace();
99     }
100 }
101 /**
102 * @param conn Connection
103 * @return Collection
104 */
105 public Collection getClientPhones(Connection conn) {
106     Collection phone = ClientPhones.getPhones(conn);
107     return phone;
108 }
109 /**
110 * @param conn Connection
111 * @return Collection
112 */
113 public Collection getPhonePicklist(Connection conn) {
114     Collection picklist = null;
115     try {
116         picklist = PhoneTypePicklist.getPicklist(conn);
117     } catch (ItemNotFoundException e) {
118         e.printStackTrace();
119     }
120     return picklist;
121 }
122 /**
123 * @param conn Connection
124 * @param pickListItem String
125 * @return String
126 */
127 public String getPicklistItem(Connection conn, String pickListItem) {
128     PhoneTypePicklist pickl = PhoneTypePicklist.find(conn, pickListItem);
129     return pickl.getItem();
130 }
131 /**
132 * @param conn Connection
133 * @param phoneTypePicklist String
134 */
135 public void addToPhonePicklist(Connection conn, String phoneTypePicklist) {
136     PhoneTypePicklist pick = new PhoneTypePicklist();
137     pick.setPicklist(phoneTypePicklist);
138     try {
139         pick.saveInsert(conn);
140     } catch (ItemNotFoundException e) {
141         e.printStackTrace();
142     }
143 }
144 /**
145 * @param conn Connection
146 * @param phoneTypePicklist String
147 */

```

```

148 public void changePhonePicklist(Connection conn, String phoneTypePicklist) {
149     PhoneTypePicklist pick = new PhoneTypePicklist();
150     pick.setPicklist(phoneTypePicklist);
151     try {
152         pick.changeInsert(conn);
153     } catch (ItemNotFoundException e) {
154         e.printStackTrace();
155     }
156 }
157 /**
158 * @param conn Connection
159 * @param phoneTypePicklist String
160 */
161 public void deletePhonePicklist(Connection conn, String phoneTypePicklist) {
162     PhoneTypePicklist pick = new PhoneTypePicklist();
163     pick.setPicklist(phoneTypePicklist);
164     try {
165         pick.saveDelete(conn);
166     } catch (ItemNotFoundException e) {
167         e.printStackTrace();
168     }
169 }
170 /**
171 * Next methods handle the staff_phone_type table.
172 * @param conn Connection
173 * @param kennitala String
174 * @return String
175 * @throws ItemNotFoundException
176 */
177 public String getStaffPhoneType(Connection conn, String kennitala) throws
ItemNotFoundException {
178     StaffPhones phones = null;
179     try {
180         phones = StaffPhones.findStaffPhoneInfo(conn, kennitala);
181     } catch (Exception e) {
182         e.printStackTrace();
183     } finally {
184     }
185     return phones.getPhoneType();
186 }
187 /**
188 * @param conn Connction
189 * @param kennitala String
190 * @return String
191 * @throws ItemNotFoundException
192 */
193 public String getStaffPhoneNumber(Connection conn, String kennitala) throws
ItemNotFoundException {
194     StaffPhones phones = null;
195     try {
196         phones = StaffPhones.findStaffPhoneInfo(conn, kennitala);
197     } catch (Exception e) {
198         e.printStackTrace();
199     } finally {
200     }
201     return phones.getNumber();
202 }
203 /**
204 * @param conn Connection
205 * @param kennitala String
206 * @return String
207 * @throws ItemNotFoundException
208 */
209 public String getClientPhoneType(Connection conn, String kennitala) throws
ItemNotFoundException {
210     StaffPhones phones = null;
211     try {
212         phones = StaffPhones.findStaffPhoneInfo(conn, kennitala);
213     } catch (Exception e) {
214         e.printStackTrace();
215     } finally {
216     }
217     return phones.getPhoneType();
218 }
219 /**
220 * @param conn Connection
221 * @param kennitala String

```

```

222      * @return String
223      * @throws ItemNotFoundException
224      */
225      public String getClientPhoneNumber(Connection conn, String kennitala) throws
ItemNotFoundException {
226          StaffPhones phones = null;
227          try {
228              phones = StaffPhones.findStaffPhoneInfo(conn, kennitala);
229          } catch (Exception e) {
230              e.printStackTrace();
231          } finally {
232          }
233          return phones.getNumber();
234      }
235
236 /**
237 * Methods for manipulating the postnr table.
238 * @param conn
239 * @param pops
240 */
241     public void addPnrAndCity(Connection conn, PostnrOps pops) {
242         pops.saveInsert(conn);
243     }
244 /**
245 * @param conn Connection
246 * @param pnr Integer
247 * @param city String
248 */
249     public void changeCity(Connection conn, int pnr, String city) {
250         PostnrOps pops = new PostnrOps();
251         pops.setPnr(pnr);
252         pops.setCity(city);
253         pops.saveCityChange(conn);
254     }
255 /**
256 * @param conn Connection
257 * @param pnr Integer
258 * @param city String
259 */
260     public void changePnr(Connection conn, int pnr, String city) {
261         PostnrOps pops = new PostnrOps();
262         pops.setPnr(pnr);
263         pops.setCity(city);
264         pops.savePnrChange(conn);
265     }
266 /**
267 * @param conn Connection
268 * @return Collection
269 */
270     public Collection getPnrAndCity(Connection conn) {
271         Collection pc = PostnrOps.getPnrs(conn);
272         return pc;
273     }
274 /**
275 * @param conn Connection
276 * @param pnr Integer
277 * @return String
278 */
279     public String getCityName(Connection conn, int pnr) {
280         PostnrOps pc = PostnrOps.findCity(conn, pnr);
281         return pc.getCity();
282     }
283 /**
284 * @param conn Connnection
285 * @param city String
286 * @return integer
287 */
288     public int getPostnumber(Connection conn, String city) {
289         PostnrOps pc = PostnrOps.findPnr(conn, city);
290         return pc.getPnr();
291     }
292 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
project\Jfactor\src\is\unak\webapp\service\ClientInterface.java

```
1 package is.unak.webapp.service;
2
3 import is.unak.webapp.activerecords.client.Client;
4 import java.util.Collection;
5 import java.sql.Connection;
6
7 /**
8 * Created by Yngvi Rafn Yngvason
9 * IDE used: IntelliJ IDEA
10 * Date: 23.1.2004
11 * Time: 09:44:36
12 */
13 public interface ClientInterface {
14
15     void addClient(Connection conn, Client c);
16
17     Collection getClients(Connection conn);
18
19     void changeClient(Connection conn, Client c);
20
21     String getClientName(Connection conn, String kt);
22
23 }
24 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\service\ClientServiceDesk.java

```

1  package is.unak.webapp.service;
2
3  import is.unak.webapp.activerecords.client.Client;
4
5  import java.util.Collection;
6  import java.sql.Connection;
7
8  /**
9   * @author Created by Yngvi Rafn Yngvason
10  * IDE used: IntelliJ IDEA
11  * Date: 23.1.2004
12  * Time: 09:45:36
13  */
14 public class ClientServiceDesk implements ClientInterface {
15
16     public void addClient(Connection conn, Client c) {
17         /* This code will be moved to the JSP page which creates the Client object.
18
19         Client client = new Client();
20         client.setKennitala("180465-5379");
21         client.setClientId(1);
22         client.setFirstName("Yngvi");
23         client.setLastName("Yngvason");
24         client.setPnr(600);
25         client.setAddress("Munkapverárstræti 1");
26         client.setPhoneId("gsm");
27         client.setCaseId(54637);
28         client.setHasAjob("t");
29         client.setHajWhere("Self employed");
30         client.setcJobId(1);
31         client.setcJobCompany("Jfactor Inc");
32         client.setcJobPosition("Manager");
33         client.setcJobStarted();
34         client.setcJobQuit();
35         client.setMarried("t");
36         client.setSpouseKt("260772-5979");
37         client.setMarriedTo("Alis Freygardsdottir");
38         client.setDivorced("f");
39         client.setDivorcedHowOften(0);
40         client.setcLawMarriage("f");
41         client.setSingle("f");
42         client.setNoOfChlidren(2);
43         client.setRelName("");
44         */
45         c.saveInsert(conn);
46     }
47
48     public Collection getClients(Connection conn) {
49         return Client.getClients(conn);
50     }
51
52     public void changeClient(Connection conn, Client c) {
53
54         c.saveChange(conn);
55     }
56
57     public String getClientName(Connection conn, String kt) {
58         Client client = Client.find(conn, kt);
59         return client.getFirstName();
60     }
61
62 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\service\JusticeInterface.java

```
1 package is.unak.webapp.service;
2
3 import java.util.Collection;
4 import java.sql.Connection;
5
6 /**
7 * Created by Yngvi Rafn Yngvason
8 * IDE used: IntelliJ IDEA
9 * Date: 1.2.2004
10 * Time: 09:22:51
11 */
12 public interface JusticeInterface {
13
14     //These methods affect the jails table.
15     public Collection getJails(Connection conn);
16     public String getJail(Connection conn, String jailName);
17     public void addJail(Connection conn, JusticeServiceDesk jsd);
18     public void changeJail(Connection conn, JusticeServiceDesk jsd);
19     public void deleteJail(Connection conn, String jailName);
20 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\service\JusticeServiceDesk.java

```
1 package is.unak.webapp.service;
2
3 import is.unak.webapp.activerecords.justicefacilities.Jails;
4
5 import java.util.Collection;
6 import java.sql.Connection;
7
8 /**
9  * Created by Yngvi Rafn Yngvason
10 * IDE used: IntelliJ IDEA
11 * Date: 1.2.2004
12 * Time: 11:06:42
13 */
14 public class JusticeServiceDesk implements JusticeInterface {
15
16     public Collection getJails(Connection conn) {
17         Collection jails = Jails.getJails(conn);
18         return jails;
19     }
20
21     /**
22      * @param conn Connection
23      * @param jailName String
24      * @return String
25      */
26     public String getJail(Connection conn, String jailName) {
27         Jails jails = Jails.find(conn, jailName);
28         return jails.getJailName();
29     }
30
31     public void addJail(Connection conn, JusticeServiceDesk jsd) {
32     }
33
34     public void changeJail(Connection conn, JusticeServiceDesk jsd) {
35     }
36
37     public void deleteJail(Connection conn, String jailName) {
38
39     }
40 }
41 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\service\PersonnelInterface.java

```
1 package is.unak.webapp.service;
2
3 import is.unak.webapp.activerecords.staff.Staff;
4 import is.unak.webapp.activerecords.staff.StaffJobs;
5 import is.unak.webapp.activerecords.staff.StaffJobTitles;
6
7 import java.sql.Connection;
8 import java.util.Collection;
9
10 /**
11 * @author Created by Yngvi Rafn Yngvason
12 * IDE used: IntelliJ IDEA
13 * Date: 18.1.2004
14 * Time: 14:33:28
15 */
16 public interface PersonnelInterface {
17
18     /**Staff table interface.*/
19     public String getStaffName(Connection conn, String kennitala);
20     public Collection getStaffMembers(Connection conn);
21     public void addStaff(Connection conn, Staff staff);
22     public void changeStaffMember(Connection conn, Staff staff);
23     public void deleteStaffMember(Connection conn, String kennitala);
24
25     /**Staff jobs table interface.*/
26     public Collection getStaffJobs(Connection conn);
27     public StaffJobs getStaffJob(Connection conn, String sJobId);
28     public void addStaffJob(Connection conn, StaffJobs sJob);
29     public void changeStaffJob(Connection conn, StaffJobs sJob);
30     public void deleteStaffJob(Connection conn, String kennitala);
31
32     /**S job titles table interface.*/
33     public Collection getStaffJobTitles(Connection conn);
34     public String getStaffJobTitle(Connection conn, String kennitala);
35     public void addStaffJobTitle(Connection conn, StaffJobTitles sjt);
36     public void changeStaffJobTitle(Connection conn, StaffJobTitles sjt);
37     public void deleteStaffJobTitle(Connection conn, String kennitala);
38 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\service\PersonnelServiceDesk.java

```

1  package is.unak.webapp.service;
2
3  import is.unak.webapp.activerecords.staff.Staff;
4  import is.unak.webapp.activerecords.staff.StaffJobs;
5  import is.unak.webapp.activerecords.staff.StaffJobTitles;
6
7  import java.util.Collection;
8  import java.sql.Connection;
9
10 public class PersonnelServiceDesk implements PersonnelInterface {
11
12     /**
13      *Following five methods affect the staff table.
14      * @param conn Connection
15      * @return Collection
16      */
17     public Collection getStaffMembers(Connection conn) {
18         Collection staff = Staff.getStaffMembers(conn);
19         return staff;
20     }
21     /**
22      * @param conn Connection
23      * @param kennitala String
24      * @return String
25      */
26     public String getStaffName(Connection conn, String kennitala) {
27         Staff staff = Staff.find(conn, kennitala);
28         return staff.getFirstName();
29     }
30     /**
31      * @param conn Connection
32      * @param staff Staff object
33      */
34     public void addStaff(Connection conn, Staff staff){
35         staff.saveInsert(conn);
36     }
37     /**
38      * @param conn Connection
39      * @param staff Staff object
40      */
41     public void changeStaffMember(Connection conn, Staff staff) {
42         staff.saveChange(conn);
43     }
44     /**
45      * @param conn Connection
46      * @param kennitala String
47      */
48     public void deleteStaffMember(Connection conn, String kennitala) {
49         Staff staff = new Staff();
50         staff.setKennitala(kennitala);
51         staff.saveDelete(conn, kennitala);
52     }
53
54     /**
55      * Following five methods affect the staff_jobs table.
56      * @param conn Connection
57      * @return Collection
58      */
59     public Collection getStaffJobs(Connection conn) {
60         Collection sJobs = StaffJobs.getStaffJobs(conn);
61         return sJobs;
62     }
63     /**
64      * @param conn Connection
65      * @param sJobId String
66      * @return StaffJobs object
67      */
68     public StaffJobs getStaffJob(Connection conn, String sJobId) {
69         StaffJobs sJob = StaffJobs.find(conn, sJobId);
70         return sJob;

```

```

71      }
72      /**
73       * @param conn Connection
74       * @param sJob StaffJobs object
75       */
76      public void addStaffJob(Connection conn, StaffJobs sJob) {
77          sJob.saveInsert(conn);
78      }
79      /**
80       * @param conn Connection
81       * @param sJob StaffJobs object
82       */
83      public void changeStaffJob(Connection conn, StaffJobs sJob) {
84          sJob.saveChange(conn);
85      }
86      /**
87       * @param conn Connection
88       * @param sJob String
89       */
90      public void deleteStaffJob(Connection conn, String sJob) {
91          StaffJobs sjob = new StaffJobs();
92          sjob.setSjobId(sJob);
93          sjob.saveDelete(conn, sJob);
94      }
95
96      /**
97       * Following five methods affect the s_job_titles table.
98       * @param conn Connection
99       * @return Collection
100      */
101     public Collection getStaffJobTitles(Connection conn) {
102         Collection sjt = StaffJobTitles.getStaffJobTitles(conn);
103         return sjt;
104     }
105
106     /**
107      * @param conn Connection
108      * @param kennitala String
109      * @return String
110      */
111     public String getStaffJobTitle(Connection conn, String kennitala) {
112         StaffJobTitles sJobTitle = StaffJobTitles.find(conn, kennitala);
113         return sJobTitle.getTitle();
114     }
115
116     /**
117      * @param conn Connection
118      * @param sjt StaffJobTitles object
119      */
120     public void addStaffJobTitle(Connection conn, StaffJobTitles sjt) {
121         sjt.saveInsert(conn);
122     }
123
124     /**
125      * @param conn Connection
126      * @param sjt StaffJobTitles object
127      */
128     public void changeStaffJobTitle(Connection conn, StaffJobTitles sjt) {
129         sjt.saveChange(conn);
130     }
131
132     /**
133      * @param conn Connection
134      * @param kennitala String
135      */
136     public void deleteStaffJobTitle(Connection conn, String kennitala) {
137         StaffJobTitles sjt = new StaffJobTitles();
138         sjt.setKennitala(kennitala);
139         sjt.saveDelete(conn);
140     }
141 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\accessories\AccessoriesSQLStatements.java

```

1  package is.unak.webapp.activerecords.accessories;
2
3  /**
4   * Created by Yngvi Rafn Yngvason
5   * IDE used: IntelliJ IDEA
6   * Date: 24.1.2004
7   * Time: 17:53:51
8   */
9  public class AccessoriesSQLStatements {
10
11     /**
12      * SQL strings for the staff_phone table.
13      */
14     final static String insertStaffPhone = "INSERT INTO staff_phones VALUES (?, ?, ?)";
15     final static String updateStaffPhone = "UPDATE staff_phones SET kennitala = ?,  
phone_type = ?, number = ?";
16     final static String findStaffPhones = "SELECT * FROM staff_phones";
17     final static String deleteStaffPhone = "DELETE FROM staff_phones WHERE kennitala = ?";
18     final static String findStaffPhone = "SELECT * FROM staff_phones WHERE kennitala = ?";
19
20     /**
21      * SQL strings for the client_phone table.
22      */
23     final static String insertClientPhone = "INSERT INTO client_phones VALUES (?, ?, ?)";
24     final static String updateClientPhone = "UPDATE client_phones SET kennitala = ?,  
phone_type = ?, number = ?";
25     final static String findClientPhones = "SELECT * FROM client_phones";
26     final static String deleteClientPhone = "DELETE FROM client_phones WHERE kennitala = ?";
27     final static String findClientPhone = "SELECT * FROM client_phones WHERE kennitala = ?";
28
29     /**
30      * Statements affecting postnumbers and cities.
31      */
32     final static String insertPnr = "INSERT INTO postnr VALUES(?, ?)";
33     final static String updateCity = "UPDATE postnr SET city = ? WHERE pnr = ?";
34     final static String updatePnr = "UPDATE postnr SET pnr = ? WHERE city = ?";
35     final static String findCity = "SELECT pnr, city FROM postnr WHERE pnr = ?";
36     final static String findPnr = "SELECT pnr, city FROM postnr WHERE city = ?";
37     final static String findPnrs = "SELECT pnr, city FROM postnr";
38     final static String deletePostnrRow = "DELETE FROM postnr WHERE pnr = ? AND city = ?";
39
40     /**
41      * Statements affecting the PhoneTypePicklist class.
42      */
43     final static String insertPicklistItem = "INSERT INTO phone_type_picklist VALUES  
(?)";
44     final static String updatePicklistItem = "UPDATE phone_type_picklist SET ph_type = ?";
45     final static String deletePicklistItem = "DELETE FROM phone_type_picklist WHERE  
ph_type = ?";
46     final static String findPicklist = "SELECT * FROM phone_type_picklist";
47     final static String findItem = "SELECT ph_type FROM phone_type_picklist WHERE ph_type  
= ?";
48 }

```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
project\Jfactor\src\is\unak\webapp\activerecords\accessories\ClientPhones.java
```

```

1  package is.unak.webapp.activerecords.accessories;
2
3
4  import is.unak.webapp.activerecords.ItemNotFoundException;
5
6  import java.util.Collection;
7  import java.util.ArrayList;
8  import java.sql.Connection;
9  import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12
13 /**
14  * Created by Yngvi Rafn Yngvason
15  * IDE used: IntelliJ IDEA
16  * Date: 26.1.2004
17  * Time: 17:13:21
18 */
19 public class ClientPhones extends PhoneInfo{
20
21     /**
22      * Returns a collection of phones from the phones table.
23      * @param conn Connection
24      * @return Collection
25     */
26     public static Collection getPhones(Connection conn) {
27         Collection phones = new ArrayList();
28         try {
29             //Inserts the data into the SQL string
30             PreparedStatement selectStmt;
31             ClientPhones cph;
32             selectStmt =
conn.prepareStatement(AccessoriesSQLStatements.findClientPhones);
33             ResultSet result = selectStmt.executeQuery();
34
35             while (result.next()) {
36                 cph = new ClientPhones();
37                 cph.setKennitala(result.getString("kennitala"));
38                 cph.setPhoneType(result.getString("phone_type"));
39                 cph.setNumber(result.getString("number"));
40                 phones.add(cph);
41             } //throw new ItemNotFoundException("There are no such numbers in the
database");
42         } catch (SQLException e) {
43             e.printStackTrace();
44         } // catch (ItemNotFoundException e) {
45         // e.printStackTrace();
46     } finally {
47     }
48     return phones;
49 }
50 /**
51  * @param conn Connection
52  * @param kennitala String
53  * @return ClientPhones object
54 */
55 public static ClientPhones findClientPhoneInfo(Connection conn, String kennitala) {
56     PreparedStatement selectStmt = null;
57     ClientPhones cph = null;
58     try {
59         selectStmt = conn.prepareStatement(AccessoriesSQLStatements.findClientPhone);
60         selectStmt.setString(1, kennitala);
61         ResultSet result = selectStmt.executeQuery();
62         if (result.next()) {
63             cph = new ClientPhones();
64             cph.setKennitala(result.getString("kennitala"));
65             cph.setPhoneType(result.getString("phone_type"));
66             cph.setNumber(result.getString("number"));
67             return cph;
}

```

```

68             } //throw new ItemNotFoundException("Sorry, but there is no phone
information for this individual");
69         } catch (SQLException e) {
70             e.printStackTrace();
71         } finally {
72             return cph;
73         }
74     }
75 /**
76 * @param conn Connection
77 * @throws ItemNotFoundException
78 */
79 public void saveInsert(Connection conn) throws ItemNotFoundException {
80     try {
81         PreparedStatement insertStmt;
82         insertStmt =
conn.prepareStatement(AccessoriesSQLStatements.insertClientPhone);
83         insertStmt.setString(1, getKennitala());
84         insertStmt.setString(2, getPhoneType());
85         insertStmt.setString(3, getNumber());
86         insertStmt.executeUpdate();
87         // conn.commit();
88         throw new ItemNotFoundException("This client is already registered.");
89     } catch (SQLException e) {
90         e.printStackTrace();
91     }
92 }
93 /**
94 * @param conn Connection
95 * @throws ItemNotFoundException
96 */
97 public void saveChange(Connection conn) throws ItemNotFoundException {
98     try {
99         PreparedStatement updateStmt;
100        updateStmt =
conn.prepareStatement(AccessoriesSQLStatements.updateClientPhone);
101        updateStmt.setString(1, getKennitala());
102        updateStmt.setString(2, getPhoneType());
103        updateStmt.setString(3, getNumber());
104        updateStmt.executeUpdate();
105        // conn.commit();
106        throw new ItemNotFoundException("This client is not in the database.");
107    } catch (SQLException e) {
108        e.printStackTrace();
109    }
110 }
111 /**
112 * @param conn Connection
113 * @throws ItemNotFoundException
114 */
115 public void saveDelete(Connection conn) throws ItemNotFoundException {
116     try {
117         PreparedStatement deleteStmt;
118         deleteStmt =
conn.prepareStatement(AccessoriesSQLStatements.deleteClientPhone);
119         deleteStmt.setString(1, getKennitala());
120         deleteStmt.executeUpdate();
121         // conn.commit();
122         throw new ItemNotFoundException("This number is not in the database.");
123     } catch (SQLException e) {
124         e.printStackTrace();
125     }
126 }
127 }

```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\accessories\MailInfo.java

```
1  package is.unak.webapp.activerecords.accessories;
2
3  /**
4   * Created by Yngvi Rafn Yngvason
5   * IDE used: IntelliJ IDEA
6   * Date: 20.1.2004
7   * Time: 23:43:21
8   */
9
10 /**
11  * This class describes the characteristics of the postnumber system
12 */
13 public class MailInfo {
14
15     private int pnr;
16     private String city;
17
18     public int getPnr() {
19         return pnr;
20     }
21
22     public void setPnr(int pnr) {
23         this.pnr = pnr;
24     }
25
26     public String getCity() {
27         return city;
28     }
29
30     public void setCity(String city) {
31         this.city = city;
32     }
33 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\accessories\PhoneInfo.java

```
1 package is.unak.webapp.activerecords.accessories;
2
3 /**
4  * Created by Yngvi Rafn Yngvason
5  * IDE used: IntelliJ IDEA
6  * Date: 18.1.2004
7  * Time: 14:57:53
8 */
9 public class PhoneInfo {
10
11     private String kennitala, phoneType, number;
12
13     public String getKennitala() {
14         return kennitala;
15     }
16
17     public void setKennitala(String kennitala) {
18         this.kennitala = kennitala;
19     }
20
21     public String getPhoneType() {
22         return phoneType;
23     }
24
25     public void setPhoneType(String phoneType) {
26         this.phoneType = phoneType;
27     }
28
29     public String getNumber() {
30         return number;
31     }
32
33     public void setNumber(String number) {
34         this.number = number;
35     }
36 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\accessories\PhoneTypePicklist.java

```

1  package is.unak.webapp.activerecords.accessories;
2
3  import is.unak.webapp.activerecords.ItemNotFoundException;
4  import java.util.Collection;
5  import java.util.ArrayList;
6  import java.sql.Connection;
7  import java.sql.PreparedStatement;
8  import java.sql.ResultSet;
9  import java.sql.SQLException;
10
11 /**
12  * Created by Yngvi Rafn Yngvason
13  * IDE used: IntelliJ IDEA
14  * Date: 26.1.2004
15  * Time: 17:55:38
16  */
17 public class PhoneTypePicklist {
18
19     String picklist, item;
20
21     /**
22      * Returns a collection of phone types from the phone_type_picklist table.
23      * @param conn Connection
24      * @return Collection
25      * @throws ItemNotFoundException
26      */
27     public static Collection getPicklist(Connection conn) throws ItemNotFoundException {
28         Collection phones = new ArrayList();
29         try {
30             //Inserts the data into the SQL string
31             PreparedStatement selectStmt;
32             PhoneTypePicklist picklist;
33             selectStmt = conn.prepareStatement(AccessoriesSQLStatements.findPicklist);
34             ResultSet result = selectStmt.executeQuery();
35             while (result.next()) {
36                 picklist = new PhoneTypePicklist();
37                 picklist.setPicklist(result.getString("ph_type"));
38                 phones.add(picklist);
39             } throw new ItemNotFoundException("The picklist is empty.");
40         } catch (SQLException e) {
41             e.printStackTrace();
42         } finally {
43         }
44         return phones;
45     }
46     /**
47      * @param conn Connection
48      * @param picklistItem String
49      * @return PhoneTypePicklist object
50      */
51     public static PhoneTypePicklist find(Connection conn, String picklistItem) {
52         PreparedStatement selectStmt = null;
53         PhoneTypePicklist picklist = null;
54         try {
55             selectStmt = conn.prepareStatement(AccessoriesSQLStatements.findItem);
56             selectStmt.setString(1, picklistItem);
57             ResultSet result = selectStmt.executeQuery();
58             if (result.next()) {
59                 picklist = newPicklistItem(result);
60             }
61         } catch (SQLException e) {
62             e.printStackTrace();
63         } finally {
64             return picklist;
65         }
66     }
67     /**
68      * @param resultSet ResultSet
69      * @return PhoneTypePicklist object
70      * @throws SQLException

```

```

71      */
72      private static PhoneTypePicklist newPicklistItem(ResultSet result) throws
73          SQLException {
74          PhoneTypePicklist picklist = new PhoneTypePicklist();
75          picklist.setPicklist(result.getString("ph_type"));
76          return picklist;
77      }
78      /**
79       * @param conn Connection
80       * @throws ItemNotFoundException
81       */
82      public void saveInsert(Connection conn) throws ItemNotFoundException {
83          try {
84              PreparedStatement insertStmt;
85              insertStmt =
86                  conn.prepareStatement(AccessoriesSQLStatements.insertPicklistItem);
87              insertStmt.setString(1, getPicklist());
88              insertStmt.executeUpdate();
89              // conn.commit();
90              throw new ItemNotFoundException("This client is not in the database.");
91          } catch (SQLException e) {
92              e.printStackTrace();
93          }
94      }
95      /**
96       * @param conn Connection
97       * @throws ItemNotFoundException
98       */
99      public void changeInsert(Connection conn) throws ItemNotFoundException {
100         try {
101             PreparedStatement insertStmt;
102             insertStmt =
103                 conn.prepareStatement(AccessoriesSQLStatements.updatePicklistItem);
104             insertStmt.setString(1, getPicklist());
105             insertStmt.executeUpdate();
106             // conn.commit();
107             throw new ItemNotFoundException("This client is not in the database.");
108         } catch (SQLException e) {
109             e.printStackTrace();
110         }
111     }
112     /**
113      public void saveDelete(Connection conn) throws ItemNotFoundException {
114         try {
115             PreparedStatement insertStmt;
116             insertStmt =
117                 conn.prepareStatement(AccessoriesSQLStatements.deletePicklistItem);
118             insertStmt.setString(1, getPicklist());
119             insertStmt.executeUpdate();
120             // conn.commit();
121             throw new ItemNotFoundException("This item is not in the database.");
122         } catch (SQLException e) {
123             e.printStackTrace();
124         }
125     }
126     //Since these are the only getters and setters for this table,
127     //I will keep them here but not in a seperate class.
128     public String getPicklist() {
129         return picklist;
130     }
131     public void setPicklist(String picklist) {
132         this.picklist = picklist;
133     }
134     public String getItem() {
135         return item;
136     }
137     public void setItem(String item) {
138         this.item = item;
139     }
140 }
141 
```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
project\Jfactor\src\is\unak\webapp\activerecords\accessories\PostnrOps.java
```

```

1  package is.unak.webapp.activerecords.accessories;
2
3  import is.unak.webapp.activerecords.ItemNotFoundException;
4
5  import java.sql.PreparedStatement;
6  import java.sql.Connection;
7  import java.sql.SQLException;
8  import java.sql.ResultSet;
9  import java.util.Collection;
10 import java.util.ArrayList;
11
12 /**
13 * Created by Yngvi Rafn Yngvason
14 * IDE used: IntelliJ IDEA
15 * Date: 21.1.2004
16 * Time: 09:14:21
17 */
18 public class PostnrOps extends MailInfo {
19     /**
20      * @param conn Connection
21      * @return Collection
22     */
23     public static Collection getPnrs(Connection conn) {
24         Collection pc = new ArrayList();
25         try {
26             PreparedStatement selectStmt;
27             PostnrOps pops;
28             selectStmt = conn.prepareStatement(AccessoriesSQLStatements.findPnrs);
29             ResultSet result = selectStmt.executeQuery();
30             while (result.next()) {
31                 pops = newMailInfo(result);
32                 pc.add(pops);
33             } throw new ItemNotFoundException("Postnumber table is empty so there is
nothing to return.");
34         } catch (SQLException e) {
35             e.printStackTrace();
36         } finally {
37             return pc;
38         }
39     }
40     /**
41      * @param conn Connection
42      * @param city String
43      * @return PostnrOps object
44     */
45     public static PostnrOps findPnr(Connection conn, String city) {
46         PreparedStatement selectStmt = null;
47         PostnrOps pops = null;
48         try {
49             selectStmt = conn.prepareStatement(AccessoriesSQLStatements.findPnr);
50             selectStmt.setString(1, city);
51             ResultSet result = selectStmt.executeQuery();
52             if (result.next()) {
53                 pops = newMailInfo(result);
54             } throw new ItemNotFoundException("This postnumber is not in the database.");
55         } catch (SQLException e) {
56             e.printStackTrace();
57         } finally {
58             return pops;
59         }
60     }
61     /**
62      * @param conn Connection
63      * @param pnr Integer
64      * @return PostnrOps object
65     */
66     public static PostnrOps findCity(Connection conn, int pnr) {
67         PreparedStatement selectStmt = null;
68         PostnrOps pops = null;
69         try {

```

```

70     selectStmt = conn.prepareStatement(AccessoriesSQLStatements.findCity);
71     selectStmt.setInt(1, pnr);
72     ResultSet result = selectStmt.executeQuery();
73     if (result.next()) {
74         pops = newMailInfo(result);
75     } throw new ItemNotFoundException("This city is not in the database.");
76 } catch (SQLException e) {
77     e.printStackTrace();
78 } finally {
79     return pops;
80 }
81 /**
82 * @param result ResultSet
83 * @return PostnrOps object
84 * @throws SQLException
85 */
86 private static PostnrOps newMailInfo(ResultSet result) throws SQLException {
87     PostnrOps pops = new PostnrOps();
88     pops.setPnr(result.getInt("pnr"));
89     pops.setCity(result.getString("city"));
90     return pops;
91 }
92 /**
93 * @param conn Connection
94 */
95 public void saveInsert(Connection conn) {
96     PreparedStatement insertStmt = null;
97     try {
98         insertStmt = conn.prepareStatement(AccessoriesSQLStatements.insertPnr);
99         insertStmt.setInt(1, getPnr());
100        insertStmt.setString(2, getCity());
101        insertStmt.executeUpdate();
102        // conn.commit();
103    } catch (SQLException e) {
104        e.printStackTrace();
105    }
106 }
107 /**
108 * @param conn Connection
109 */
110 public void saveCityChange(Connection conn) {
111     PreparedStatement updateStmt = null;
112     try {
113         updateStmt = conn.prepareStatement(AccessoriesSQLStatements.updateCity);
114         updateStmt.setString(1, getCity());
115         updateStmt.setInt(2, getPnr());
116         updateStmt.executeUpdate();
117         // conn.commit();
118     } catch (SQLException e) {
119         e.printStackTrace();
120     }
121 }
122 /**
123 * @param conn Connection
124 */
125 public void savePnrChange(Connection conn) {
126     PreparedStatement updateStmt = null;
127     try {
128         updateStmt = conn.prepareStatement(AccessoriesSQLStatements.updatePnr);
129         updateStmt.setInt(1, getPnr());
130         updateStmt.setString(2, getCity());
131         updateStmt.executeUpdate();
132         // conn.commit();
133     } catch (SQLException e) {
134         e.printStackTrace();
135     }
136 }
137 }
138 }
```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
project\Jfactor\src\is\unak\webapp\activerecords\accessories\StaffPhones.java
```

```

1  package is.unak.webapp.activerecords.accessories;
2
3  import is.unak.webapp.activerecords.ItemNotFoundException;
4
5  import java.util.Collection;
6  import java.util.ArrayList;
7  import java.sql.*;
8
9  /**
10   * Created by Yngvi Rafn Yngvason
11   * IDE used: IntelliJ IDEA
12   * Date: 13.11.2003
13   * Time: 09:33:26
14   */
15
16 /**
17 * @author Yngvi Rafn Yngvason
18 */
19
20 public class StaffPhones extends PhoneInfo {
21
22     //Returns a collection of phones from the phones table
23     public static Collection getPhones(Connection conn) {
24         Collection phones = new ArrayList();
25         try {
26             //Inserts the data into the SQL string
27             PreparedStatement selectStmt;
28             StaffPhones sph;
29             selectStmt = conn.prepareStatement(AccessoriesSQLStatements.findStaffPhones);
30             ResultSet result = selectStmt.executeQuery();
31             while (result.next()) {
32                 sph = new StaffPhones();
33                 sph.setKennitala(result.getString("kennitala"));
34                 sph.setPhoneType(result.getString("phone_type"));
35                 sph.setNumber(result.getString("number"));
36                 phones.add(sph);
37             }
38         } catch (SQLException e) {
39             e.printStackTrace();
40         } finally {
41         }
42         return phones;
43     }
44 /**
45 * @param conn
46 * @param kennitala
47 * @return StaffPhones object
48 */
49 public static StaffPhones findStaffPhoneInfo(Connection conn, String kennitala) {
50     PreparedStatement selectStmt = null;
51     StaffPhones sph = null;
52     try {
53         selectStmt = conn.prepareStatement(AccessoriesSQLStatements.findStaffPhone);
54         selectStmt.setString(1, kennitala);
55         ResultSet result = selectStmt.executeQuery();
56         if (result.next()) {
57             sph = new StaffPhones();
58             sph.setKennitala(result.getString("kennitala"));
59             sph.setPhoneType(result.getString("phone_type"));
60             sph.setNumber(result.getString("number"));
61             return sph;
62         } throw new ItemNotFoundException("Sorry, but there is no phone information
for this individual");
63     } catch (SQLException e) {
64         e.printStackTrace();
65     } finally {
66         return sph;
67     }
68 }
69 /**

```

```
70      * @param conn Connection
71      */
72  public void saveInsert(Connection conn) {
73      try {
74          PreparedStatement insertStmt;
75          insertStmt =
76          conn.prepareStatement(AccessoriesSQLStatements.insertStaffPhone);
77          insertStmt.setString(1, getKennitala());
78          insertStmt.setString(2, getPhoneType());
79          insertStmt.setString(3, getNumber());
80          insertStmt.executeUpdate();
81          // conn.commit();
82      } catch (SQLException e) {
83          e.printStackTrace();
84      }
85  /**
86   * @param conn Connection
87   */
88  public void saveChange(Connection conn) {
89      try {
90          PreparedStatement updateStmt;
91          updateStmt =
92          conn.prepareStatement(AccessoriesSQLStatements.updateStaffPhone);
93          updateStmt.setString(1, getKennitala());
94          updateStmt.setString(2, getPhoneType());
95          updateStmt.setString(3, getNumber());
96          updateStmt.executeUpdate();
97          // conn.commit();
98      } catch (SQLException e) {
99          e.printStackTrace();
100     }
101 /**
102  * @param conn Connection
103 */
104 public void saveDelete(Connection conn){
105     try {
106         PreparedStatement deleteStmt;
107         deleteStmt =
108         conn.prepareStatement(AccessoriesSQLStatements.deleteStaffPhone);
109         deleteStmt.setString(1, getKennitala());
110         deleteStmt.executeUpdate();
111     } catch (SQLException e) {
112         e.printStackTrace();
113     }
114 }
115 }
```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
    project\Jfactor\src\is\unak\webapp\activerecords\client\Client.java
```

```

1  package is.unak.webapp.activerecords.client;
2
3  import java.util.Collection;
4  import java.util.ArrayList;
5  import java.sql.Connection;
6  import java.sql.SQLException;
7  import java.sql.PreparedStatement;
8  import java.sql.ResultSet;
9
10 /**
11  * Created by Yngvi Rafn Yngvason
12  * IDE used: IntelliJ IDEA
13  * Date: 19.1.2004
14  * Time: 15:58:26
15  */
16 public class Client extends ClientPerson{
17     /**
18      * @param conn Connection
19      * @return Collection
20      */
21     public static Collection getClients(Connection conn) {
22         Collection client = new ArrayList();
23         try {
24             findClients(conn, client);
25         } catch (SQLException e) {
26             e.printStackTrace();
27         } finally {
28             return client;
29         }
30     }
31     /**
32      * @param conn Connection
33      * @param clients Collection
34      * @throws SQLException
35      */
36     private static void findClients(Connection conn, Collection clients) throws
37     SQLException {
38         PreparedStatement selectStmt;
39         Client client;
40         selectStmt = conn.prepareStatement(ClientSQLStatements.findClients);
41         ResultSet result = selectStmt.executeQuery();
42         while (result.next()) {
43             client = newClient(result);
44             clients.add(client);
45         }
46     }
47     /**
48      * @param conn Connection
49      * @param kennitala String
50      * @return Client object
51      */
52     public static Client find(Connection conn, String kennitala) {
53         PreparedStatement selectStmt = null;
54         Client client = null;
55         try {
56             selectStmt = conn.prepareStatement(ClientSQLStatements.findClientName);
57             selectStmt.setString(1, kennitala);
58             ResultSet result = selectStmt.executeQuery();
59             if (result.next()) {
60                 client = newClient(result);
61             }
62         } catch (SQLException e) {
63             e.printStackTrace();
64         } finally {
65             return client;
66         }
67     }
68     /**
69      * @param result ResultSet
70      * @return Client object

```

```

70      * @throws SQLException
71      */
72  private static Client newClient(ResultSet result) throws SQLException {
73      Client client = new Client();
74      client.setKennitala(result.getString("ckennitala"));
75      client.setClientId(result.getInt("client_Id"));
76      client.setFirstName(result.getString("cfirstname"));
77      client.setLastName(result.getString("clastname"));
78      client.setPnr(result.getInt("pnr"));
79      client.setAddress(result.getString("address"));
80      client.setPhoneId(result.getString("phone_id"));
81      client.setCaseId(result.getInt("case_id"));
82      client.setHasAjob(result.getString("has_a_job"));
83      client.setHajWhere(result.getString("haj_where"));
84      client.setcJobId(result.getInt("cjob_id"));
85      client.setcJobCompany(result.getString("cjob_company"));
86      client.setcJobPosition(result.getString("cjob_position"));
87      client.setcJobStarted(result.getDate("cjob_started"));
88      client.setcJobQuit(result.getDate("cjob_quit"));
89      client.setMarried(result.getString("married"));
90      client.setSpouseKt(result.getString("spouse_kt"));
91      client.setMarriedTo(result.getString("married_to"));
92      client.setDivorced(result.getString("divorced"));
93      client.setDivorcedHowOften(result.getInt("div_how_often"));
94      client.setcLawMarriage(result.getString("c_law_marriage"));
95      client.setSingle(result.getString("single"));
96      client.setNoOfChlidren(result.getInt("no_of_children"));
97      client.setRelName(result.getString("rel_name"));
98      client.setOwnApmt(result.getString("own_apmt"));
99      client.setRentApmt(result.getString("rent_apmt"));
100     client.setFriendRelApmt(result.getString("friend_rel_apmt"));
101     client.setFrWhere(result.getString("fr_where"));
102     client.setHomeless(result.getString("homeless"));
103     client.setHlWhere(result.getString("hl_where"));
104     client.setDeptId(result.getString("dept_id"));
105     client.setDeptType(result.getString("dept_type"));
106     client.setLoanId(result.getInt("loan_id"));
107     client.setLoanOwner(result.getString("loan_owner"));
108     client.setDeptAmount(result.getString("dept_amount"));
109     client.setAnalysisResult(result.getString("analysis_result"));
110     client.setFirstSip(result.getDate("first_sip"));
111     client.setFirstDrugAbuse(result.getDate("first_drug_abuse"));
112     client.setFirstTreatment(result.getDate("first_treatment"));
113     client.setReasons(result.getBlob("reasons")); //THE REASONS FIELD NEEDS TO BE
ANALYSED BETTER FOR DATA MINING REASONS
114     client.setFrequency(result.getInt("frequency"));
115     client.setFrequencyDWM(result.getString("frequency_dwm"));
116     client.setQty(result.getInt("qty"));
117     client.setQtyDWM(result.getString("qty_dwm"));
118     client.setComments(result.getBlob("comments"));
119     client.setIpId(result.getInt("ip_id"));
120     client.setDrugId(result.getString("drug_id"));
121     client.setStepDownId(result.getInt("step_down_id"));
122     client.setDischargeDate(result.getDate("discharge_date"));
123     return client;
124 }
125 /**
126  * @param conn Connection
127 */
128 public void saveInsert(Connection conn) {
129     PreparedStatement insertStmt = null;
130     try {
131         insertStmt = conn.prepareStatement(ClientSQLStatements.insertClient);
132         insertStmt.setString(1, getKennitala());
133         insertStmt.setInt(2, getClientId());
134         insertStmt.setInt(3, getJobId());
135         insertStmt.setString(4, getFirstName());
136         insertStmt.setString(5, getLastName());
137         insertStmt.setInt(6, getPnr());
138         insertStmt.setString(7, getAddress());
139         insertStmt.executeUpdate();
140         // conn.commit();
141     } catch (SQLException e) {
142         e.printStackTrace();
143     }
144 }
145 /**

```

```

146     * @param conn Connection
147     */
148     public void saveChange(Connection conn) {
149         PreparedStatement updateStmt = null;
150         try {
151             updateStmt = conn.prepareStatement(ClientSQLStatements.updateClient);
152             updateStmt.setInt(1, getClientId());
153             updateStmt.setInt(2, getJobId());
154             updateStmt.setString(3, getFirstName());
155             updateStmt.setString(4, getLastName());
156             updateStmt.setInt(5, getPnr());
157             updateStmt.setString(6, getAddress());
158             updateStmt.setString(7, getKennitala());
159             updateStmt.executeUpdate();
160             // conn.commit();
161         } catch (SQLException e) {
162             e.printStackTrace();
163         }
164     }
165 /**
166     * @param conn Connection
167     * @param kennitala String
168     */
169     public void saveDelete(Connection conn, String kennitala){
170         PreparedStatement deleteStmt = null;
171         try {
172             deleteStmt = conn.prepareStatement(ClientSQLStatements.deleteClient);
173             deleteStmt.setString(1, kennitala);
174             deleteStmt.executeUpdate();
175             // conn.commit();
176         } catch (SQLException e) {
177             e.printStackTrace();
178         }
179     }
180 /**
181     * @param conn Connection
182     */
183     public void findStaffID(Connection conn) {
184         PreparedStatement stmt = null;
185         try {
186             stmt = conn.prepareStatement(ClientSQLStatements.findClients);
187             stmt.executeQuery();
188         } catch (SQLException e) {
189             e.printStackTrace();
190         }
191     }
192 }
193 }
```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year  
project\Jfactor\src\is\unak\webapp\activerecords\client\ClientPerson.java
```

```
1 package is.unak.webapp.activerecords.client;
2
3 import java.sql.Date;
4 import java.sql.Blob;
5
6 /**
7 * Created by Yngvi Rafn Yngvason
8 * IDE used: IntelliJ IDEA
9 * Date: 22.1.2004
10 * Time: 18:12:59
11 */
12 public class ClientPerson {
13     /**
14      * This is a ClientPerson object, it holds information about a client.
15      */
16     private int clientId, jobId, age, pnr, caseId, divorcedHowOften, noOfChlidren,
loanId;
17     private int frequency, qty, ipId, stepDownId, cJobId;
18     private String gender,firstName,lastName, kennitala, spouseKt, address, phoneId,
cJobCompany, cJobPosition;
19     private String marriedTo, relName, frWhere, hlWhere, deptId, deptType, loanOwner,
analysisResult, frequencyDWM;
20     private String qtyDWM, drugId, hajWhere;
21     private Date cJobStarted, cJobQuit, firstSip, firstDrugAbuse, firstTreatment,
dischargeDate;
22     private String hasAjob, married, divorced, cLawMarriage, single, ownApmt, rentApmt,
friendRelApmt, homeless;
23     private String deptAmount;
24     private Blob reasons, comments;
25
26     public int getClientId() {
27         return clientId;
28     }
29
30     public void setClientId(int staffId) {
31         this.clientId = staffId;
32     }
33
34     public int getJobId() {
35         return jobId;
36     }
37
38     public void setJobId(int jobId) {
39         this.jobId = jobId;
40     }
41
42     public String getKennitala() {
43         return kennitala;
44     }
45
46     public void setKennitala(String kennitala) {
47         this.kennitala = kennitala;
48     }
49
50     public String getGender() {
51         return gender;
52     }
53
54     public void setGender(String gender) {
55         this.gender = gender;
56     }
57     public String getFirstName() {
58         return firstName;
59     }
60
61     public void setFirstName(String firstName) {
62         this.firstName = firstName;
63     }
64
65     public String getLastname() {
```

```
66         return lastName;
67     }
68
69     public void setLastName(String lastName) {
70         this.lastName = lastName;
71     }
72
73     public int getPnr() {
74         return pnr;
75     }
76
77     public void setPnr(int pnr) {
78         this.pnr = pnr;
79     }
80
81     public String getAddress() {
82         return address;
83     }
84
85     public void setAddress(String address) {
86         this.address = address;
87     }
88
89     public int getAge() {
90         return age;
91     }
92
93     public void setAge(int age) {
94         this.age = age;
95     }
96
97     public String getAnalysisResult() {
98         return analysisResult;
99     }
100
101    public void setAnalysisResult(String analysisResult) {
102        this.analysisResult = analysisResult;
103    }
104
105    public int getCaseId() {
106        return caseId;
107    }
108
109    public void setCaseId(int caseId) {
110        this.caseId = caseId;
111    }
112
113    public String getcJobCompany() {
114        return cJobCompany;
115    }
116
117    public void setcJobCompany(String cJobCompany) {
118        this.cJobCompany = cJobCompany;
119    }
120
121    public int getcJobId() {
122        return cJobId;
123    }
124
125    public void setcJobId(int cJobId) {
126        this.cJobId = cJobId;
127    }
128
129    public String getcJobPosition() {
130        return cJobPosition;
131    }
132
133    public void setcJobPosition(String cJobPosition) {
134        this.cJobPosition = cJobPosition;
135    }
136
137    public Date getcJobQuit() {
138        return cJobQuit;
139    }
140
141    public void setcJobQuit(Date cJobQuit) {
142        this.cJobQuit = cJobQuit;
```

```
143     }
144
145     public Date getcJobStarted() {
146         return cJobStarted;
147     }
148
149     public void setcJobStarted(Date cJobStarted) {
150         this.cJobStarted = cJobStarted;
151     }
152
153     public String getcLawMarriage() {
154         return cLawMarriage;
155     }
156
157     public void setcLawMarriage(String cLawMarriage) {
158         this.cLawMarriage = cLawMarriage;
159     }
160
161     public Blob getComments() {
162         return comments;
163     }
164
165     public void setComments(Blob comments) {
166         this.comments = comments;
167     }
168
169     public String getDeptAmount() {
170         return deptAmount;
171     }
172
173     public void setDeptAmount(String deptAmount) {
174         this.deptAmount = deptAmount;
175     }
176
177     public String getDeptId() {
178         return deptId;
179     }
180
181     public void setDeptId(String deptId) {
182         this.deptId = deptId;
183     }
184
185     public String getDeptType() {
186         return deptType;
187     }
188
189     public void setDeptType(String deptType) {
190         this.deptType = deptType;
191     }
192
193     public Date getDischargeDate() {
194         return dischargeDate;
195     }
196
197     public void setDischargeDate(Date dischargeDate) {
198         this.dischargeDate = dischargeDate;
199     }
200
201     public String getDivorced() {
202         return divorced;
203     }
204
205     public void setDivorced(String divorced) {
206         this.divorced = divorced;
207     }
208
209     public int getDivorcedHowOften() {
210         return divorcedHowOften;
211     }
212
213     public void setDivorcedHowOften(int divorcedHowOften) {
214         this.divorcedHowOften = divorcedHowOften;
215     }
216
217     public String getDrugId() {
218         return drugId;
219     }
```

```

220
221     public void setDrugId(String drugId) {
222         this.drugId = drugId;
223     }
224
225     public Date getFirstDrugAbuse() {
226         return firstDrugAbuse;
227     }
228
229     public void setFirstDrugAbuse(Date firstDrugAbuse) {
230         this.firstDrugAbuse = firstDrugAbuse;
231     }
232
233     public Date getFirstSip() {
234         return firstSip;
235     }
236
237     public void setFirstSip(Date firstSip) {
238         this.firstSip = firstSip;
239     }
240
241     public Date getFirstTreatment() {
242         return firstTreatment;
243     }
244
245     public void setFirstTreatment(Date firstTreatment) {
246         this.firstTreatment = firstTreatment;
247     }
248
249     public int getFrequency() {
250         return frequency;
251     }
252
253     public void setFrequency(int frequency) {
254         this.frequency = frequency;
255     }
256
257     public String getFrequencyDWM() {
258         return frequencyDWM;
259     }
260
261     public void setFrequencyDWM(String frequencyDWM) {
262         this.frequencyDWM = frequencyDWM;
263     }
264
265     public String getFriendRelApmt() {
266         return friendRelApmt;
267     }
268
269     public void setFriendRelApmt(String friendRelApmt) {
270         this.friendRelApmt = friendRelApmt;
271     }
272
273     public String getFrWhere() {
274         return frWhere;
275     }
276
277     public void setFrWhere(String frWhere) {
278         this.frWhere = frWhere;
279     }
280
281     public String getHasAjob() {
282         return hasAjob;
283     }
284
285     public void setHasAjob(String hasAjob) {
286         this.hasAjob = hasAjob;
287     }
288
289     public String getHajWhere() {
290         return hajWhere;
291     }
292
293     public void setHajWhere(String hajWhere) {
294         this.hajWhere = hajWhere;
295     }
296

```

```
297     public String getHlWhere() {
298         return hlWhere;
299     }
300
301     public void setHlWhere(String hlWhere) {
302         this.hlWhere = hlWhere;
303     }
304
305     public String getHomeless() {
306         return homeless;
307     }
308
309     public void setHomeless(String homeless) {
310         this.homeless = homeless;
311     }
312
313     public int getIpId() {
314         return ipId;
315     }
316
317     public void setIpId(int ipId) {
318         this.ipId = ipId;
319     }
320
321     public int getLoanId() {
322         return loanId;
323     }
324
325     public void setLoanId(int loanId) {
326         this.loanId = loanId;
327     }
328
329     public String getLoanOwner() {
330         return loanOwner;
331     }
332
333     public void setLoanOwner(String loanOwner) {
334         this.loanOwner = loanOwner;
335     }
336
337     public String getMarried() {
338         return married;
339     }
340
341     public void setMarried(String married) {
342         this.married = married;
343     }
344
345     public String getMarriedTo() {
346         return marriedTo;
347     }
348
349     public void setMarriedTo(String marriedTo) {
350         this.marriedTo = marriedTo;
351     }
352
353     public int getNoOfChlidren() {
354         return noOfChlidren;
355     }
356
357     public void setNoOfChlidren(int noOfChlidren) {
358         this.noOfChlidren = noOfChlidren;
359     }
360
361     public String getOwnApmt() {
362         return ownApmt;
363     }
364
365     public void setOwnApmt(String ownApmt) {
366         this.ownApmt = ownApmt;
367     }
368
369     public String getPhoneId() {
370         return phoneId;
371     }
372
373     public void setPhoneId(String phoneId) {
```

```
374         this.phoneId = phoneId;
375     }
376
377     public int getQty() {
378         return qty;
379     }
380
381     public void setQty(int qty) {
382         this.qty = qty;
383     }
384
385     public String getQtyDWM() {
386         return qtyDWM;
387     }
388
389     public void setQtyDWM(String qtyDWM) {
390         this.qtyDWM = qtyDWM;
391     }
392
393     public Blob getReasons() {
394         return reasons;
395     }
396
397     public void setReasons(Blob reasons) {
398         this.reasons = reasons;
399     }
400
401     public String getRelName() {
402         return relName;
403     }
404
405     public void setRelName(String relName) {
406         this.relName = relName;
407     }
408
409     public String getRentApmt() {
410         return rentApmt;
411     }
412
413     public void setRentApmt(String rentApmt) {
414         this.rentApmt = rentApmt;
415     }
416
417     public String getSingle() {
418         return single;
419     }
420
421     public void setSingle(String single) {
422         this.single = single;
423     }
424
425     public String getSpouseKt() {
426         return spouseKt;
427     }
428
429     public void setSpouseKt(String spouseKt) {
430         this.spouseKt = spouseKt;
431     }
432
433     public int getStepDownId() {
434         return stepDownId;
435     }
436
437     public void setStepDownId(int stepDownId) {
438         this.stepDownId = stepDownId;
439     }
440 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\client\ClientSQLStatements.java

```
1 package is.unak.webapp.activerecords.client;
2
3 /**
4  * Created by Yngvi Rafn Yngvason
5  * IDE used: IntelliJ IDEA
6  * Date: 22.1.2004
7  * Time: 17:28:37
8 */
9 public class ClientSQLStatements {
10     /**
11      * Statements affecting clients.
12     */
13     final static String insertClient = "INSERT INTO client VALUES (NULL, ?, ?, ?, ?, ?)";
14     final static String updateClient = "UPDATE client SET cFirstName = ?, cLastName = ?,  
cAge = ? WHERE cid = ?";
15     final static String findClients = "SELECT * FROM client";
16     final static String findClientName = "SELECT kennitala, firstname, lastname FROM  
client WHERE kennitala = ?";
17     final static String deleteClient = "DELETE FROM client WHERE kennitala = ?";
18
19 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\justicefacilities\Courts.java

```

1  package is.unak.webapp.activerecords.justicefacilities;
2
3  import java.util.Collection;
4  import java.util.ArrayList;
5  import java.sql.Connection;
6  import java.sql.PreparedStatement;
7  import java.sql.ResultSet;
8  import java.sql.SQLException;
9
10 /**
11  * Created by Yngvi Rafn Yngvason
12  * IDE used: IntelliJ IDEA
13  * Date: 30.1.2004
14  * Time: 12:20:12
15  */
16 public class Courts {
17
18     String courtName, courtAddress;
19     int courtPnr;
20
21 /**
22  * Returns a collection of phone types from the jails table.
23  * @param conn Connection
24  * @return Collection object
25  */
26 public static Collection getCourts(Connection conn) {
27     Collection courts = new ArrayList();
28     try {
29         PreparedStatement selectStmt;
30         Courts c;
31         selectStmt = conn.prepareStatement(JusticeSQLStatements.findCourts);
32         ResultSet result = selectStmt.executeQuery();
33         while (result.next()) {
34             c = new Courts();
35             c.setCourtName(result.getString("court_name"));
36             c.setCourtAddress(result.getString("court_address"));
37             c.setCourtPnr(result.getInt("court_pnr"));
38             courts.add(c);
39         }
40     } catch (SQLException e) {
41         e.printStackTrace();
42     } finally {
43     }
44     return courts;
45 }
46 /**
47  * @param conn Connection
48  * @param courtName String
49  * @return Courts object
50  */
51 public static Courts find(Connection conn, String courtName) {
52     PreparedStatement selectStmt = null;
53     Courts courts = null;
54     try {
55         selectStmt = conn.prepareStatement(JusticeSQLStatements.findCourt);
56         selectStmt.setString(1, courtName);
57         ResultSet result = selectStmt.executeQuery();
58         if (result.next()) {
59             courts = newCourt(result);
60         }
61     } catch (SQLException e) {
62         e.printStackTrace();
63     } finally {
64         return courts;
65     }
66 }
67 /**
68  * @param result ResultSet
69  * @return Courts object
70  * @throws SQLException

```

```

71      */
72  private static Courts newCourt(ResultSet result) throws SQLException {
73      Courts courts = new Courts();
74      courts.setCourtName(result.getString("court_name"));
75      courts.setCourtAddress(result.getString("court_address"));
76      courts.setCourtPnr(result.getInt("court_pnr"));
77      return courts;
78  }
79 /**
80 * @param conn Connection
81 */
82 public void saveInsert(Connection conn) {
83     try {
84         PreparedStatement insertStmt;
85         insertStmt = conn.prepareStatement(JusticeSQLStatements.insertCourt);
86         insertStmt.setString(1, getCourtName());
87         insertStmt.executeUpdate();
88         // conn.commit();
89     } catch (SQLException e) {
90         e.printStackTrace();
91     }
92 }
93 /**
94 * @param conn Connection
95 */
96 public void saveChange(Connection conn) {
97     try {
98         PreparedStatement insertStmt;
99         insertStmt = conn.prepareStatement(JusticeSQLStatements.updateCourt);
100        insertStmt.setString(1, getCourtName());
101        insertStmt.executeUpdate();
102        // conn.commit();
103    } catch (SQLException e) {
104        e.printStackTrace();
105    }
106 }
107 /**
108 * @param conn Connection
109 */
110 public void saveDelete(Connection conn) {
111     try {
112         PreparedStatement insertStmt;
113         insertStmt = conn.prepareStatement(JusticeSQLStatements.deleteCourt);
114         insertStmt.setString(1, getCourtName());
115         insertStmt.executeUpdate();
116         // conn.commit();
117     } catch (SQLException e) {
118         e.printStackTrace();
119     }
120 }
121 //Getters and setters for the courts table.
122 public String getCourtAddress() {
123     return courtAddress;
124 }
125
126 public void setCourtAddress(String courtAddress) {
127     this.courtAddress = courtAddress;
128 }
129
130 public String getCourtName() {
131     return courtName;
132 }
133
134 public void setCourtName(String courtName) {
135     this.courtName = courtName;
136 }
137
138 public int getCourtPnr() {
139     return courtPnr;
140 }
141
142 public void setCourtPnr(int courtPnr) {
143     this.courtPnr = courtPnr;
144 }
145
146
147 }

```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\justicefacilities\Jails.java

```

1  package is.unak.webapp.activerecords.justicefacilities;
2
3  import java.util.Collection;
4  import java.util.ArrayList;
5  import java.sql.Connection;
6  import java.sql.PreparedStatement;
7  import java.sql.ResultSet;
8  import java.sql.SQLException;
9
10 /**
11  * Created by Yngvi Rafn Yngvason
12  * IDE used: IntelliJ IDEA
13  * Date: 30.1.2004
14  * Time: 12:20:03
15  */
16 public class Jails {
17
18     String jailName, jailAddress;
19     int jailPnr;
20
21 /**
22  * Returns a collection of phone types from the jails table.
23  * @param conn Connection
24  * @return Collection
25  */
26 public static Collection getJails(Connection conn) {
27     Collection jails = new ArrayList();
28     try {
29         PreparedStatement selectStmt;
30         Jails j;
31         selectStmt = conn.prepareStatement(JusticeSQLStatements.findJails);
32         ResultSet result = selectStmt.executeQuery();
33         while (result.next()) {
34             j = new Jails();
35             j.setJailName(result.getString("jail_name"));
36             j.setJailAddress(result.getString("jail_address"));
37             j.setJailPnr(result.getInt("jail_pnr"));
38             jails.add(j);
39         }
40     } catch (SQLException e) {
41         e.printStackTrace();
42     } finally {
43     }
44     return jails;
45 }
46 /**
47  * Finds the requested jail name.
48  * @param conn Connection
49  * @param jailName String
50  * @return Jails object
51  */
52 public static Jails find(Connection conn, String jailName) {
53     PreparedStatement selectStmt = null;
54     Jails jails = null;
55     try {
56         selectStmt = conn.prepareStatement(JusticeSQLStatements.findJail);
57         selectStmt.setString(1, jailName);
58         ResultSet result = selectStmt.executeQuery();
59         if (result.next()) {
60             jails = newJail(result);
61         }
62     } catch (SQLException e) {
63         e.printStackTrace();
64     } finally {
65         return jails;
66     }
67 }
68 /**
69  * Prepares the Jails object with the result from the query.
70  * @param result ResultSet

```

```

71      * @return Jails object
72      * @throws SQLException
73      */
74      private static Jails newJail(ResultSet result) throws SQLException {
75          Jails jails = new Jails();
76          jails.setJailName(result.getString("jail_name"));
77          jails.setJailAddress(result.getString("jail_address"));
78          jails.setJailPnr(result.getInt("jail_pnr"));
79          return jails;
80      }
81      /**
82      * Inserts into the Jail table.
83      * @param conn Connection
84      */
85      public void saveInsert(Connection conn) {
86          try {
87              PreparedStatement insertStmt;
88              insertStmt = conn.prepareStatement(JusticeSQLStatements.insertJail);
89              insertStmt.setString(1, getJailName());
90              insertStmt.executeUpdate();
91              // conn.commit();
92          } catch (SQLException e) {
93              e.printStackTrace();
94          }
95      }
96      /**
97      * Saves changes into the Jail table.
98      * @param conn Connection
99      */
100     public void saveChange(Connection conn) {
101         try {
102             PreparedStatement insertStmt;
103             insertStmt = conn.prepareStatement(JusticeSQLStatements.updateJail);
104             insertStmt.setString(1, getJailName());
105             insertStmt.executeUpdate();
106             // conn.commit();
107         } catch (SQLException e) {
108             e.printStackTrace();
109         }
110     }
111     /**
112     * Deletes the selected row from Jail table.
113     * @param conn Connection
114     */
115     public void saveDelete(Connection conn) {
116         try {
117             PreparedStatement insertStmt;
118             insertStmt = conn.prepareStatement(JusticeSQLStatements.deleteJail);
119             insertStmt.setString(1, getJailName());
120             insertStmt.executeUpdate();
121             // conn.commit();
122         } catch (SQLException e) {
123             e.printStackTrace();
124         }
125     }
126     //Getters and setters for the jails table.
127     public String getJailAddress() {
128         return jailAddress;
129     }
130
131     public void setJailAddress(String jailAddress) {
132         this.jailAddress = jailAddress;
133     }
134
135     public String getJailName() {
136         return jailName;
137     }
138
139     public void setJailName(String jailName) {
140         this.jailName = jailName;
141     }
142
143     public int getJailPnr() {
144         return jailPnr;
145     }
146
147     public void setJailPnr(int jailPnr) {

```

```
148         this.jailPnr = jailPnr;  
149     }  
150 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\justicefacilities\JusticeSQLStatements.java

```

1  package is.unak.webapp.activerecords.justicefacilities;
2
3  /**
4   * Created by Yngvi Rafn Yngvason
5   * IDE used: IntelliJ IDEA
6   * Date: 30.1.2004
7   * Time: 12:22:40
8   */
9  public class JusticeSQLStatements {
10
11     //Statements that affect the jails table.
12     final static String insertJail = "INSERT INTO jails VALUES(?, ?, ?)";
13     final static String updateJail = "UPDATE jails WHERE jail_name = ?";
14     final static String findJails = "SELECT * FROM jails";
15     final static String findJail = "SELECT * FROM jails WHERE jail_name = ?";
16     final static String deleteJail = "DELETE FROM jails WHERE jail_name = ?";
17
18     //Statements that affect the jailtime table.
19     final static String insertJailTime = "INSERT INTO jails VALUES(?, ?, ?, ?, ?, ?)";
20     final static String updateJailTime = "UPDATE jails WHERE jail_name = ?";
21     final static String findJailTimes = "SELECT * FROM jails";
22     final static String findJailTime = "SELECT * FROM jails WHERE jail_name = ?";
23     final static String deleteJailTime = "DELETE FROM jails WHERE jail_name = ?";
24
25     //Statements that affect the courts table.
26     final static String insertCourt = "INSERT INTO courts VALUES(?, ?, ?)";
27     final static String updateCourt = "UPDATE courts WHERE court_name = ?";
28     final static String findCourts = "SELECT * FROM courts";
29     final static String findCourt = "SELECT * FROM courts WHERE court_name = ?";
30     final static String deleteCourt = "DELETE FROM courts WHERE court_name = ?";
31
32     //Statements that affect the court_cases table.
33     final static String insertCourtCase = "INSERT INTO court_cases VALUES(?, ?, ?, ?, ?, ? ,?)";
34     final static String updateCourtCase = "UPDATE court_cases WHERE case_id = ?";
35     final static String findCourtCases = "SELECT * FROM court_cases";
36     final static String findCourtCase = "SELECT * FROM court_cases WHERE case_id = ? ";
37     final static String deleteCourtCase = "DELETE FROM court_cases WHERE case_id = ?";
38 }

```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
project\Jfactor\src\is\unak\webapp\activerecords\staff\Staff.java
```

```

1  package is.unak.webapp.activerecords.staff;
2
3  import java.util.Collection;
4  import java.util.ArrayList;
5  import java.sql.Connection;
6  import java.sql.SQLException;
7  import java.sql.PreparedStatement;
8  import java.sql.ResultSet;
9
10 /**
11  * Created by Yngvi Rafn Yngvason
12  * IDE used: IntelliJ IDEA
13  * Date: 18.1.2004
14  * Time: 12:55:32
15  */
16 public class Staff extends StaffPerson {
17     /**
18      * @param conn Connection
19      * @return Collection
20      */
21     public static Collection getStaffMembers(Connection conn) {
22         Collection staff = new ArrayList();
23         try {
24             findStaff(conn, staff);
25         } catch (SQLException e) {
26             e.printStackTrace();
27         } finally {
28             return staff;
29         }
30     }
31     /**
32      *
33      * @param conn Connection
34      * @param staffMembers Collection
35      * @throws SQLException
36      */
37     private static void findStaff(Connection conn, Collection staffMembers) throws
SQLException {
38         PreparedStatement selectStmt;
39         Staff staff;
40         selectStmt = conn.prepareStatement(StaffSQLStatements.findStaffMembers);
41         ResultSet result = selectStmt.executeQuery();
42         while (result.next()) {
43             staff = newStaff(result);
44             staffMembers.add(staff);
45         }
46     }
47     /**
48      *
49      * @param conn Connection
50      * @param kennitala String
51      * @return Staff object
52      */
53     public static Staff find(Connection conn, String kennitala) {
54         PreparedStatement selectStmt = null;
55         Staff staff = null;
56         try {
57             selectStmt = conn.prepareStatement(StaffSQLStatements.findStaffName);
58             selectStmt.setString(1, kennitala);
59             ResultSet result = selectStmt.executeQuery();
60             if (result.next()) {
61                 staff = newStaff(result);
62             }
63         } catch (SQLException e) {
64             e.printStackTrace();
65         } finally {
66             return staff;
67         }
68     }
69 /**

```

```

70      *
71      * @param resultSet
72      * @return Staff object
73      * @throws SQLException
74      */
75      private static Staff newStaff(ResultSet result) throws SQLException {
76          Staff staff = new Staff();
77          staff.setKennitala(result.getString("kennitala"));
78          staff.setStaffId(result.getInt("staff_id"));
79          staff.setJobId(result.getInt("s_job_id"));
80          staff.setFirstName(result.getString("firstname"));
81          staff.setLastName(result.getString("lastname"));
82          staff.setPnr(result.getInt("pnr"));
83          staff.setAddress(result.getString("address"));
84          return staff;
85      }
86      /**
87      * @param conn Connection
88      */
89      public void saveInsert(Connection conn) {
90          PreparedStatement insertStmt = null;
91          try {
92              insertStmt = conn.prepareStatement(StaffSQLStatements.insertStaffMember);
93              insertStmt.setString(1, getKennitala());
94              insertStmt.setInt(2, getStaffId());
95              insertStmt.setInt(3, getJobId());
96              insertStmt.setString(4, getFirstName());
97              insertStmt.setString(5, getLastName());
98              insertStmt.setInt(6, getPnr());
99              insertStmt.setString(7, getAddress());
100             insertStmt.executeUpdate();
101             // conn.commit();
102         } catch (SQLException e) {
103             e.printStackTrace();
104         }
105     }
106     /**
107     * @param conn Connection
108     */
109    public void saveChange(Connection conn) {
110        PreparedStatement updateStmt = null;
111        try {
112            updateStmt = conn.prepareStatement(StaffSQLStatements.updateStaffMember);
113            updateStmt.setInt(1, getStaffId());
114            updateStmt.setInt(2, getJobId());
115            updateStmt.setString(3, getFirstName());
116            updateStmt.setString(4, getLastName());
117            updateStmt.setInt(5, getPnr());
118            updateStmt.setString(6, getAddress());
119            updateStmt.setString(7, getKennitala());
120            updateStmt.executeUpdate();
121            // conn.commit();
122        } catch (SQLException e) {
123            e.printStackTrace();
124        }
125    }
126    /**
127     * @param conn Connection
128     * @param kennitala String
129     */
130    public void saveDelete(Connection conn, String kennitala) {
131        PreparedStatement deleteStmt = null;
132        try {
133            deleteStmt = conn.prepareStatement(StaffSQLStatements.deleteStaffMember);
134            deleteStmt.setString(1, kennitala);
135            deleteStmt.executeUpdate();
136            // conn.commit();
137        } catch (SQLException e) {
138            e.printStackTrace();
139        }
140    }
141    }
142    /**
143     *
144     * @param conn Connection
145     */

```

```
147     public void findStaffID(Connection conn) {
148         PreparedStatement stmt = null;
149         try {
150             stmt = conn.prepareStatement(StaffSQLStatements.findStaffMembers);
151             stmt.executeQuery();
152         } catch (SQLException e) {
153             e.printStackTrace();
154         }
155     }
156 }
157 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\staff\StaffJobs.java

```

1  package is.unak.webapp.activerecords.staff;
2
3  import java.util.Collection;
4  import java.util.ArrayList;
5  import java.sql.*;
6
7  /**
8   * Created by Yngvi Rafn Yngvason
9   * IDE used: IntelliJ IDEA
10  * Date: 18.1.2004
11  * Time: 14:45:30
12  */
13 public class StaffJobs {
14
15     String sJobId, jobName, jobDescription;
16
17     /**
18      * Returns a collection of staff Jobs from the staff_jobs table.
19      * @param conn Connection
20      * @return
21      */
22     public static Collection getStaffJobs(Connection conn) {
23         Collection sJobs = new ArrayList();
24         try {
25             //Inserts the data into the SQL string
26             PreparedStatement selectStmt;
27             StaffJobs staffJobs;
28             selectStmt = conn.prepareStatement(StaffSQLStatements.findStaffJobs);
29             ResultSet result = selectStmt.executeQuery();
30             while (result.next()) {
31                 staffJobs = new StaffJobs();
32                 staffJobs.setSjobId(result.getString("s_job_id"));
33                 staffJobs.setJobName(result.getString("job_name"));
34                 staffJobs.setJobDescription(result.getString("job_description"));
35                 sJobs.add(staffJobs);
36             }
37         } catch (SQLException e) {
38             e.printStackTrace();
39         } finally {
40         }
41         return sJobs;
42     }
43     /**
44      * @param conn
45      * @param sJobId
46      * @return
47      */
48     public static StaffJobs find(Connection conn, String sJobId) {
49         PreparedStatement selectStmt = null;
50         StaffJobs staffJobs = null;
51         try {
52             selectStmt = conn.prepareStatement(StaffSQLStatements.findStaffJob);
53             selectStmt.setString(1, sJobId);
54             ResultSet result = selectStmt.executeQuery();
55             if (result.next()) {
56                 staffJobs = newStaffJob(result);
57             }
58         } catch (SQLException e) {
59             e.printStackTrace();
60         } finally {
61             return staffJobs;
62         }
63     }
64     /**
65      * @param resultSet
66      * @return
67      * @throws SQLException
68      */
69     private static StaffJobs newStaffJob(ResultSet result) throws SQLException {
70         StaffJobs sJobs = new StaffJobs();

```

```

71     sJobs.setSjobId(result.getString("s_job_id"));
72     sJobs.setJobName(result.getString("job_name"));
73     sJobs.setJobDescription(result.getString("job_description"));
74     return sJobs;
75 }
76 /**
77 * @param conn Connection
78 */
79 public void saveInsert(Connection conn) {
80     try {
81         PreparedStatement insertStmt;
82         insertStmt = conn.prepareStatement(StaffSQLStatements.insertStaffJob);
83         insertStmt.setString(1, getSjobId());
84         insertStmt.setString(2, getJobName());
85         insertStmt.setString(3, getJobDescription());
86         insertStmt.executeUpdate();
87         // conn.commit();
88     } catch (SQLException e) {
89         e.printStackTrace();
90     }
91 }
92 /**
93 * @param conn Connection
94 */
95 public void saveChange(Connection conn) {
96     try {
97         PreparedStatement updateStmt;
98         updateStmt = conn.prepareStatement(StaffSQLStatements.updateStaffJob);
99         updateStmt.setString(1, getJobName());
100        updateStmt.setString(2, getJobDescription());
101        updateStmt.setString(3, getSjobId());
102        updateStmt.executeUpdate();
103        // conn.commit();
104    } catch (SQLException e) {
105        e.printStackTrace();
106    }
107 }
108 /**
109 * @param conn Connection
110 * @param sJob String
111 */
112 public void saveDelete(Connection conn, String sJob) {
113     try {
114         PreparedStatement insertStmt;
115         insertStmt = conn.prepareStatement(StaffSQLStatements.deleteStaffJob);
116         insertStmt.setString(1, getSjobId());
117         insertStmt.executeUpdate();
118         // conn.commit();
119     } catch (SQLException e) {
120         e.printStackTrace();
121     }
122 }
123 /**
124 *Getters and setters for the staff_jobs table.
125 */
126
127
128 public String getJobDescription() {
129     return jobDescription;
130 }
131
132 public void setJobDescription(String jobDescription) {
133     this.jobDescription = jobDescription;
134 }
135
136 public String getJobName() {
137     return jobName;
138 }
139
140 public void setJobName(String jobName) {
141     this.jobName = jobName;
142 }
143
144 public String getSjobId() {
145     return sJobId;
146 }
147

```

```
148     public void setSjobId(String sJobId) {  
149         this.sJobId = sJobId;  
150     }  
151 }
```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
    project\Jfactor\src\is\unak\webapp\activerecords\staff\StaffJobTitles.java
```

```

1  package is.unak.webapp.activerecords.staff;
2
3  import java.util.Collection;
4  import java.util.ArrayList;
5  import java.sql.Connection;
6  import java.sql.PreparedStatement;
7  import java.sql.ResultSet;
8  import java.sql.SQLException;
9
10
11 /**
12 * @author Yngvi Rafn Yngvason
13 * IDE used: IntelliJ IDEA
14 * Date: 13.11.2003
15 * Time: 09:33:26
16 */
17
18 public class StaffJobTitles {
19
20     String kennitala, staffId, title;
21
22 /**
23 * Returns a collection of phone types from the phone_type_picklist table.
24 * @param conn
25 * @return Collection
26 */
27 public static Collection getStaffJobTitles(Connection conn) {
28     Collection jobTitles = new ArrayList();
29     try {
30         //Inserts the data into the SQL string
31         PreparedStatement selectStmt;
32         StaffJobTitles sjt;
33         selectStmt = conn.prepareStatement(StaffSQLStatements.findSjobTitles);
34         ResultSet result = selectStmt.executeQuery();
35         while (result.next()) {
36             sjt = new StaffJobTitles();
37             sjt.setKennitala(result.getString("kennitala"));
38             sjt.setStaffId(result.getString("staff_id"));
39             sjt.setTitle(result.getString("title"));
40             jobTitles.add(sjt);
41         }
42     } catch (SQLException e) {
43         e.printStackTrace();
44     } finally {
45     }
46     return jobTitles;
47 }
48 /**
49 * @param conn Connection
50 * @param sJobId String
51 * @return StaffJobTitles object
52 */
53 public static StaffJobTitles find(Connection conn, String sJobId) {
54     PreparedStatement selectStmt = null;
55     StaffJobTitles sjt = null;
56     try {
57         selectStmt = conn.prepareStatement(StaffSQLStatements.findSjobTitle);
58         selectStmt.setString(1, sJobId);
59         ResultSet result = selectStmt.executeQuery();
60         if (result.next()) {
61             sjt = new SJT(result);
62         }
63     } catch (SQLException e) {
64         e.printStackTrace();
65     } finally {
66         return sjt;
67     }
68 }
69 /**
70 * @param result ResultSet

```

```

71      * @return StaffJobTitles object
72      * @throws SQLException
73      */
74     private static StaffJobTitles newSJT(ResultSet result) throws SQLException {
75         StaffJobTitles sjt = new StaffJobTitles();
76         sjt.setKennitala(result.getString("kennitala"));
77         sjt.setStaffId(result.getString("staff_id"));
78         sjt.setTitle(result.getString("title"));
79         return sjt;
80     }
81     /**
82      * @param conn
83      */
84     public void saveInsert(Connection conn) {
85         try {
86             PreparedStatement insertStmt;
87             insertStmt = conn.prepareStatement(StaffSQLStatements.insertSjobTitle);
88             insertStmt.setString(1, getKennitala());
89             insertStmt.setString(2, getStaffId());
90             insertStmt.setString(3, getTitle());
91             insertStmt.executeUpdate();
92             // conn.commit();
93         } catch (SQLException e) {
94             e.printStackTrace();
95         }
96     }
97     /**
98      * @param conn
99      */
100    public void saveChange(Connection conn) {
101        try {
102            PreparedStatement updateStmt;
103            updateStmt = conn.prepareStatement(StaffSQLStatements.updateSjobTitle);
104            updateStmt.setString(1, getStaffId());
105            updateStmt.setString(2, getTitle());
106            updateStmt.setString(3, getKennitala());
107            updateStmt.executeUpdate();
108            // conn.commit();
109        } catch (SQLException e) {
110            e.printStackTrace();
111        }
112    }
113    /**
114      * @param conn
115      */
116    public void saveDelete(Connection conn) {
117        try {
118            PreparedStatement insertStmt;
119            insertStmt = conn.prepareStatement(StaffSQLStatements.deleteSjobTitle);
120            insertStmt.setString(1, getKennitala());
121            insertStmt.executeUpdate();
122            // conn.commit();
123        } catch (SQLException e) {
124            e.printStackTrace();
125        }
126    }
127
128    //Since these are the only getters and setters for this table,
129    //I will keep them here but not in a seperate class.
130    public String getKennitala() {
131        return kennitala;
132    }
133
134    public void setKennitala(String kennitala) {
135        this.kennitala = kennitala;
136    }
137
138    public String getStaffId() {
139        return staffId;
140    }
141
142    public void setStaffId(String staffId) {
143        this.staffId = staffId;
144    }
145
146    public String getTitle() {
147        return title;
148    }

```

```
148      }
149
150  public void setTitle(String title) {
151      this.title = title;
152  }
153 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\staff\StaffPerson.java

```

1  package is.unak.webapp.activerecords.staff;
2
3  /**
4   * Created by IntelliJ IDEA.
5   * User: Yngvi Rafn Yngvason
6   * Date: 6.11.2003
7   * Time: 09:52:38
8   * To change this template use Options | File Templates.
9   */
10  public class StaffPerson {
11      /**
12      * This is a StaffPerson object, it holds information about a staff member.
13      */
14      private int staffId, jobId, age, pnr;
15      private String firstName, lastName, kennitala, address;
16
17      public int getStaffId() {
18          return staffId;
19      }
20
21      public void setStaffId(int staffId) {
22          this.staffId = staffId;
23      }
24
25      public int getJobId() {
26          return jobId;
27      }
28
29      public void setJobId(int jobId) {
30          this.jobId = jobId;
31      }
32
33      public String getKennitala() {
34          return kennitala;
35      }
36
37      public void setKennitala(String kennitala) {
38          this.kennitala = kennitala;
39      }
40
41      public String getFirstName() {
42          return firstName;
43      }
44
45      public void setFirstName(String firstName) {
46          this.firstName = firstName;
47      }
48
49      public String getLastName() {
50          return lastName;
51      }
52
53      public void setLastName(String lastName) {
54          this.lastName = lastName;
55      }
56
57      public int getPnr() {
58          return pnr;
59      }
60
61      public void setPnr(int pnr) {
62          this.pnr = pnr;
63      }
64
65      public String getAddress() {
66          return address;
67      }
68
69      public void setAddress(String address) {
70          this.address = address;

```

```
71      }
72
73  public int getAge() {
74      return age;
75  }
76
77  public void setAge(int age) {
78      this.age = age;
79  }
80 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\staff\StaffSQLStatements.java

```

1  package is.unak.webapp.activerecords.staff;
2
3  /**
4   * Created by Yngvi Rafn Yngvason
5   * IDE used: IntelliJ IDEA
6   * Date: 22.1.2004
7   * Time: 17:17:40
8   */
9  public class StaffSQLStatements {
10
11     /**
12      * Statements affecting the staff table.
13      */
14     final static String insertStaffMember = "INSERT INTO staff VALUES (?, ?, ?, ?, ?, ?, ?, ?)";
15     final static String updateStaffMember = "UPDATE staff SET staff_id = ?, s_job_id = ?, "
16         + "firstname = ?, lastname = ?, pnr = ?, address = ? WHERE kennitala = ?";
17     final static String findStaffMembers = "SELECT * FROM staff";
18     final static String findStaffName = "SELECT * FROM staff WHERE kennitala = ?";
19     final static String deleteStaffMember = "DELETE FROM staff WHERE kennitala = ?";
20
21     /**
22      * Statements affecting the staff_jobs table.
23      */
24     final static String insertStaffJob = "INSERT INTO staff_jobs VALUES(?, ?, ?)";
25     final static String updateStaffJob = "UPDATE staff_jobs SET job_name = ?, "
26         + "job_description = ? WHERE s_job_id = ?";
27     final static String deleteStaffJob = "DELETE FROM staff_jobs WHERE s_job_id = ?";
28     final static String findStaffJobs = "SELECT * FROM staff_jobs";
29     final static String findStaffJob = "SELECT * FROM staff_jobs WHERE s_job_id = ?";
30
31     /**
32      * Statements affecting the s_job_titles table.
33      */
34     final static String insertSjobTitle = "INSERT INTO s_job_titles VALUES(?, ?, ?)";
35     final static String updateSjobTitle = "UPDATE s_job_titles SET staff_id = ?, title = "
36         + "? WHERE kennitala = ?";
37     final static String deleteSjobTitle = "DELETE FROM s_job_titles WHERE kennitala = ?";
38     final static String findSjobTitles = "SELECT * FROM s_job_titles";
39     final static String findSjobTitle = "SELECT * FROM s_job_titles WHERE s_job_id = ?";
40 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\DomainObject.java

```
1 package is.unak.webapp.activerecords;
2
3 /**
4 * Created by Yngvi Rafn Yngvason
5 * IDE used: IntelliJ IDEA
6 * Date: 8.2.2004
7 * Time: 13:18:34
8 */
9 public class DomainObject {
10     //This code is from Patterns of Enterprise Application Architecture by Martin Fowler
11     // (2003)
12     protected void markNew(){
13         UnitOfWork.getCurrent().registerNew(this);
14     }
15     protected void markClean(){
16         UnitOfWork.getCurrent().registerClean(this);
17     }
18     protected void markDirty(){
19         UnitOfWork.getCurrent().registerDirty(this);
20     }
21     protected void markRemoved(){
22         UnitOfWork.getCurrent().registerRemoved(this);
23     }
24     public void insert(){
25     }
26 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\ItemNotFoundException.java

```
1 package is.unak.webapp.activerecords;
2
3 /**
4 * Created by Yngvi Rafn Yngvason
5 * IDE used: IntelliJ IDEA
6 * Date: 26.1.2004
7 * Time: 15:04:14
8 */
9 public class ItemNotFoundException extends Exception {
10
11     public ItemNotFoundException(String s){
12         super(s);
13     }
14
15     ItemNotFoundException(){
16         this("");
17     }
18 }
```

C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year project\Jfactor\src\is\unak\webapp\activerecords\UnitOfWork.java

```

1  package is.unak.webapp.activerecords;
2
3  /**
4   * Created by Yngvi Rafn Yngvason
5   * IDE used: IntelliJ IDEA
6   * Date: 8.2.2004
7   * Time: 12:52:41
8   */
9  public class UnitOfWork {
10     //This code is from Patterns of Enterprise Application Architecture by Martin Fowler
11     // (2003)
12     /* private List newObjects = new ArrayList();
13      private List dirtyObjects = new ArrayList();
14      private List removedObjects = new ArrayList(); */
15     public void registerNew(DomainObject obj){
16         /* Assert.notNull("id not null", obj.getId());
17          Assert.isTrue("object not dirty", !dirtyObjects.contains(obj));
18          Assert.isTrue("object not removed", !removedObjects.contains(obj));
19          Assert.isTrue("object not already registered new", !newObjects.contains(obj));
20          newObjects.add(obj);
21      }
22
23     public void registerDirty(DomainObject obj){
24         /* Assert.notNull("id not null", obj.getId());
25         Assert.isTrue("object not removed", !removedObjects.contains(obj));
26         if(!dirtyObjects.contains(obj) && !newObjects.contains(obj)){
27             dirtyObjects.add(obj);
28         }
29     }
30
31     public void registerRemoved(DomainObject obj){
32         /* Assert.notNull("id not null", obj.getId());
33         if(newObjects.remove(obj))
34             return;
35         dirtyObjects.remove(obj);
36         if(!removedObjects.contains(obj)) {
37             removedObjects.add(obj);
38         }
39     }
40
41     public void registerClean(DomainObject obj){
42         /* Assert.notNull("id not null", obj.getId());
43     }
44
45
46     //This code is from Patterns of Enterprise Application Architecture by Martin Fowler
47     // (2003)
48     private static ThreadLocal current = new ThreadLocal();
49
50     public static void newCurrent(){
51         setCurrent(new UnitOfWork());
52     }
53
54     public static void setCurrent(UnitOfWork uow){
55         current.set(uow);
56     }
57
58     public static UnitOfWork getCurrent(){
59         return (UnitOfWork) current.get();
60     }
61 }
```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year  
project\Jfactor\src\is\unak\webapp\database\DatabaseConnection.java
```

```
1 package is.unak.webapp.database;  
2  
3 import java.sql.SQLException;  
4 import java.sql.Connection;  
5 import java.sql.DriverManager;  
6 import java.sql.*;  
7  
8 /**  
9  * Created by Yngvi Rafn Yngvason  
10 * IDE used IntelliJ IDEA  
11 * Date: 25.9.2003  
12 * Time: 10:20:05  
13 */  
14 public class DatabaseConnection {  
15     static Connection conn;  
16     /**  
17      * @return Connection  
18     */  
19     public Connection openConnection() {  
20         try {  
21             Class.forName("com.mysql.jdbc.Driver").newInstance();  
22             conn =  
DriverManager.getConnection("jdbc:mysql://localhost/samhjalpdb?user=yngvi");  
23             conn.setAutoCommit(false);  
24             return conn;  
25         } catch (InstantiationException e) {  
26             e.printStackTrace();  
27         } catch (IllegalAccessException e) {  
28             e.printStackTrace();  
29         } catch (ClassNotFoundException e) {  
30             e.printStackTrace();  
31         } catch (SQLException e) {  
32             e.printStackTrace();  
33         }  
34         return conn;  
35     }  
36     /**  
37      * @param conn Connection  
38      * @throws SQLException  
39     */  
40     public static void close(Connection conn) throws SQLException {  
41         if (conn != null) {  
42             conn.close();  
43         }  
44     }  
45 }
```

```
C:\Documents and Settings\Yngvi Rafn Yngvason\My Documents\6th_semester\Final year
    project\Jfactor\src\is\unak\webapp\presentation\KennitalaCheck.java
```

```

1  package is.unak.webapp.presentation;
2
3  import javax.servlet.http.HttpServletRequest;
4  import javax.servlet.http.HttpServletResponse;
5  import javax.servlet.http.HttpServlet;
6  import javax.servlet.ServletException;
7  import java.io.IOException;
8  import java.io.PrintWriter;
9
10 /**
11  * @author Created by Yngvi Rafn Yngvason
12  * IDE used: IntelliJ IDEA
13  * Date: 19.2.2004
14  * Time: 15:18:04
15  */
16
17 /**
18  * This class is for checking the correctness of the icelanding social security number
19  * called kennitala.
20  * It will return an error page if the checkSum number does not equal the nineth number
21  * in the kennitala.
22  * But if the numbers match the request will be forwarded to another servlet that will
23  * process the request
24  * and return a page with the requested data.
25  */
26
27 public class KennitalaCheck extends HttpServlet {
28
29     int one, two, three, four, five, six, seven, eight, nine, ten;
30     int sum1, sum2, sum3, sum4, sum5, sum6, sum7, sum8;
31     int totalSum, remainder, checkSum;
32     int divisor = 11;
33     int check1 = 3;
34     int check2 = 2;
35     int check3 = 7;
36     int check4 = 6;
37     int check5 = 5;
38     int check6 = 4;
39     int check7 = 3;
40     int check8 = 2;
41
42     /**
43      * @param req Request
44      * @param res Response
45      * @throws ServletException
46      * @throws IOException
47      */
48
49     public void doPost(HttpServletRequest req, HttpServletResponse res)
50         throws ServletException, IOException {
51
52         res.setContentType("text/html");
53         PrintWriter out = res.getWriter();
54
55         /**Get the kennitala and split it up such that each number is placed into its own
56         variable. */
57         String kennitala = req.getParameter("kennitala");
58         one = Integer.parseInt(req.getParameter(String.valueOf(kennitala.charAt(0))));
59         two = Integer.parseInt(req.getParameter(String.valueOf(kennitala.charAt(1))));
60         three = Integer.parseInt(req.getParameter(String.valueOf(kennitala.charAt(2))));
61         four = Integer.parseInt(req.getParameter(String.valueOf(kennitala.charAt(3))));
62         five = Integer.parseInt(req.getParameter(String.valueOf(kennitala.charAt(4))));
63         six = Integer.parseInt(req.getParameter(String.valueOf(kennitala.charAt(5))));
64         seven = Integer.parseInt(req.getParameter(String.valueOf(kennitala.charAt(6))));
65         eight = Integer.parseInt(req.getParameter(String.valueOf(kennitala.charAt(7))));
66         nine = Integer.parseInt(req.getParameter(String.valueOf(kennitala.charAt(8))));
67
68         /**Check numbers multiplied with the individual numbers from the kennitala. */
69         sum1 = one * check1;
70         sum2 = two * check2;
71         sum3 = three * check3;
72         sum4 = four * check4;
73
74         totalSum = sum1 + sum2 + sum3 + sum4;
75
76         remainder = totalSum % divisor;
77
78         if (remainder == 0) {
79             // Forward the request to another servlet
80             // ...
81         } else {
82             // Create an error page
83             // ...
84         }
85     }
86 }
```

```

67     sum5 = five * check5;
68     sum6 = six * check6;
69     sum7 = seven * check7;
70     sum8 = eight * check8;
71
72     /**The checkSum should be equal to the nineth number in the kennitala. */
73     totalSum = sum1 + sum2 + sum3 + sum4 + sum5 + sum6 + sum7 + sum8;
74     remainder = totalSum % divisor;
75     checkSum = divisor - remainder;
76
77     if (checkSum == divisor) {
78         checkSum = 0;
79     }
80
81     /** The checkSum should be equal to the nineth number in the kennitala. */
82     // The if statement tests that.
83     if (checkSum != nine){
84         out.println("<html>");
85         out.println("<head><title>Villa i kennitölu</title></head>");
86         out.println("<body>");
87         out.println("Það er villa i kennitölunni, reyndu aftur.");
88         out.println("checkSum er: " + checkSum + " og vartalan er: " + nine);
89         out.println("</body></html>");
90     } else {
91         //
92         //
93         //
94         //
95         //
96         //
97         return;
98     }
99 }
100
101
102 }
```

```

<html>
<head>
<title>testpage.gif</title>
<meta http-equiv="Content-Type" content="text/html;">
<meta name="description" content="FW MX DW MX HTML">
<!-- Fireworks MX Dreamweaver MX target. Created Wed Apr 14 18:20:16 GMT+0000 (GMT Standard
Time) 2004-->
<script language="JavaScript">
<!--
function MM_findObj(n, d) { //v4.01
    var p,i,x; if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
        d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
    if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
    for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
    if(!x && d.getElementById) x=d.getElementById(n); return x;
}
/* Functions that swaps images. */
function MM_swapImage() { //v3.0
    var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i<(a.length-2);i+=3)
        if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src;
    x.src=a[i+2];}
}
function MM_swapImgRestore() { //v3.0
    var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
}

/* Functions that swaps down images. */
function MM_nbGroup(event, grpName) { //v6.0
var i,img,nbArr,args=MM_nbGroup.arguments;
if (event == "init" && args.length > 2) {
    if ((img = MM_findObj(args[2])) != null && !img.MM_init) {
        img.MM_init = true; img.MM_up = args[3]; img.MM_dn = img.src;
        if ((nbArr = document[grpName]) == null) nbArr = document[grpName] = new Array();
        nbArr[nbArr.length] = img;
        for (i=4; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
            if (!img.MM_up) img.MM_up = img.src;
            img.src = img.MM_dn = args[i+1];
            nbArr[nbArr.length] = img;
        }
    } else if (event == "over") {
        document.MM_nbOver = nbArr = new Array();
        for (i=1; i < args.length-1; i+=3) if ((img = MM_findObj(args[i])) != null) {
            if (!img.MM_up) img.MM_up = img.src;
            img.src = (img.MM_dn && args[i+2]) ? args[i+2] : ((args[i+1])?args[i+1] : img.MM_up);
            nbArr[nbArr.length] = img;
        }
    } else if (event == "out" ) {
        for (i=0; i < document.MM_nbOver.length; i++) { img = document.MM_nbOver[i]; img.src =
(img.MM_dn) ? img.MM_dn : img.MM_up; }
        else if (event == "down") {
            nbArr = document[grpName];
            if (nbArr) for (i=0; i < nbArr.length; i++) { img=nbArr[i]; img.src = img.MM_up; img.MM_dn
= 0; }
            document[grpName] = nbArr = new Array();
            for (i=2; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = img.MM_dn = (args[i+1])? args[i+1] : img.MM_up;
                nbArr[nbArr.length] = img;
            }
        }
    }
}

/* Functions that handle preload. */
function MM_preloadImages() { //v3.0
    var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
        var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
        if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}

//-->
</script>
</head>

```

```

<body bgcolor="#ffffff"
onLoad="MM_preloadImages('../images/testpage/heim_f2.gif','../images/testpage/heim_f4.gif','..
/images/testpage/heim_f3.gif','../images/testpage/innlogin_f2.gif','../images/testpage/testpage_
r3_c6_f2.gif','../images/testpage/testpage_r3_c6_f4.gif','../images/testpage/testpage_r3_c6_f
3.gif','../images/testpage/vidtal2_f2.gif','..../images/testpage/vidtal2_f4.gif','../images/test
page/vidtal2_f3.gif','../images/testpage/testpage_r3_c10_f2.gif','..../images/testpage/testpage_
r3_c10_f4.gif','..../images/testpage/testpage_r3_c10_f3.gif','..../images/testpage/testpage_r3_c12_
f2.gif','..../images/testpage/testpage_r3_c12_f4.gif','..../images/testpage/testpage_r3_c12_f3.gi
f');">
<!--The following section is an HTML table which reassembles the sliced image in a browser.-->
<!--Copy the table section including the opening and closing table tags, and paste the data
where-->
<!--you want the reassembled image to appear in the destination document. -->
<===== BEGIN COPYING THE HTML HERE =====>
<table border="0" cellpadding="0" cellspacing="0" width="800">
<!-- fwtable fwsid="Umsjonarmadur_top.png" fwbase="testpage.gif" fwstyle="Dreamweaver" fwdocid
= "742308039" fnested="0" -->
<tr>
<!-- Shim row, height 1. -->
<td></td>
</tr>

<tr><!-- row 1 -->
<td colspan="14"></td>
<td></td>
</tr>
<tr><!-- row 2 -->
<td rowspan="2" style="background-color: #cccccc;">
<td colspan="13" style="background-color: #cccccc;">
<td></td>
</tr>
<tr><!-- row 3 -->
<td height="21" style="background-color: #cccccc;"><a href="#" onMouseOut="MM_nbGroup('out');" onMouseOver="MM_nbGroup('over','heim','../images/testpage/heim_f2.gif','../images/testpage/heim_f4.gif',1);"
onClick="MM_nbGroup('down','navbar1','heim','../images/testpage/heim_f3.gif',1);"></a></td>
<td style="background-color: #cccccc;">
<td style="background-color: #cccccc;"><a href="#" onMouseOut="MM_swapImgRestore();" onMouseOver="MM_swapImage('innlogin','','../images/testpage/innlogin_f2.gif',1);"></a></td>
<td style="background-color: #cccccc;">
<td style="background-color: #cccccc;"><a href="#" onMouseOut="MM_nbGroup('out');" onMouseOver="MM_nbGroup('over','testpage_r3_c6','../images/testpage/testpage_r3_c6_f2.gif','../images/testpage/testpage_r3_c6_f4.gif',1);"
onClick="MM_nbGroup('down','navbar1','testpage_r3_c6','../images/testpage/testpage_r3_c6_f3.gif',1);"></a></td>
<td style="background-color: #cccccc;">
<td style="background-color: #cccccc;"><a href="#" onMouseOut="MM_nbGroup('out');" onMouseOver="MM_nbGroup('over','vidtal2','../images/testpage/vidtal2_f2.gif','../images/testpage/vidtal2_f4.gif',1);"
onClick="MM_nbGroup('down','navbar1','vidtal2','../images/testpage/vidtal2_f3.gif',1);"></a></td>

```

```

<td></td>
<td><a href="#" onMouseOut="MM_nbGroup('out');"
onMouseOver="MM_nbGroup('over','testpage_r3_c10','','../images/testpage/testpage_r3_c10_f2.gif','
../images/testpage/testpage_r3_c10_f4.gif',1);"
onClick="MM_nbGroup('down','navbar1','testpage_r3_c10','','../images/testpage/testpage_r3_c10_f3.
gif',1);"></a></td>
<td></td>
<td><a href="#" onMouseOut="MM_nbGroup('out');"
onMouseOver="MM_nbGroup('over','testpage_r3_c12','','../images/testpage/testpage_r3_c12_f2.gif','
../images/testpage/testpage_r3_c12_f4.gif',1);"
onClick="MM_nbGroup('down','navbar1','testpage_r3_c12','','../images/testpage/testpage_r3_c12_f3.
gif',1);"></a></td>
<td colspan="2"></td>
<td></td>
</tr>
<!-- This table was automatically created with Macromedia Fireworks -->
<!-- http://www.macromedia.com -->
</table>
<===== STOP COPYING THE HTML HERE =====>
<p><font size="2" face="Verdana, Arial, Helvetica, sans-serif">Jfactor is a client
centered system for addiction rehabilitation centres.<br>
It is focused on recording all essential information regarding the client's
addiction as well as<br>
his health, finances, family, housing and social situation.</font></p>
<p><font size="2" face="Verdana, Arial, Helvetica, sans-serif">This information
will be used in data mining for both known and unknown patterns in the<br>
addicts life.</font></p>
<p><font size="2" face="Verdana, Arial, Helvetica, sans-serif">Enjoy Jfactor.</font></p>
<p>&nbsp;</p>
<p><font size="1" face="Verdana, Arial, Helvetica, sans-serif">To see other already
created please check out the following links:</font></p>
<p><font size="1" face="Verdana, Arial, Helvetica, sans-serif"><a
href="../html/login.htm">Login
page</a> - <a href="../html/kerfisstjori_home.htm">System Administrator homepage</a>
- <a href="../html/kerfisstjori_staff.htm">System Administrator staffpage</a>
- <a href="../html/jspstestpage.htm">JSP test page</a> </font></p>
<p>&nbsp;</p>
<map name="testpage_r1_c1Map">
<area shape="rect" coords="5,0,184,59" href="jfactor_info.htm">
<area shape="rect" coords="689,3,825,51" href="www.samhjalp.is">
</map>
</body>
</html>

```

jsptestpage.htm

```

<html>
<head>
    <title>JSP test page</title>
    <meta http-equiv="Content-Type" content="text/html;">
    <META http-equiv=Content-Language content=is>
    <META http-equiv=pragma content=no-cache>
    <META http-equiv=Cache-Control content=must-revalidate>
    <META http-equiv=Cache-Control content=no-cache>
    <META http-equiv=Cache-Control content=no-store>
    <META http-equiv=expires content=-1>

    <script language="JavaScript" type="text/JavaScript">
    <!--
    function MM_reloadImage() { //v3.0
        var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
        var i,j=d.MM_p.length,a=MM_reloadImage.arguments; for(i=0; i<a.length; i++)
            if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
    }

    function MM_findObj(n, d) { //v4.01
        var p,i,x; if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
            d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
        if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
        for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
        if(!x && d.getElementById) x=d.getElementById(n); return x;
    }

    function MM_nbGroup(event, grpName) { //v6.0
        var i,img,nbArr,args=MM_nbGroup.arguments;
        if (event == "init" && args.length > 2) {
            if ((img = MM_findObj(args[2])) != null && !img.MM_init) {
                img.MM_init = true; img.MM_up = args[3]; img.MM_dn = img.src;
                if ((nbArr = document[grpName]) == null) nbArr = document[grpName] = new Array();
                nbArr[nbArr.length] = img;
                for (i=4; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                    if (!img.MM_up) img.MM_up = img.src;
                    img.src = img.MM_dn = args[i+1];
                    nbArr[nbArr.length] = img;
                }
            } else if (event == "over") {
                document.MM_nbOver = nbArr = new Array();
                for (i=1; i < args.length-1; i+=3) if ((img = MM_findObj(args[i])) != null) {
                    if (!img.MM_up) img.MM_up = img.src;
                    img.src = (img.MM_dn && args[i+2]) ? args[i+2] : ((args[i+1])? args[i+1] : img.MM_up);
                    nbArr[nbArr.length] = img;
                }
            } else if (event == "out" ) {
                for (i=0; i < document.MM_nbOver.length; i++) {
                    img = document.MM_nbOver[i]; img.src = (img.MM_dn) ? img.MM_dn : img.MM_up; }
            } else if (event == "down") {
                nbArr = document[grpName];
                if (nbArr)
                    for (i=0; i < nbArr.length; i++) { img=nbArr[i]; img.src = img.MM_up; img.MM_dn = 0; }
                document[grpName] = nbArr = new Array();
                for (i=2; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                    if (!img.MM_up) img.MM_up = img.src;
                    img.src = img.MM_dn = (args[i+1])? args[i+1] : img.MM_up;
                    nbArr[nbArr.length] = img;
                }
            }
        }

        function MM_swapImgRestore() { //v3.0
            var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
        }

        function MM_swapImage() { //v3.0
            var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i<(a.length-2);i+=3)
                if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src;
                x.src=a[i+2];}
        }
    //-->
    </script>
</head>

```

```

<body bgcolor="white"
onLoad="MM_preloadImages('../images/testpage/heim_f3.gif','../images/testpage/heim_f2.gif','..
/images/testpage/heim_f4.gif','../images/testpage/innlogin_f2.gif','../images/testpage/testpage_
r3_c6_f3.gif','../images/testpage/testpage_r3_c6_f2.gif','../images/testpage/testpage_r3_c6_f
4.gif','../images/testpage/vidtal2_f3.gif','..7images/testpage/vidtal2_f2.gif','../images/test
page/vidtal2_f4.gif','../images/testpage/testpage_r3_c10_f3.gif','..7images/testpage/testpage_
r3_c10_f2.gif','..7images/testpage/testpage_r3_c10_f4.gif','..7images/testpage/testpage_r3_c12_
f3.gif','..7images/testpage/testpage_r3_c12_f2.gif','..7images/testpage/testpage_r3_c12_f4.gi
f')">
<table border="0" cellpadding="0" cellspacing="0" width="800">
  <!-- fwtable fwsrc="Umsjonarmadur_top.png" fwbase="testpage.gif" fwstyle="Dreamweaver"
fwdocid = "742308039" fnested="0" -->
  <tr>
    <!-- Shim row, height 1. -->
    <td></td>
    <td></td>
  </tr>
  <tr>
    <!-- row 1 -->
    <td colspan="14"></td>
    <td></td>
  </tr>
  <tr>
    <!-- row 2 -->
    <td rowspan="2"></td>
    <td colspan="13"></td>
    <td></td>
  </tr>
  <tr>
    <!-- row 3 -->
    <td><a href="../html/jspptestpage.htm" target="_top"
onClick="MM_nbGroup('down','navbar1','heim','../images/testpage/heim_f3.gif',1);"
onMouseOver="MM_nbGroup('over','heim','../images/testpage/heim_f2.gif','../images/testpage/he
im_f4.gif',1);" onMouseOut="MM_nbGroup('out');"></a></td>
    <td></td>
    <td><a href="#" onMouseOut="MM_swapImgRestore()"></a></td>
    <td></td>
    <td><a href="javascipt:;" target="_top"
onClick="MM_nbGroup('down','navbar1','testpage_r3_c6','../images/testpage/testpage_r3_c6_f3.gi
f',1);"
onMouseOver="MM_nbGroup('over','testpage_r3_c6','../images/testpage/testpage_r3_c6_f2.gif','..
/images/testpage/testpage_r3_c6_f4.gif',1);" onMouseOut="MM_nbGroup('out');"></a></td>
    <td></td>
    <td><a href="javascipt:;" target="_top"
onClick="MM_nbGroup('down','navbar1','vidtal2','..7images/testpage/vidtal2_f3.gif',1);"
onMouseOver="MM_nbGroup('over','vidtal2','..7images/testpage/vidtal2_f2.gif','..7images/testpa
ge/vidtal2_f4.gif',1);" onMouseOut="MM_nbGroup('out');"></a></td>

```

```

<td></td>
    <td><a href="javascript:;" target="_top"
onClick="MM_nbGroup('down','navbar1','testpage_r3_c10','../images/testpage/testpage_r3_c10_f3.
gif',1);"
onMouseOver="MM_nbGroup('over','testpage_r3_c10','../images/testpage/testpage_r3_c10_f2.gif','
../images/testpage/testpage_r3_c10_f4.gif',1);" onMouseOut="MM_nbGroup('out');"></a></td>
    <td></td>
    <td><a href="javascript:;" target="_top"
onClick="MM_nbGroup('down','navbar1','testpage_r3_c12','../images/testpage/testpage_r3_c12_f3.
gif',1);"
onMouseOver="MM_nbGroup('over','testpage_r3_c12','../images/testpage/testpage_r3_c12_f2.gif','
../images/testpage/testpage_r3_c12_f4.gif',1);" onMouseOut="MM_nbGroup('out');"></a></td>
    <td colspan="2"></td>
    <td></td>
</tr>
<!-- This table was automatically created with Macromedia Fireworks --&gt;
<!-- http://www.macromedia.com --&gt;
&lt;/table&gt;
&lt;p&gt;&lt;font face="Verdana, Arial, Helvetica, sans-serif"&gt;&lt;b&gt;Add in the information
and press Innskr&amp;aacute;.&lt;/b&gt;&lt;/font&gt;&lt;/p&gt;
&lt;p&gt;&lt;font face="Verdana, Arial, Helvetica, sans-serif"&gt;The result should be the
&lt;b&gt;first name&lt;/b&gt; along with the&lt;b&gt; postnumber&lt;/b&gt; and&lt;b&gt; city&lt;/b&gt;.&lt;/font&gt;&lt;/p&gt;
&lt;form name="pnrandcity" method="post" action="../jsp/jspptestpage.jsp"&gt;
&lt;table width="423" border="0" cellspacing="2" cellpadding="0"&gt;
    &lt;tr bgcolor="#CCCCCC"&gt;
        &lt;td width="182" height="30"&gt;&lt;div align="right"&gt;&lt;font size="2" face="Verdana, Arial,
Helvetica, sans-serif"&gt;Kennitala:
            &lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
        &lt;td width="241" height="30"&gt; &lt;div align="left"&gt;&lt;font size="2" face="Verdana, Arial,
Helvetica, sans-serif"&gt;
            &lt;input name="kennitala" type="text" id="kennitala" size="11" maxlength="11"&gt;
        &lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
    &lt;/tr&gt;
    &lt;tr bgcolor="#CCCCCC"&gt;
        &lt;td height="30"&gt;&lt;div align="right"&gt;&lt;font face="Verdana, Arial, Helvetica, sans-
serif"&gt;&lt;font size="2"&gt;First
            name:&lt;/font&gt;&lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
        &lt;td height="30"&gt;&lt;div align="left"&gt;&lt;font face="Verdana, Arial, Helvetica, sans-serif"&gt;&lt;font
size="2"&gt;
            &lt;input name="firstName" type="text" id="firstName2"&gt;
        &lt;/font&gt;&lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
    &lt;/tr&gt;
    &lt;tr bgcolor="#CCCCCC"&gt;
        &lt;td height="30"&gt;&lt;div align="right"&gt;&lt;font face="Verdana, Arial, Helvetica, sans-
serif"&gt;&lt;font size="2"&gt;Last
            name:&lt;/font&gt;&lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
        &lt;td height="30"&gt;&lt;div align="left"&gt;&lt;font face="Verdana, Arial, Helvetica, sans-serif"&gt;&lt;font
size="2"&gt;
            &lt;input name="lastName" type="text" id="lastName2"&gt;
        &lt;/font&gt;&lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
    &lt;/tr&gt;
    &lt;tr bgcolor="#CCCCCC"&gt;
        &lt;td height="30"&gt;&lt;div align="right"&gt;&lt;font face="Verdana, Arial, Helvetica, sans-
serif"&gt;&lt;font size="2"&gt;Address:&lt;/font&gt;&lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
        &lt;td height="30"&gt;&lt;div align="left"&gt;&lt;font face="Verdana, Arial, Helvetica, sans-serif"&gt;&lt;font
size="2"&gt;
            &lt;input name="address" type="text" id="address2"&gt;
        &lt;/font&gt;&lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
    &lt;/tr&gt;
    &lt;tr bgcolor="#CCCCCC"&gt;
        &lt;td height="30"&gt;&lt;div align="right"&gt;&lt;font face="Verdana, Arial, Helvetica, sans-
serif"&gt;&lt;font size="2"&gt;Pnr:&lt;/font&gt;&lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
        &lt;td height="30"&gt;&lt;div align="left"&gt;&lt;font face="Verdana, Arial, Helvetica, sans-serif"&gt;&lt;font
size="2"&gt;
            &lt;input name="pnr" type="text" id="pnr2" size="6" maxlength="6"&gt;
        &lt;/font&gt;&lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
    &lt;/tr&gt;
    &lt;tr bgcolor="#CCCCCC"&gt;
        &lt;td height="30"&gt;&lt;div align="right"&gt;&lt;font face="Verdana, Arial, Helvetica, sans-
serif"&gt;&lt;font size="2"&gt;City:&lt;/font&gt;&lt;/font&gt;&lt;/div&gt;&lt;/td&gt;
</pre>

```

```
<td height="30"><div align="left"><font face="Verdana, Arial, Helvetica, sans-serif"><font size="2">
    <input name="city" type="text" id="city2">
</font></font></div></td>
</tr>
<tr bgcolor="#FFFFFF">
    <td height="30"><div align="right"><font face="Verdana, Arial, Helvetica, sans-serif"><font size="2"></font></font></div></td>
    <td height="30"><div align="right"><font face="Verdana, Arial, Helvetica, sans-serif"><font size="2">
        <input name="submit" type="submit" value="Innskr&aacute;" >
    </font></font></div></td>
</tr>
</table>

</form>

<p><font size="1" face="Verdana, Arial, Helvetica, sans-serif">To see other already
    created please check out the following links:</font></p>
<p><font size="1" face="Verdana, Arial, Helvetica, sans-serif"><a href="../html/login.htm">Login
    page</a> - <a href="../html/kerfisstjori_home.htm">System Administrator homepage</a>
    - <a href="../html/kerfisstjori_staff.htm">System Administrator staffpage</a>
</font></p>
<p> </p>
<map name="testpage_r1_c1Map">
    <area shape="rect" coords="690,3,797,49" href="http://www.samhjalp.is" alt="Go to Samhj&aacute;lp's
    main web site.">
    <area shape="rect" coords="2,1,182,51" href="jfactor_info.htm">
</map>
</body>
</html>
```

kerfisstjori_home.htm

```

<html>
<head>
<title>kerfisstjori_home.jpg</title>
<meta http-equiv="Content-Type" content="text/html;">
<meta name="description" content="FW MX DW MX HTML">
<META http-equiv=Content-Language content=is>
<META http-equiv=pragma content=no-cache>
<META http-equiv=Cache-Control content=must-revalidate>
<META http-equiv=Cache-Control content=no-cache>
<META http-equiv=Cache-Control content=no-store>
<META http-equiv=expires content=-1>

<!-- Fireworks MX Dreamweaver MX target. Created Tue Apr 13 15:55:25 GMT+0000 (GMT Standard
Time) 2004--&gt;
&lt;script language="JavaScript"&gt;
&lt;!--
function MM_findObj(n, d) { //v4.01
    var p,i,x; if(!d) d=document; if((p=n.indexOf("?"))&gt;0&amp;&amp;parent.frames.length) {
        d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
    if(!(x=d[n])&amp;&amp;d.all) x=d.all[n]; for (i=0;!x&amp;i&lt;d.forms.length;i++) x=d.forms[i][n];
    for(i=0;!x&amp;&amp;d.layers&amp;&amp;i&lt;d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
    if(!x &amp;&amp; d.getElementById) x=d.getElementById(n); return x;
}
/* Functions that swaps images. */
function MM_swapImage() { //v3.0
    var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i&lt;(a.length-2);i+=3)
        if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src;
    x.src=a[i+2];}
}
function MM_swapImgRestore() { //v3.0
    var i,x,a=document.MM_sr; for(i=0;a&amp;&amp;i&lt;a.length&amp;&amp;(x=a[i])&amp;&amp;x.oSrc;i++) x.src=x.oSrc;
}

/* Functions that swaps down images. */
function MM_nbGroup(event, grpName) { //v6.0
var i,img,nbArr,args=MM_nbGroup.arguments;
    if (event == "init" &amp;&amp; args.length &gt; 2) {
        if ((img = MM_findObj(args[2])) != null &amp;&amp; !img.MM_init) {
            img.MM_init = true; img.MM_up = args[3]; img.MM_dn = img.src;
            if ((nbArr = document[grpName]) == null) nbArr = document[grpName] = new Array();
            nbArr[nbArr.length] = img;
            for (i=4; i &lt; args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = img.MM_dn = args[i+1];
                nbArr[nbArr.length] = img;
            }
        } else if (event == "over") {
            document.MM_nbOver = nbArr = new Array();
            for (i=1; i &lt; args.length-1; i+=3) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = (img.MM_dn &amp;&amp; args[i+2]) ? args[i+2] : ((args[i+1])?args[i+1] : img.MM_up);
                nbArr[nbArr.length] = img;
            }
        } else if (event == "out" ) {
            for (i=0; i &lt; document.MM_nbOver.length; i++) { img = document.MM_nbOver[i]; img.src =
(img.MM_dn) ? img.MM_dn : img.MM_up; }
        } else if (event == "down") {
            nbArr = document[grpName];
            if (nbArr) for (i=0; i &lt; nbArr.length; i++) { img=nbArr[i]; img.src = img.MM_up; img.MM_dn
= 0; }
            document[grpName] = nbArr = new Array();
            for (i=2; i &lt; args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = img.MM_dn = (args[i+1])? args[i+1] : img.MM_up;
                nbArr[nbArr.length] = img;
            }
        }
    }
/* Functions that handle preload. */
function MM_preloadImages() { //v3.0
var d=document; if(d.images){if(!d.MM_p) d.MM_p=new Array();
var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i&lt;a.length; i++)
    if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}
</pre>

```

```

}

//-->
</script>
<link href="../prufa.css" rel="stylesheet" type="text/css">
</head>
<body bgcolor="#ffffff"
onLoad="MM_preloadImages('../images/kerfisstjori_home/heim_f2.jpg','../images/kerfisstjori_home/heim_f4.jpg','../images/kerfisstjori_home/heim_f3.jpg','../images/kerfisstjori_home/innlogin_f2.jpg','../images/kerfisstjori_home/innlogin_f4.jpg','../images/kerfisstjori_home/innlogin_f3.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c8_f2.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c8_f4.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c8_f3.jpg','../images/kerfisstjori_home/vidta12_f2.jpg','../images/kerfisstjori_home/vidta12_f4.jpg','../images/kerfisstjori_home/vidta12_f3.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c14_f2.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c14_f4.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c14_f3.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c17_f2.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c17_f3.jpg','../images/kerfisstjori_home/kerfisstjori_home_r10_c6_f2.jpg','../images/kerfisstjori_home/kerfisstjori_home_r10_c10_f2.jpg');">
<!!--The following section is an HTML table which reassembles the sliced image in a browser.-->
<!--Copy the table section including the opening and closing table tags, and paste the data
where-->
<!--you want the reassembled image to appear in the destination document. -->
<!===== BEGIN COPYING THE HTML HERE =====>
<table width="800" border="0" align="center" cellpadding="0" cellspacing="0">
<!-- fwtable fwsrc="Kerfisstjori_home.png" fwbase="kerfisstjori_home.jpg"
fwstyle="Dreamweaver" fwdocid = "742308039" fnested="0" -->
  <tr>
<!-- Shim row, height 1. -->
    <td></td>
    <td></td>
  </tr>

```

```

<tr><!-- row 1 -->
<td colspan="22"></td>
<td></td>
</tr>
<tr><!-- row 2 -->
<td colspan="2"></td>
<td colspan="2"><a href="#" onMouseOut="MM_nbGroup('out');" onMouseOver="MM_nbGroup('over','heim','../images/kerfisstjori_home/heim_f2.jpg','../images/kerfisstjori_home/heim_f4.jpg',1);onClick="MM_nbGroup('down','navbar1','heim','../images/kerfisstjori_home/heim_f3.jpg',1);"></a></td>
<td></td>
<td><a href="kerfisstjori_staff.htm" onMouseOut="MM_nbGroup('out');" onMouseOver="MM_nbGroup('over','innlogin','../images/kerfisstjori_home/innlogin_f2.jpg','../images/kerfisstjori_home/innlogin_f4.jpg',1);onClick="MM_nbGroup('down','navbar1','innlogin','../images/kerfisstjori_home/innlogin_f3.jpg',1);"></a></td>
<td></td>
<td colspan="3"><a href="#" onMouseOut="MM_nbGroup('out');" onMouseOver="MM_nbGroup('over','kerfisstjori_home_r2_c8','../images/kerfisstjori_home/kerfisstjori_home_r2_c8_f2.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c8_f4.jpg',1);onClick="MM_nbGroup('down','navbar1','kerfisstjori_home_r2_c8','../images/kerfisstjori_home/kerfisstjori_home_r2_c8_f3.jpg',1);"></a></td>
<td></td>
<td><a href="#" onMouseOut="MM_nbGroup('out');" onMouseOver="MM_nbGroup('over','vidtal2','../images/kerfisstjori_home/vidtal2_f2.jpg','../images/kerfisstjori_home/vidtal2_f4.jpg',1);onClick="MM_nbGroup('down','navbar1','vidtal2','../images/kerfisstjori_home/vidtal2_f3.jpg',1);"></a></td>
<td></td>
<td colspan="2"><a href="#" onMouseOut="MM_nbGroup('out');" onMouseOver="MM_nbGroup('over','kerfisstjori_home_r2_c14','../images/kerfisstjori_home/kerfisstjori_home_r2_c14_f2.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c14_f4.jpg',1);onClick="MM_nbGroup('down','navbar1','kerfisstjori_home_r2_c14','../images/kerfisstjori_home/kerfisstjori_home_r2_c14_f3.jpg',1);"></a></td>
<td></td>
<td colspan="2"><a href="#" onMouseOut="MM_nbGroup('out');" onMouseOver="MM_nbGroup('over','kerfisstjori_home_r2_c17','../images/kerfisstjori_home/kerfisstjori_home_r2_c17_f2.jpg','../images/kerfisstjori_home/kerfisstjori_home_r2_c17_f4.jpg',1);onClick="MM_nbGroup('down','navbar1','kerfisstjori_home_r2_c17','../images/kerfisstjori_home/kerfisstjori_home_r2_c17_f3.jpg',1);"></a></td>
<td colspan="4"></td>
<td></td>
</tr>
<tr><!-- row 3 -->
<td colspan="22"></td>
<td></td>

```

```

</tr>
<tr><!-- row 4 -->
    <td rowspan="8"></td>
        <td colspan="18"></td>
            <td colspan="3"></td>
                <td></td>
            </tr>
<tr><!-- row 5 -->
    <td rowspan="4" colspan="2"></td>
        <td rowspan="3" colspan="14"></td>
            <td rowspan="3" colspan="2"></td>
                <td rowspan="7"></td>
                    <td></td>
                    <td rowspan="2"></td>
                        <td></td>
                    </tr>
<tr><!-- row 6 -->
    <td></td>
        <td></td>
    </tr>
<tr><!-- row 7 -->
    <td colspan="2" rowspan="5" bgcolor="#FFFFFF">&nbsp;</td>
        <td></td>
    </tr>
<tr><!-- row 8 -->
    <td colspan="16"></td>
        <td></td>
    </tr>
<tr><!-- row 9 -->
    <td colspan="18"></td>
        <td></td>
    </tr>
<tr><!-- row 10 -->
    <td colspan="4"></td>
        <td colspan="3"><a href="#" onMouseOut="MM_swapImgRestore();"
onMouseOver="MM_swapImage('kerfisstjori_home_r10_c6','','../images/kerfisstjori_home/kerfisstjori_home_r10_c6_f2.jpg',1);"></a></td>
            <td></td>
            <td colspan="5"><a href="#" onMouseOut="MM_swapImgRestore();"
onMouseOver="MM_swapImage('kerfisstjori_home_r10_c10','','../images/kerfisstjori_home/kerfisstjori_home_r10_c10_f2.jpg',1);"></a></td>
        </tr>

```

```
src="../images/kerfisstjori_home/kerfisstjori_home_r10_c10.jpg" width="133" height="25"
border="0" alt=""></a></td>
<td colspan="5"></td>
<td></td>
</tr>
<tr><!-- row 11 -->
<td colspan="18"></td>
<td></td>
</tr>
<tr><!-- row 12 -->
<td colspan="22" bgcolor="#FFFFFF">&nbsp;</td>
<td></td>
</tr>
<!-- This table was automatically created with Macromedia Fireworks -->
<!-- http://www.macromedia.com -->
</table>
<===== STOP COPYING THE HTML HERE =====>
<p align="center"><font size="1">Akureyri 2004 &copy; Yngvi Rafn Yngvason </font></p>
<map name="kerfisstjori_home_r1_c1Map">
  <area shape="rect" coords="4,1,178,57" href="http://jfactor_info.htm">
  <area shape="rect" coords="693,2,794,49" href="www.samhjalp.is">
</map>
</body>
</html>
```

kerfisstjori_staff.jsp

```

<html>
<head>
<title>Kerfisstj&ouml;ri - Starfsmenn</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <meta http-equiv=Content-Language content=is>
    <meta http-equiv=pragma content=no-cache>
    <meta http-equiv=Cache-Control content=must-revalidate>
    <meta http-equiv=Cache-Control content=no-cache>
    <meta http-equiv=Cache-Control content=no-store>
    <meta http-equiv=expires content=-1>

<script language="JavaScript">
function MM_findObj(n, d) { //v4.01
    var p,i,x; if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
        d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
    if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
    for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
    if(!x && d.getElementById) x=d.getElementById(n); return x;
}
/* Functions that swaps down images. */
function MM_nbGroup(event, grpName) { //v6.0
var i,img,nbArr,args=MM_nbGroup.arguments;
    if (event == "init" && args.length > 2) {
        if ((img = MM_findObj(args[2])) != null && !img.MM_init) {
            img.MM_init = true; img.MM_up = args[3]; img.MM_dn = img.src;
            if ((nbArr = document[grpName]) == null) nbArr = document[grpName] = new Array();
            nbArr[nbArr.length] = img;
            for (i=4; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = img.MM_dn = args[i+1];
                nbArr[nbArr.length] = img;
            }
        } else if (event == "over") {
            document.MM_nbOver = nbArr = new Array();
            for (i=1; i < args.length-1; i+=3) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = (img.MM_dn && args[i+2]) ? args[i+2] : ((args[i+1])?args[i+1] : img.MM_up);
                nbArr[nbArr.length] = img;
            }
        } else if (event == "out" ) {
            for (i=0; i < document.MM_nbOver.length; i++) { img = document.MM_nbOver[i]; img.src =
(img.MM_dn) ? img.MM_dn : img.MM_up; }
        } else if (event == "down" ) {
            nbArr = document[grpName];
            if (nbArr) for (i=0; i < nbArr.length; i++) { img=nbArr[i]; img.src = img.MM_up; img.MM_dn
= 0; }
            document[grpName] = nbArr = new Array();
            for (i=2; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = img.MM_dn = (args[i+1])? args[i+1] : img.MM_up;
                nbArr[nbArr.length] = img;
            }
        }
    }
/* Functions that handle preload. */
function MM_preloadImages() { //v3.0
    var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
        var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
        if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}
</script>

<link href="../prufa.css" rel="stylesheet" type="text/css">
<link href="../prufa.css" rel="stylesheet" type="text/css">
</head>
<body bgcolor="#E1E1E1">
onLoad="MM_preloadImages('../images/heim_f2.jpg','../images/heim_f4.jpg','../images/heim_f3.jp
g','../images/innlogg_f2.jpg','../images/innlogg_f4.jpg','../images/innlogg_f3.jpg','../im
ages/kerfisstjori_top_r3_c6_f2.jpg','../images/kerfisstjori_top_r3_c6_f4.jpg','../images/kerfisstj
ori_top_r3_c6_f3.jpg','../images/vidtal2_f2.jpg','../images/vidtal2_f4.jpg','../images/vidtal2_
f3.jpg','../images/kerfisstjori_top_r3_c10_f2.jpg','../images/kerfisstjori_top_r3_c10_f4.jpg'
,'../images/kerfisstjori_top_r3_c10_f3.jpg','../images/kerfisstjori_top_r3_c12_f2.jpg','../ima
ges/kerfisstjori_top_r3_c12_f4.jpg','../images/kerfisstjori_top_r3_c12_f3.jpg');">

```

```

<table width="800" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr>
    <!-- Shim row, height 1. -->
    <td></td>
    <td></td>
  </tr>
  <tr>
    <!-- row 1 -->
    <td colspan="13"></td>
    <td></td>
  </tr>
  <tr>
    <!-- row 2 -->
    <td rowspan="2"></td>
    <td colspan="12"></td>
    <td></td>
  </tr>
  <tr>
    <!-- row 3 -->
    <td><a href="kerfisstjori_home.htm" onMouseOut="MM_nbGroup('out');"
      onMouseOver="MM_nbGroup('over','heim','../images/heim_f2.jpg','../images/heim_f4.jpg',1);"
      onClick="MM_nbGroup('down','navbar1','heim','../images/heim_f3.jpg',1);"></a></td>
    <td></td>
    <td><a href="#" onMouseOut="MM_nbGroup('out');"
      onMouseOver="MM_nbGroup('over','innlogin','../images/innlogin_f2.jpg','../images/innlogin_f4.jpg'
      ,1);"
      onClick="MM_nbGroup('down','navbar1','innlogin','../images/innlogin_f3.jpg',1);"></a></td>
    <td></td>
    <td><a href="#" onMouseOut="MM_nbGroup('out');"
      onMouseOver="MM_nbGroup('over','kerfisstjori_top_r3_c6','../images/kerfisstjori_top_r3_c6_f2.j
      pg','../images/kerfisstjori_top_r3_c6_f4.jpg',1);"
      onClick="MM_nbGroup('down','navbar1','kerfisstjori_top_r3_c6','../images/kerfisstjori_top_r3_c
      6_f3.jpg',1);"></a></td>
    <td></td>
    <td><a href="#" onMouseOut="MM_nbGroup('out');"
      onMouseOver="MM_nbGroup('over','vidtal2','../images/vidtal2_f2.jpg','../images/vidtal2_f4.jpg'
      ,1);"
      onClick="MM_nbGroup('down','navbar1','vidtal2','../images/vidtal2_f3.jpg',1);"></a></td>
    <td></td>
    <td><a href="#" onMouseOut="MM_nbGroup('out');"
      onMouseOver="MM_nbGroup('over','kerfisstjori_top_r3_c10','../images/kerfisstjori_top_r3_c10_f2.j
      pg','../images/kerfisstjori_top_r3_c10_f4.jpg',1);"
      onClick="MM_nbGroup('down','navbar1','kerfisstjori_top_r3_c10','../images/kerfisstjori_top_r3_
      c10_f3.jpg',1);"></a></td>
    <td></td>
    <td><a href="#" onMouseOut="MM_nbGroup('out');"
      onMouseOver="MM_nbGroup('over','kerfisstjori_top_r3_c12','../images/kerfisstjori_top_r3_c12_f2.j
      pg','../images/kerfisstjori_top_r3_c12_f4.jpg',1);"
      onClick="MM_nbGroup('down','navbar1','kerfisstjori_top_r3_c12','../images/kerfisstjori_top_r3_
      c12_f3.jpg',1);"></a></td>
  
```

```

<td></td>
<td></td>
</tr>
<tr>
<td colspan="13" rowspan="3" valign="top" bgcolor="#FFFFFF"><font size="1" face="Verdana,
Arial, Helvetica, sans-serif"><br>
</font>
<table width="748" border="0" align="center" cellpadding="0" cellspacing="0">
<tr>
<td colspan="2"> <div align="right"><font size="1" face="Verdana, Arial, Helvetica,
sans-serif"></font></div>
<fieldset>
<div align="center">
<legend><font size="2" face="Verdana, Arial, Helvetica, sans-serif">Bæta
við starfsmanni</font></legend>
<form name="addStaff" method="post" action="../jsp/jspptestpage.jsp">
<table width="100%" border="0" align="left" cellpadding="0" cellspacing="0">
<tr>
<td width="9%"><div align="right"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">Kennitala:
</font></div></td>
<td width="11%"><input name="kennitala" type="text" id="kennitala"
size="11" maxlength="11" tabindex="1">
</td>
<td width="41%" valign="middle"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">Starfsmanna
n&uacute;mer:
<input name="staffid" type="text" id="staffid2" size="4" maxlength="4"
tabindex="2">
</font></td>
<td colspan="2"><div align="center"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">Karl
<input type="radio" name="radiobutton" value="radiobutton"
tabindex="10">
Kona
<input type="radio" name="radiobutton" value="radiobutton"
tabindex="11">
</font></div></td>
</tr>
<tr>
<td><div align="right"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">Nafn:</font></div></td>
<td colspan="2"><input name="firstname" type="text" id="firstname"
size="20" maxlength="20" tabindex="3">
<input name="lastname" type="text" id="lastname" size="30"
maxlength="30" tabindex="4">
<td width="17%"><div align="right"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">R&aacute; &eth; ningar
dags:</font></div></td>
<td width="22%"><input name="hire_date" type="text" id="hire_date"
size="14" maxlength="14" tabindex="12">
</tr>
<tr>
<td><div align="right"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">Heimili:</font></div></td>
<td colspan="2"><input name="address" type="text" id="address" size="35"
maxlength="35" tabindex="5">
<td><div align="right"><font size="1" face="Verdana, Arial, Helvetica,
sans-serif">Notendanafn:
</font></div></td>
<td><font size="1" face="Verdana, Arial, Helvetica, sans-serif">
<input name="username" type="text" id="username2" size="20"
maxlength="20" tabindex="13">
</font></td>
</tr>
<tr>
<td><div align="right"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">Sta&eth;ur:</font></div></td>
<td colspan="2"><input name="pnr" type="text" id="pnr" size="4"
maxlength="4" tabindex="6">
<input name="city" type="text" id="city" size="30" maxlength="30"
tabindex="7">
<td><div align="right"><font size="1" face="Verdana, Arial, Helvetica,
sans-serif">Lykil&eth;:
</font></div></td>
<td><font size="1" face="Verdana, Arial, Helvetica, sans-serif">

```

```

        <input name="password" type="password" id="password2" size="20"
maxlength="20" tabindex="14">
            </font></td>
        </tr>
        <tr>
            <td height="26"><div align="right"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">Starfsheiti:</font></div></td>
            <td colspan="2" > <div align="left"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">
                <select name="jobname" size="1" tabindex="8">
                    <option>--- Veldu starfsheiti ---</option>
                    <option>R&aacute;&eth;gjafi</option>
                    <option>Skrifstofustj&oacute;ri</option>
                    <option>Vaktma&eth;ur</option>
                    <option>A&eth;sto&eth;arma&eth;ur</option>
                    <option>Matr&aacute;eth;ur</option>
                    <option>Umsj&oacute;narma&eth;ur</option>
                    <option>Me&eth;fer&eth;arstj&oacute;ri</option>
                    <option>Sta&eth;arstj&oacute;ri</option>
                    <option>L&aacute;knir</option>
                    <option>Lyffr&aacute;eth;ingur</option>
                    <option>Gjaldkeri</option>
                    <option>Forst&ouml;eth;uma&eth;lpar</option>
                </select>
            </font></div></td>
            <td align="right"><font size="1" face="Verdana, Arial, Helvetica,
sans-serif">Sta&eth;festa
                lykilor&eth;:</font></div></td>
            <td><font size="1" face="Verdana, Arial, Helvetica, sans-serif">
                <input name="password2" type="password" id="password" size="20"
maxlength="20" tabindex="14">
            </font></td>
        </tr>
        <tr>
            <td align="left"><div align="right"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">Netfang:</font></div></td>
            <td colspan="2" > <div align="left"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">
                <input name="email" type="text" id="email" size="20" maxlength="20"
tabindex="9">
            </font></div></td>
            <td align="right"><font size="1" face="Verdana, Arial, Helvetica,
sans-serif">A&eth;gangs
                heimild:</font></div></td>
            <td><font size="1" face="Verdana, Arial, Helvetica, sans-serif">
                <select name="select" tabindex="15">
                    <option>- Veldu a&eth;gang -</option>
                    <option>R&aacute;&eth;gjafi</option>
                    <option>Vaktma&eth;ur</option>
                    <option>Umsj&oacute;narma&eth;ur</option>
                    <option>Yfirma&eth;ur</option>
                    <option>Kerfisstj&oacute;ri</option>
                </select>
            </font></td>
        </tr>
        <tr>
            <td align="left"><div align="right"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">S&iacute;mit&aacute;ki:</font></div></td>
            <td colspan="4" > <font size="1" face="Verdana, Arial, Helvetica, sans-
serif">
                <select name="phonetype" id="select5" tabindex="16">
                    <option>Heimas&iacute;mi</option>
                    <option>Vinnus&iacute;mi</option>
                    <option>GSM</option>
                    <option>Fax</option>
                </select>
                <font size="1" face="Verdana, Arial, Helvetica, sans-
serif">N&uacute;mer:</font>
                <input name="number" type="text" id="number3" tabindex="17">
                <input name="Add" type="submit" id="Add3" value="B&aacute;ta vi&eth;">
            </font></td>
        </tr>
    </table>
</form>
</div>
</fieldset></td>
</tr>

```

```

<tr>
    <td width="707"><div align="right"></div></td>
</tr>
<tr>
    <td width="707" valign="middle"><form name="addPhone" method="post"
action="../jsp/jsptestpage.jsp">
        <div align="right"><font size="1" face="Verdana, Arial, Helvetica, sans-serif">
            <input type="submit" name="Submit" value="Vista starfsmann">
        </font> <font size="1" face="Verdana, Arial, Helvetica, sans-serif">
            <input name="Reset" type="reset" id="Reset4" value="Hreinsa">
        </font> </div>
    </form></td>
    <td width="1">&ampnbsp</td>
</tr>
<tr>
    <td colspan="2" bgcolor="#CCCCCC"><font size="2" face="Verdana, Arial, Helvetica,
sans-serif">&ampnbsp</font><font size="2" face="Verdana, Arial, Helvetica, sans-
serif">Starfsmenn
        Samhj&aacute;lpar</font></td>
    <td width="40">&ampnbsp</td>
</tr>
<tr>
    <td height="25" colspan="2" bgcolor="#FFFFFF" dir="ltr" lang="is">&ampnbsp</td>
    <td>&ampnbsp</td>
</tr>
</table>

<div align="left"></div>
    <div align="left"><font size="1" face="Verdana, Arial, Helvetica, sans-
serif"></font></div></td>
    </tr>
    </table>
    <font size="1" face="Verdana, Arial, Helvetica, sans-serif">&ampnbsp</font></td>
    <td height="22">&ampnbsp</td>
</tr>
<tr>
    <td height="22">&ampnbsp</td>
</tr>
<tr>
    <td height="129" class="sidebar">&ampnbsp</td>
</tr>
<map name="m_kerfisstjori_top_r1_c1">
    <area shape="rect" coords="11,0,175,50" href="http://jfactor_info.htm" alt="Um Jfactor" >
    <area shape="rect" coords="704,6,792,46" href="http://samhjalp.is" alt="Heimasiða
Samhjálpar" >
</map>
</body>
</html>

```

login.htm

```

<html>
<head>
<title>login.jpg</title>
<meta http-equiv="Content-Type" content="text/html;">
<meta name="description" content="FW MX DW MX HTML">
<!-- Fireworks MX Dreamweaver MX target. Created Wed Apr 14 10:38:20 GMT+0000 (GMT Standard
Time) 2004-->
</head>
<body bgcolor="#ffffff">
<!--The following section is an HTML table which reassembles the sliced image in a browser.-->
<!--Copy the table section including the opening and closing table tags, and paste the data
where-->
<!--you want the reassembled image to appear in the destination document. -->
<===== BEGIN COPYING THE HTML HERE =====>
<p>&nbsp;</p><table width="394" border="0" align="center" cellpadding="0" cellspacing="0">
<!-- fwtable fwsr="Login.png" fwbase="login.jpg" fwstyle="Dreamweaver" fwdocid = "742308039"
fnested="0" -->
<tr>
<!-- Shim row, height 1. -->
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr><!-- row 1 -->
<td colspan="5"></td>
<td></td>
</tr>
<tr><!-- row 2 -->
<td rowspan="5"></td>
<td colspan="3"></td>
<td rowspan="5"></td>
<td></td>
</tr>
<tr><!-- row 3 -->
<td colspan="3"></td>
<td></td>
</tr>
<tr><!-- row 4 -->
<td rowspan="2"></td>
<td height="104"><div align="center">
<form name="form1" method="post" action="">
<div align="center"><font size="1" face="Verdana, Arial, Helvetica, sans-serif">Username:<br>
<input name="user" type="text" id="user" size="20" maxlength="20">
<br>
<font size="1" face="Verdana, Arial, Helvetica, sans-serif">Password:<br>
<input name="password" type="text" id="user3" size="20" maxlength="20">
<br>
<font size="1" face="Verdana, Arial, Helvetica, sans-serif">
<input type="submit" name="Submit" value="Innskr&aacute;u;"/>
</font> </font> </div>
</form>
<font size="1" face="Verdana, Arial, Helvetica, sans-serif"></font></div></td>
<td rowspan="2" height="108" border="0" alt=""></td>
<td></td>
</tr>
<tr><!-- row 5 -->
<td></td>
<td></td>
</tr>
<tr><!-- row 6 -->

```

Appendices for the Jfactor dissertation

Yngvi Rafn Yngvason

4/14/2004

```
<td height="2" colspan="3"></td>
<td></td>
</tr>
<!-- This table was automatically created with Macromedia Fireworks -->
<!-- http://www.macromedia.com -->
</table>
<!===== STOP COPYING THE HTML HERE =====>
</body>
</html>
```

jsptestpage.jsp

```

<%@ page contentType="text/html" %>
<html>
<%@ page import="is.unak.webapp.database.DatabaseConnection,
                is.unak.webapp.service.AccessoriesInterface,
                is.unak.webapp.service.AccessoriesServiceDesk,
                is.unak.webapp.service.PersonnelInterface,
                is.unak.webapp.service.PersonnelServiceDesk,
                java.sql.Connection,
                is.unak.webapp.activerecords.accessories.MailInfo,
                is.unak.webapp.activerecords.accessories.PostnrOps,
                is.unak.webapp.activerecords.staff.Staff,
                is.unak.webapp.presentation.KennitalaCheck"
%>

<head>
    <META http-equiv=Content-Language content=is>
    <META http-equiv=pragma content=no-cache>
    <META http-equiv=Cache-Control content=must-revalidate>
    <META http-equiv=Cache-Control content=no-cache>
    <META http-equiv=Cache-Control content=no-store>
    <META http-equiv=expires content=1>
<script language="JavaScript" type="text/JavaScript">
<!--
function MM_reloadImage(imgObj,src) { //v3.0
    var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
        var i,j=d.MM_p.length,a=MM_reloadImage.arguments; for(i=0; i<a.length; i++)
            if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}

function MM_findObj(n, d) { //v4.01
    var p,i,x; if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
        d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
    if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length; i++) x=d.forms[i][n];
    for(i=0;!x&&d.layers&&i<d.layers.length; i++) x=MM_findObj(n,d.layers[i].document);
    if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_nbGroup(event, grpName) { //v6.0
    var i,img,nbArr,args=MM_nbGroup.arguments;
    if (event == "init" && args.length > 2) {
        if ((img = MM_findObj(args[2])) != null && !img.MM_init) {
            img.MM_init = true; img.MM_up = args[3]; img.MM_dn = img.src;
            if ((nbArr = document[grpName]) == null) nbArr = document[grpName] = new Array();
            nbArr[nbArr.length] = img;
            for (i=4; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = img.MM_dn = args[i+1];
                nbArr[nbArr.length] = img;
            }
        } else if (event == "over") {
            document.MM_nbOver = nbArr = new Array();
            for (i=1; i < args.length-1; i+=3) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = (img.MM_dn && args[i+2]) ? args[i+2] : ((args[i+1])? args[i+1] : img.MM_up);
                nbArr[nbArr.length] = img;
            }
        } else if (event == "out" ) {
            for (i=0; i < document.MM_nbOver.length; i++) {
                img = document.MM_nbOver[i]; img.src = (img.MM_dn) ? img.MM_dn : img.MM_up; }
        } else if (event == "down") {
            nbArr = document[grpName];
            if (nbArr)
                for (i=0; i < nbArr.length; i++) { img=nbArr[i]; img.src = img.MM_up; img.MM_dn = 0; }
            document[grpName] = nbArr = new Array();
            for (i=2; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = img.MM_dn = (args[i+1])? args[i+1] : img.MM_up;
                nbArr[nbArr.length] = img;
            }
        }
    }

function MM_swapImgRestore() { //v3.0
    var i,x,a=document.MM_sr; for(i=0;a&&i<a.length&&(x=a[i])&&x.oSrc;i++) x.src=x.oSrc;
}

```

```

}

function MM_swapImage() { //v3.0
    var i,j=0,x,a=MM_swapImage.arguments; document.MM_sr=new Array; for(i=0;i<(a.length-2);i+=3)
        if ((x=MM_findObj(a[i]))!=null){document.MM_sr[j++]=x; if(!x.oSrc) x.oSrc=x.src;
    x.src=a[i+2];}
}
//-->
</script>
</head>
<body>
onLoad="MM_preloadImages('../images/testpage/heim_f3.gif','../images/testpage/heim_f2.gif','..
/images/testpage/heim_f4.gif','..../images/testpage/innlogin_f2.gif','..../images/testpage/testpage_
r3_c6_f3.gif','..../images/testpage/testpage_r3_c6_f2.gif','..../images/testpage/testpage_r3_c6_f
4.gif','..../images/testpage/vidtal2_f3.gif','..../images/testpage/vidtal2_f2.gif','..../images/test
page/vidtal2_f4.gif','..../images/testpage/testpage_r3_c10_f3.gif','..../images/testpage/testpage_
r3_c10_f2.gif','..../images/testpage/testpage_r3_c10_f4.gif','..../images/testpage/testpage_r3_c12
_f3.gif','..../images/testpage/testpage_r3_c12_f2.gif','..../images/testpage/testpage_r3_c12_f4.gi
f')"
<%
    String kennitala;
//    int staffId;
//    int sJobId;
    String firstName;
    String lastName;
    String address;
    int pnr;
    String city;
//    String pwdOne;
//    String pwdTwo;
    boolean pwdCheck = true;
    DatabaseConnection db = new DatabaseConnection();
    AccessoriesInterface ai = new AccessoriesServiceDesk();
    PersonnelInterface pi = new PersonnelServiceDesk();
    Connection conn = db.openConnection();
    KennitalaCheck kc = new KennitalaCheck();
    PostnrOps pops = new PostnrOps();
    Staff s = new Staff();
%>
<%
    kennitala = request.getParameter("kennitala");
    firstName = request.getParameter("firstName");
    lastName = request.getParameter("lastName");
    address = request.getParameter("address");
    pnr = Integer.parseInt(request.getParameter("pnr"));
    city = request.getParameter("city");
//    pwdOne = request.getParameter("pwdOne");
//    pwdTwo = request.getParameter("pwdTwo");
    s.setKennitala(kennitala);
    s.setFirstName(firstName);
    s.setLastName(lastName);
    s.setAddress(address);
    pi.addStaff(conn, s);

    pops.setPnr(pnr);
    pops.setCity(city);
    ai.addPnrAndCity(conn, pops);

    firstName = pi.getStaffName(conn, kennitala);
    pnr = ai.getPostnumber(conn, city);
    city = ai.getCityName(conn, pnr);
%>
<table border="0" cellpadding="0" cellspacing="0" width="800">
    <!-- fwtable fwsrc="Ümsjonarmadur_top.png" fwbase="testpage.gif" fwstyle="Dreamweaver"
fwdocid = "742308039" fwnested="0" -->
    <tr>
        <!-- Shim row, height 1. -->
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>
        <td></td>

```

```

<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
  <!-- row 1 -->
  <td colspan="14"></td>
  <td></td>
</tr>
<tr>
  <!-- row 2 -->
  <td rowspan="2"></td>
  <td colspan="13"></td>
  <td></td>
</tr>
<tr>
  <!-- row 3 -->
  <td><a href="../html/jspptestpage.htm" target="_top" onClick="MM_nbGroup('down','navbar1','heim','../images/testpage/heim_f3.gif',1);"
onMouseOver="MM_nbGroup('over','heim','../images/testpage/heim_f2.gif','../images/testpage/heim_f4.gif',1);"
onMouseOut="MM_nbGroup('out');"></a></td>
  <td></td>
  <td><a href="#" onMouseOut="MM_swapImgRestore()" onMouseOver="MM_swapImage('innlogin','','../images/testpage/innlogin_f2.gif',1);"></a></td>
  <td></td>
  <td><a href="javascript:;" target="_top" onClick="MM_nbGroup('down','navbar1','testpage_r3_c6','../images/testpage/testpage_r3_c6_f3.gif',1);"
onMouseOver="MM_nbGroup('over','testpage_r3_c6','../images/testpage/testpage_r3_c6_f2.gif','../images/testpage/testpage_r3_c6_f4.gif',1);"
onMouseOut="MM_nbGroup('out');"></a></td>
  <td></td>
  <td><a href="javascript:;" target="_top" onClick="MM_nbGroup('down','navbar1','vidtal2','../images/testpage/vidtal2_f3.gif',1);"
onMouseOver="MM_nbGroup('over','vidtal2','../images/testpage/vidtal2_f2.gif','../images/testpage/vidtal2_f4.gif',1);"
onMouseOut="MM_nbGroup('out');"></a></td>
  <td></td>
  <td><a href="javascript:;" target="_top" onClick="MM_nbGroup('down','navbar1','testpage_r3_c10','../images/testpage/testpage_r3_c10_f3.gif',1);"
onMouseOver="MM_nbGroup('over','testpage_r3_c10','../images/testpage/testpage_r3_c10_f2.gif','../images/testpage/testpage_r3_c10_f4.gif',1);"
onMouseOut="MM_nbGroup('out');"></a></td>
  <td></td>
  <td><a href="javascript:;" target="_top" onClick="MM_nbGroup('down','navbar1','testpage_r3_c12','../images/testpage/testpage_r3_c12_f3.gif',1);"
onMouseOver="MM_nbGroup('over','testpage_r3_c12','../images/testpage/testpage_r3_c12_f2.gif','../images/testpage/testpage_r3_c12_f4.gif',1);"
onMouseOut="MM_nbGroup('out');"></a></td>
  <td colspan="2"></td>
  <td></td>
</tr>
<!-- This table was automatically created with Macromedia Fireworks -->
<!-- http://www.macromedia.com -->

```

```

</table>
<map name="testpage_r1_c1Map">
  <area shape="rect" coords="690,3,797,49" href="http://www.samhjalp.is" alt="Go to Samhj&aacute;lp's main web site.">
  <area shape="rect" coords="2,1,182,51" href="http://html/jfactor_info.htm">
</map>
<p><font size="2" face="Verdana, Arial, Helvetica, sans-serif">This is a database test to prove that the JSP <br> is working on top of the domain layer in the Jfactor system.</font></p>
<p><font size="2" face="Verdana, Arial, Helvetica, sans-serif">Information displayed is the <b>first name</b>, <b>postnumber</b> and <b>city</b>.</font></p>
<font face="Verdana, Arial, Helvetica, sans-serif">
<p><font size="2">And the result is:</font></p>

</font>
<font face="Verdana, Arial, Helvetica, sans-serif">
<table width="542" border="0" cellspacing="0" cellpadding="0">
  <tr bgcolor="#FEDDAB">
    <td width="158"><b><i><font size="2"> First name </font></i></b></td>
    <td width="103"><b><i><font size="2"> Postn<font face="Verdana, Arial, Helvetica, sans-serif">r.</font></i></b></td>
    <td width="281"><b><i><font size="2"> City </font></i></b></td>
  </tr>
  <tr>
    <td><p><font size="2"> <%=firstName%> </font></p></td>
    <td><font size="2"><%=pnr%></font></td>
    <td><font size="2"><%=city%> </font></td>
  </tr>
</table>
</font>
<p>&nbsp;</p>
<p><font size="1" face="Verdana, Arial, Helvetica, sans-serif">To see other already created please check out the following links:</font></p>
<p><font size="1" face="Verdana, Arial, Helvetica, sans-serif"><a href="http://html/login.htm">Login page</a> - <a href="http://html/kerfisstjori_home.htm">System Administrator homepage</a> - <a href="http://html/kerfisstjori_staff.htm">System Administrator staffpage</a> - <a href="http://html/jsptestpage.htm">JSP test page</a> </font></p>
</body>
</html>

```

kerfisstjori_staff.jsp

```

<%@ page contentType="text/html" %>
<html>
<%@ page import="is.unak.webapp.database.DatabaseConnection,
                 is.unak.webapp.service.AccessoriesInterface,
                 is.unak.webapp.service.AccessoriesServiceDesk,
                 is.unak.webapp.service.PersonnelInterface,
                 is.unak.webapp.service.PersonnelServiceDesk,
                 java.sql.Connection,
                 is.unak.webapp.activerecords.accessories.MailInfo,
                 is.unak.webapp.activerecords.accessories.PostnrOps,
                 is.unak.webapp.activerecords.staff.Staff,
                 is.unak.webapp.presentation.KennitalaCheck"
                 %>
<head>
<title>Kerfisstj&ouml;ri - Starfsmenn</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta http-equiv=Content-Language content=is>
<meta http-equiv=pragma content=no-cache>
<meta http-equiv=Cache-Control content=must-revalidate>
<meta http-equiv=Cache-Control content=no-cache>
<meta http-equiv=Cache-Control content=no-store>
<meta http-equiv=expires content=-1>
<!-- Fireworks MX Dreamweaver MX target. Created Thu Feb 26 21:57:07 GMT+0000 (GMT Standard
Time) 2004-->
<script language="JavaScript">
<!--
/* Functions that swaps down images. */

/* Functions that handle preload. */
function MM_preloadImages() { //v3.0
    var d=document; if(d.images){ if(!d.MM_p) d.MM_p=new Array();
        var i,j=d.MM_p.length,a=MM_preloadImages.arguments; for(i=0; i<a.length; i++)
        if (a[i].indexOf("#")!=0){ d.MM_p[j]=new Image; d.MM_p[j++].src=a[i];}}
}

function MM_findObj(n, d) { //v4.01
    var p,i,x; if(!d) d=document; if((p=n.indexOf("?"))>0&&parent.frames.length) {
        d=parent.frames[n.substring(p+1)].document; n=n.substring(0,p);}
    if(!(x=d[n])&&d.all) x=d.all[n]; for (i=0;!x&&i<d.forms.length;i++) x=d.forms[i][n];
    for(i=0;!x&&d.layers&&i<d.layers.length;i++) x=MM_findObj(n,d.layers[i].document);
    if(!x && d.getElementById) x=d.getElementById(n); return x;
}

function MM_nbGroup(event, grpName) { //v6.0
    var i,img,nbArr,args=MM_nbGroup.arguments;
    if (event == "init" && args.length > 2) {
        if ((img = MM_findObj(args[2])) != null && !img.MM_init) {
            img.MM_init = true; img.MM_up = args[3]; img.MM_dn = img.src;
            if ((nbArr = document[grpName]) == null) nbArr = document[grpName] = new Array();
            nbArr[nbArr.length] = img;
            for (i=4; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = img.MM_dn = args[i+1];
                nbArr[nbArr.length] = img;
            }
        }
    } else if (event == "over") {
        document.MM_nbOver = nbArr = new Array();
        for (i=1; i < args.length-1; i+=3) if ((img = MM_findObj(args[i])) != null) {
            if (!img.MM_up) img.MM_up = img.src;
            img.src = (img.MM_dn && args[i+2]) ? args[i+2] : ((args[i+1])? args[i+1] : img.MM_up);
            nbArr[nbArr.length] = img;
        }
    } else if (event == "out" ) {
        for (i=0; i < document.MM_nbOver.length; i++) {
            img = document.MM_nbOver[i]; img.src = (img.MM_dn) ? img.MM_dn : img.MM_up; }
        else if (event == "down") {
            nbArr = document[grpName];
            if (nbArr)
                for (i=0; i < nbArr.length; i++) { img=nbArr[i]; img.src = img.MM_up; img.MM_dn = 0; }
            document[grpName] = nbArr = new Array();
            for (i=2; i < args.length-1; i+=2) if ((img = MM_findObj(args[i])) != null) {
                if (!img.MM_up) img.MM_up = img.src;
                img.src = img.MM_dn = (args[i+1])? args[i+1] : img.MM_up;
            }
        }
    }
}

```

```

        nbArr[nbArr.length] = img;
    }
}
//-->
</script>
<link href="../prufa.css" rel="stylesheet" type="text/css">

</head>
<body bgcolor="#FFFFCC">
onLoad="MM_preloadImages('../images/heim_f3.jpg','../images/heim_f2.jpg','../images/heim_f4.jp
g','../images/innlogn_f3.jpg','../images/innlogn_f2.jpg','../images/innlogn_f4.jpg','../images
/kerfisstjori_top_r3_c6_f3.jpg','../images/kerfisstjori_top_r3_c6_f2.jpg','../images/kerfisstj
ori_top_r3_c6_f4.jpg','../images/vidtal2_f3.jpg','../images/vidtal2_f2.jpg','../images/vidtal2
_f4.jpg','../images/kerfisstjori_top_r3_c10_f3.jpg','../images/kerfisstjori_top_r3_c10_f2.jpg'
,'../images/kerfisstjori_top_r3_c10_f4.jpg','../images/kerfisstjori_top_r3_c12_f3.jpg','../ima
ges/kerfisstjori_top_r3_c12_f2.jpg','../images/kerfisstjori_top_r3_c12_f4.jpg')">
<%
    String kennitala;
//    int staffId;
//    int sJobId;
    String firstName;
    String lastName;
    String address;
    int pnr;
    String city;
//    String pwdOne;
//    String pwdTwo;
    boolean pwdCheck = true;
    DatabaseConnection db = new DatabaseConnection();
    AccessoriesInterface ai = new AccessoriesServiceDesk();
    PersonnelInterface pi = new PersonnelServiceDesk();
    Connection conn = db.openConnection();
    KennitalaCheck kc = new KennitalaCheck();
    PostnrOps pops = new PostnrOps();
    Staff s = new Staff();
%>
<%
    kennitala = request.getParameter("kennitala");
    firstName = request.getParameter("firstName");
    lastName = request.getParameter("lastName");
    address = request.getParameter("address");
    pnr = Integer.parseInt(request.getParameter("pnr"));
    city = request.getParameter("city");
//    pwdOne = request.getParameter("pwdOne");
//    pwdTwo = request.getParameter("pwTwo");
    s.setKennitala(kennitala);
    s.setFirstName(firstName);
    s.setLastName(lastName);
    s.setAddress(address);
    pi.addStaff(conn, s);

    pops.setPnr(pnr);
    pops.setCity(city);
    ai.addPnrAndCity(conn, pops);

    firstName = pi.getStaffName(conn, kennitala);
    pnr = ai.getPostnumber(conn, city);
    city = ai.getCityName(conn, pnr);
%>

<table width="800" border="0" align="center" cellpadding="0" cellspacing="0">
<tr>
    <!-- Shim row, height 1. -->
    <td></td>
    <td></td>

```

```

</tr>
<tr>
  <!-- row 1 -->
  <td colspan="13"></td>
    <td></td>
  </tr>
<tr>
  <!-- row 2 -->
  <td rowspan="2"></td>
    <td colspan="12"></td>
      <td></td>
  </tr>
<tr>
  <!-- row 3 -->
  <td><a href="javascript:;" target="_top"
onClick="MM_nbGroup('down','navbar1','heim','../images/heim_f3.jpg',1);"
onMouseOver="MM_nbGroup('over','heim','../images/heim_f2.jpg','../images/heim_f4.jpg',1);"
onMouseOut="MM_nbGroup('out');"></a></td>
    <td></td>
      <td><a href="javascript:;" target="_top"
onClick="MM_nbGroup('down','navbar1','innlogin','../images/innlogin_f3.jpg',1);"
onMouseOver="MM_nbGroup('over','innlogin','../images/innlogin_f2.jpg','../images/innlogin_f4.jpg'
,1);" onMouseOut="MM_nbGroup('out');"></a></td>
        <td></td>
          <td><a href="javascript:;" target="_top"
onClick="MM_nbGroup('down','navbar1','kerfisstjori_top_r3_c6','../images/kerfisstjori_top_r3_c
6_f3.jpg',1);"
onMouseOver="MM_nbGroup('over','kerfisstjori_top_r3_c6','../images/kerfisstjori_top_r3_c6_f2.j
pg','../images/kerfisstjori_top_r3_c6_f4.jpg',1);" onMouseOut="MM_nbGroup('out');"></a></td>
            <td></td>
              <td><a href="javascript:;" target="_top"
onClick="MM_nbGroup('down','navbar1','vidtal2','../images/vidtal2_f3.jpg',1);"
onMouseOver="MM_nbGroup('over','vidtal2','../images/vidtal2_f2.jpg','../images/vidtal2_f4.jpg'
,1);" onMouseOut="MM_nbGroup('out');"></a></td>
                <td></td>
                  <td><a href="javascript:;" target="_top"
onClick="MM_nbGroup('down','navbar1','kerfisstjori_top_r3_c10','../images/kerfisstjori_top_r3_
c10_f3.jpg',1);"
onMouseOver="MM_nbGroup('over','kerfisstjori_top_r3_c10','../images/kerfisstjori_top_r3_c10_f2
.jpg','../images/kerfisstjori_top_r3_c10_f4.jpg',1);" onMouseOut="MM_nbGroup('out');"></a></td>
                    <td></td>
                      <td><a href="javascript:;" target="_top"
onClick="MM_nbGroup('down','navbar1','kerfisstjori_top_r3_c12','../images/kerfisstjori_top_r3_
c12_f3.jpg',1);"
onMouseOver="MM_nbGroup('over','kerfisstjori_top_r3_c12','../images/kerfisstjori_top_r3_c12_f2
.jpg','../images/kerfisstjori_top_r3_c12_f4.jpg',1);" onMouseOut="MM_nbGroup('out');"></a></td>
                        <td></td>
                          <td></td>
</tr>
<tr>
  <td colspan="13" rowspan="3" valign="top" bgcolor="#FFFFFF"><font size="1" face="Verdana,
Arial, Helvetica, sans-serif"><br>
    </font> <table width="748" border="0" align="center" cellpadding="0" cellspacing="0">
      <tr>
        <td colspan="2"> <div align="right"><font size="1" face="Verdana, Arial, Helvetica,
sans-serif"></font></div>
          <fieldset>
            <div align="center">

```

```

<legend><font size="2" face="Verdana, Arial, Helvetica, sans-serif">Bæta  

við starfsmanni</font></legend>
<form name="addStaff" method="post" action="/jsp/kerfisstjori_staff.jsp">
    <table width="100%" border="0" align="left" cellpadding="0" cellspacing="0">
        <tr>
            <td width="9%"><div align="right"><font size="1" face="Verdana, Arial,  

Helvetica, sans-serif">Kennitala:</font></div></td>
            <td width="11%"><input name="kennitala" type="text" id="kennitala"  

size="11" maxlength="11" tabindex="1">
            </td>
            <td width="41%" valign="middle"><font size="1" face="Verdana, Arial,  

Helvetica, sans-serif">Starfsmanna  

n&uacute;mer:  

<input name="staffid" type="text" id="staffid2" size="4" maxlength="4"  

tabindex="2">
            </font></td>
            <td colspan="2"><div align="center"><font size="1" face="Verdana, Arial,  

Helvetica, sans-serif">Karl  

<input type="radio" name="radiobutton" value="radiobutton"  

tabindex="10">
                Kona  

<input type="radio" name="radiobutton" value="radiobutton"  

tabindex="11">
            </font></div></td>
        </tr>
        <tr>
            <td align="right"><font size="1" face="Verdana, Arial,  

Helvetica, sans-serif">Nafn:</font></td>
            <td colspan="2"><input name="firstname" type="text" id="firstname"  

size="20" maxlength="20" tabindex="3">
                <input name="lastname" type="text" id="lastname" size="30"  

maxlength="30" tabindex="4">
            <td width="17%"><div align="right"><font size="1" face="Verdana, Arial,  

Helvetica, sans-serif">R&aacute; &eth; ningar  

dags: </font></div></td>
            <td width="22%"><input name="hire_date" type="text" id="hire_date"  

size="14" maxlength="14" tabindex="12">
            </td>
        </tr>
        <tr>
            <td align="right"><font size="1" face="Verdana, Arial,  

Helvetica, sans-serif">Heimili:</font></td>
            <td colspan="2"><input name="address" type="text" id="address" size="35"  

maxlength="35" tabindex="5">
            <td align="right"><font size="1" face="Verdana, Arial, Helvetica,  

sans-serif">Notendanafn:  

</font></div></td>
            <td><font size="1" face="Verdana, Arial, Helvetica, sans-serif">  

<input name="username" type="text" id="username2" size="20"  

maxlength="20" tabindex="13">
            </font></td>
            </tr>
            <tr>
                <td align="right"><font size="1" face="Verdana, Arial,  

Helvetica, sans-serif">Sta&eth;ur:</font></td>
                <td colspan="2"><input name="pnr" type="text" id="pnr" size="4"  

maxlength="4" tabindex="6">
                    <input name="city" type="text" id="city" size="30" maxlength="30"  

tabindex="7">
                <td align="right"><font size="1" face="Verdana, Arial, Helvetica,  

sans-serif">Lykilor&eth;:  

</font></div></td>
                <td><font size="1" face="Verdana, Arial, Helvetica, sans-serif">  

<input name="password" type="password" id="password2" size="20"  

maxlength="20" tabindex="14">
                </font></td>
                </tr>
                <tr>
                    <td height="26"><div align="right"><font size="1" face="Verdana, Arial,  

Helvetica, sans-serif">Starfsheiti:</font></div></td>
                    <td colspan="2"> <div align="left"><font size="1" face="Verdana, Arial,  

Helvetica, sans-serif">
                        <select name="jobname" size="1" tabindex="8">
                            <option>--- Veldu starfsheiti ---</option>
                            <option>R&aacute; &eth; gjafi</option>
                            <option>Skrifstofustj&oacute;ri</option>
                            <option>Vaktma&eth;ur</option>

```

```

<option>A&eth;sto&eth;arma&eth;ur</option>
<option>Matr&aacute;&eth;ur</option>
<option>Umsj&oacute;narma&eth;ur</option>
<option>Me&eth;fer&eth;arstj&oacute;ri</option>
<option>Sta&eth;arstj&oacute;ri</option>
<option>L&aacute;knir</option>
<option>Lyffr&aacute;g&eth;ingur</option>
<option>Gjaldkeri</option>
<option>Forst&ouml;&eth;uma&eth;ur Samhj&aacute;lpar</option>
</select>
</font></div></td>
<td> <div align="right"><font size="1" face="Verdana, Arial, Helvetica,
sans-serif">Sta&eth;festa
lykilor&eth;:</font></div></td>
<td><font size="1" face="Verdana, Arial, Helvetica, sans-serif">
<input name="password2" type="password" id="password" size="20"
maxlength="20" tabindex="14">
</font></td>
</tr>
<tr>
<td align="left"><div align="right"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">Netfang:</font></div></td>
<td colspan="2"><div align="left"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">
<input name="email" type="text" id="email" size="20" maxlength="20"
tabindex="9">
</font></div></td>
<td align="right"><font size="1" face="Verdana, Arial, Helvetica,
sans-serif">A&eth;gangs
heimild:</font></div></td>
<td><font size="1" face="Verdana, Arial, Helvetica, sans-serif">
<select name="select" tabindex="15">
<option>- Veldu a&eth;gang -</option>
<option>R&aacute; &eth;gjafi</option>
<option>Vaktma&eth;ur</option>
<option>Umsj&oacute;narma&eth;ur</option>
<option>Yfirma&eth;ur</option>
<option>Kerfisstj&oacute;ri</option>
</select>
</font></td>
</tr>
<tr>
<td align="left"><div align="right"><font size="1" face="Verdana, Arial,
Helvetica, sans-serif">S&iacute;mit&aacute;ki:</font></div></td>
<td colspan="4"><font size="1" face="Verdana, Arial, Helvetica, sans-
serif">
<select name="phonetype" id="select5" tabindex="16">
<option>Heimas&iacute;mi</option>
<option>Vinnus&iacute;mi</option>
<option>GSM</option>
<option>Fax</option>
</select>
<font size="1" face="Verdana, Arial, Helvetica, sans-
serif">N&uacute;mero:</font>
<input name="number" type="text" id="number3" tabindex="17">
<input name="Add" type="submit" id="Add3" value="B&aacute;ta vi&eth;:">
</font></td>
</tr>
</table>
</form>
</div>
</fieldset></td>
</tr>
<tr>
<td width="707"><div align="right"></div></td>
</tr>
<tr>
<td width="707" valign="middle"><form name="addPhone" method="post"
action="/jsp/kerfisstjori_staff.jsp">
<div align="right"><font size="1" face="Verdana, Arial, Helvetica, sans-serif">
<input type="submit" name="Submit" value="Vista starfsmann">
</font> <font size="1" face="Verdana, Arial, Helvetica, sans-serif">
<input name="Reset" type="reset" id="Reset4" value="Hreinsa">
</font> </div>
</form></td>
<td width="1">&nbsp;</td>
</tr>

```

```
<tr>
    <td colspan="2" bgcolor="#CCCCCC"><font size="2" face="Verdana, Arial, Helvetica,
    sans-serif">&nbsp;</font><font size="2" face="Verdana, Arial, Helvetica, sans-
    serif">Starfsmenn
        Samhj&aacute;lpar</font></td>
        <td width="40">&nbsp;</td>
    </tr>
    <tr>
        <td height="25" colspan="2" bgcolor="#FFFFFF" dir="ltr" lang="is">This
            is a database test to prove that the JSP is working on top of the
            domain layer in the Jfactor system.<br>
            And the result is: <%=firstName%> <br> <%=pnر%> <%=city%> </td>
        <td>&nbsp;</td>
    </tr>
</table>
<div align="left"></div>
<div align="left"><font size="1" face="Verdana, Arial, Helvetica, sans-
    serif"></font></div></td>
</tr>
</table>
<map name="m_kerfisstjori_top_r1_c1">
    <area shape="rect" coords="11,0,175,50" href="http://jfactor.info.htm" alt="Um Jfactor" >
    <area shape="rect" coords="704,6,792,46" href="http://samhjalp.is" alt="Heimasiða
Samhjálpar" >
</map>
</body>
</html>
```

Digital copy of code

Appendices for the Jfactor dissertation

Yngvi Rafn Yngvason

4/14/2004