News System
EDA031 Project

Erik Stenlund, Alexander Karlsson, Kit Gustavsson, Erik Nimmermark

11 april 2015

1 System Design

1.1 Description and responsibilities of classes

1.1.1 Classes used by server

`BestServer`
This program has the main method for the primary memory server and is the server that use primary memory to store the newsgroups and articles. It uses a `Server` object to start the server and communicate with connected clients using a `MessageHandler`. The server reads the messages that follows the given protocol and handles the commands, given by `Protocol` that is received from the client. For storing of the data the `BestServer` contains an object of the class `MemoryDatabase`.

`DiscServer`
This program has the main method for the hard drive server and is a server that stores newsgroups and articles on the hard drive. It uses a `Server` object to start the server and then handles the communication with the connected clients using a `MessageHandler`. It works the same way as the `BestServer` but uses a `DiscDatabase` instead of the `MemoryDatabase` for storage of the data.

`Database`
This is the superclass of the `DiscDatabase` and the `MemoryDatabase` and it contains virtual members that handle adding, removing and accessing newsgroups and articles. It also contains a private attribute that decides which the next newsgroup ID(?) to be used is. The header file also contains structs that represents both the articles and the newsgroups that are used throughout the server programs. The newsgroup struct contains a map which contains all the articles in that specific newsgroup, the map uses the article id as key.

`DiscDatabase`
The `DiscDatabase` class extends the `Database` class and uses the hard drive for storing the newsgroups. At startup it check if the folder named root exists in the directory of the program. If not i creates it, the root folder is used to store the database on the hard drive. If the root folder exists the program creates newsgroup structs for each subfolder in root and article structs for all files in the subfolders. To decide which id a new newsgroup or article should have each folder contains a file called nextID which contains the next id for the article if it is in a subfolder or the next id for the newsgroup if its in the root folder. The class also has memberfunctions for adding, removing and accessing articles or newsgroups from the database.

`MemoryDatabase`
This class also extends the `Database` class and uses the memory for storing the newsgroups and articles which means that they won't be saved when closing the server. The class uses a map for storing the newsgroups structs using the id as key. It has members for accessing, inserting and removing newsgroups from the list and for accessing, inserting and removing articles from a newsgroup.

1.1.2 Classes used by client

`Client`
This is the class with the main method on the client side. The main sets up the connection and listens to user input. Depending on what command, from `Protocol`, the user typed it send the correct command (according to the protocol) to carry out what the user intended to do or prints an error message if it cant be done for some reason. All the communication between the client and the server is sent using the `MessageHandler` class.

1.1.3 Classes used by both server and client

`Protocol`
Class with all the enums for the different flags a message can contain. This is used to get rid of magic numbers in the code and make it easier to understand.

`Messagehandler`
The class that is responsible for sending the data between the server and the client. The methods contains the logic for receiving and sending bytes in different ways depending on that kind of message that is to be sent/received. It uses the `Connection` class to actually send and receive bytes of data.

`Connection`
The class that represents the connection with socket and address. This is that class that writes and reads data. The methods for sending and receiving data is used by the `MessageHandler` class to actually send and receive data between the client and server.

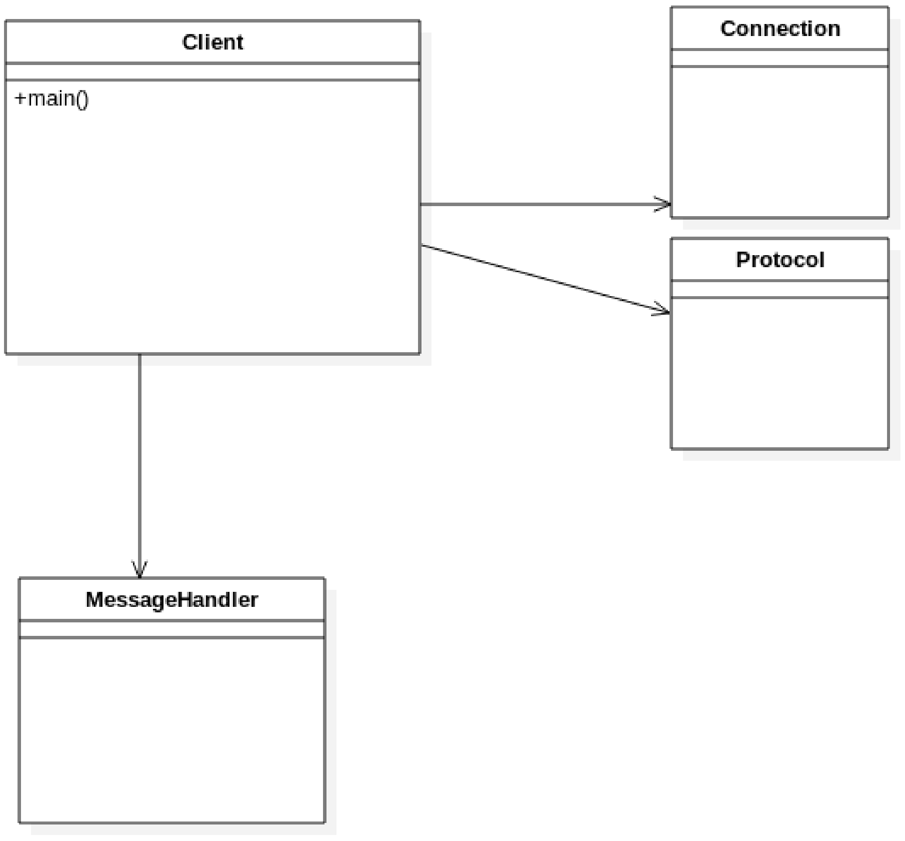
2 Conclusion

Our programs do fulfill all the given requriements from the specification. The client both reads and present input and server replies from and to the terminal. The client program command help gives all information about how to use the program and all errors are handled and prints an error message to the clients terminal window.

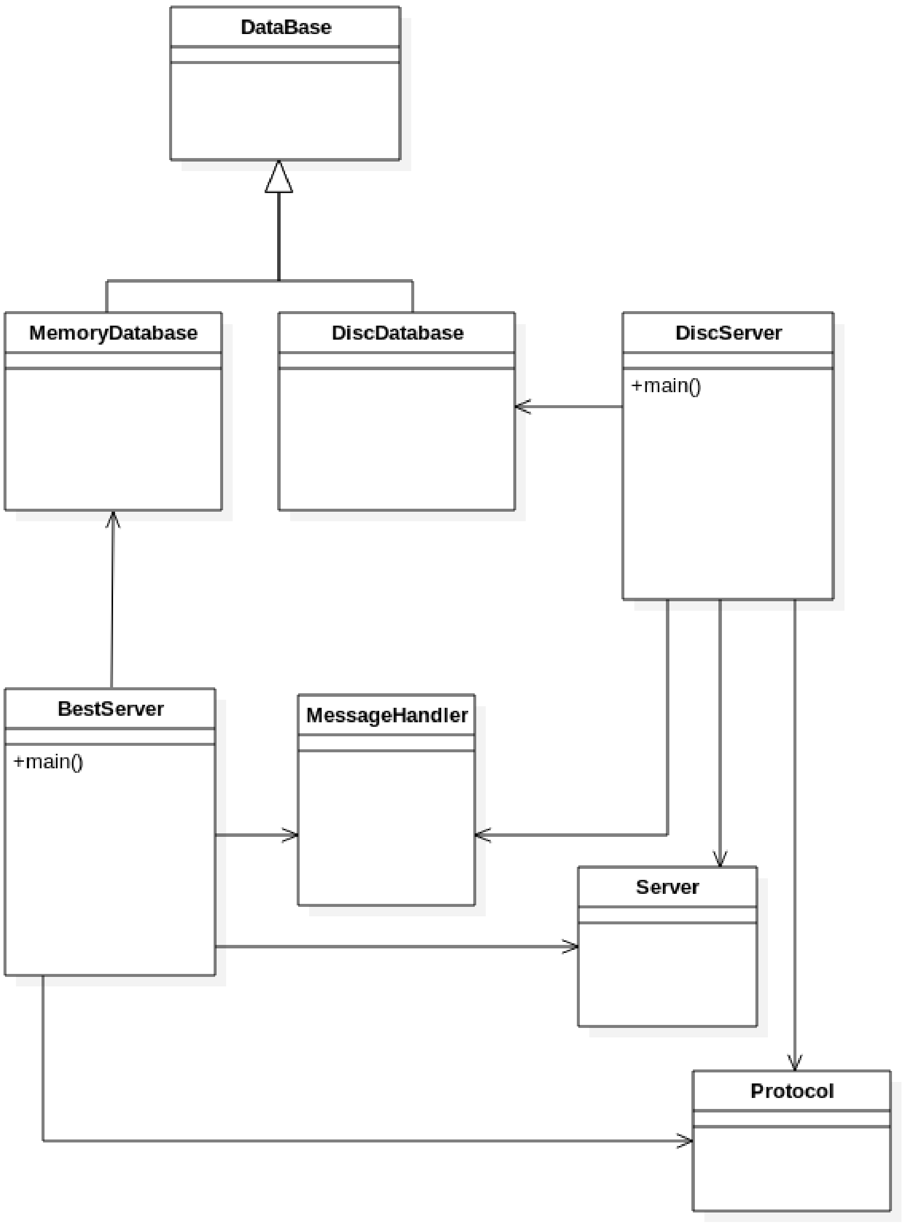
We have got two different server version, one using the primary memory and one using the hard drive. Both our Databases guarantee that each newsgroup has an unique id number and that the article id are unique for each newsgroup, this is done using files containing the next id descriped earlier. The newsgroups and articles are ordered using the id number which is incremented for each new newsgroup and article. This makes the articles and newsgroups listed in chronological order. The servers also handles as many (as long as there is memory) newsgroups and articles as possible with as long titles as possible.

One problem with our DiscDatabase is that it reads all information about the database from the hard drive once each startup. That means that removing needed files from the hard drive during execution of the server won't be seen by clients connected to the server. We see this actions as incorrect use of the server and the one editing the dataebase files should be held responsible in that case.

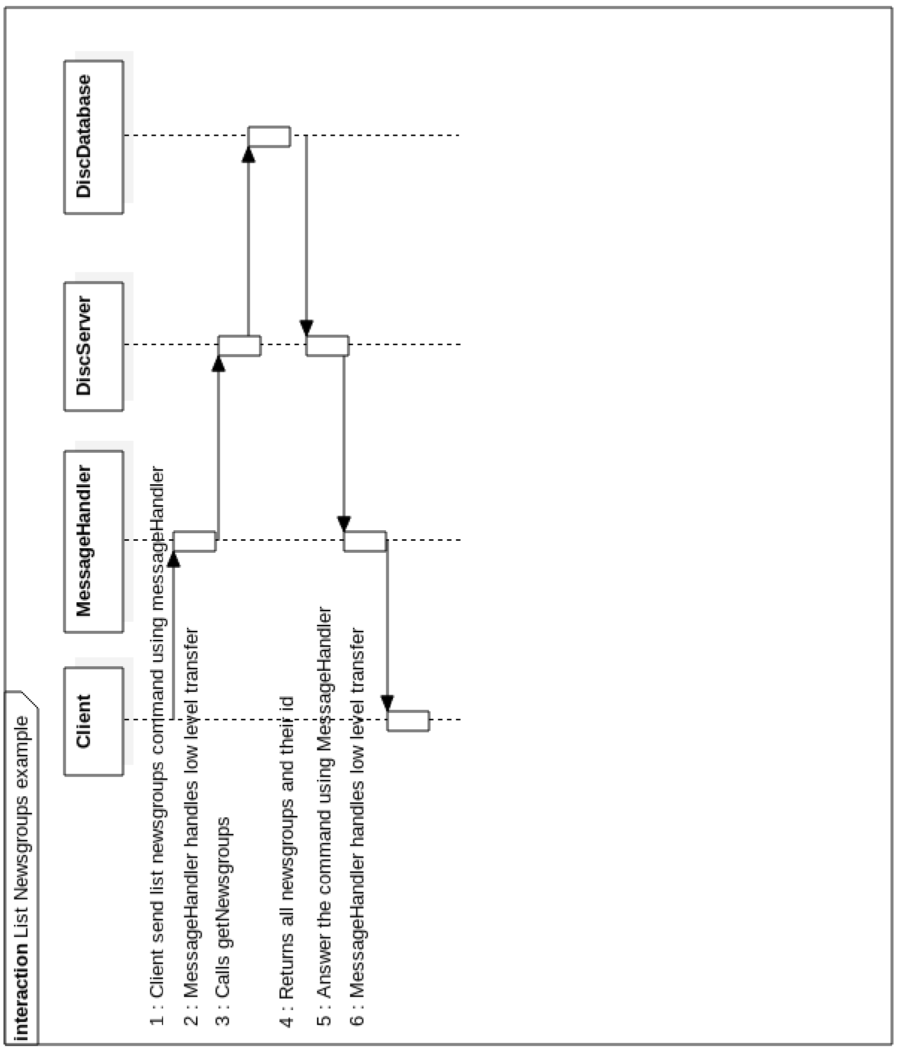
3 UML and Sequence diagrams



Figur 1: UML for Client program



Figur 2: UML for Server programs


interaction List Newsgroups example
Client
MessageHandler
DiscServer
DiscDatabase
1 : Client send list newsgroups command using messageHandler
2 : MessageHandler handles low level transfer
3 : Calls getNewsgroups
4 : Returns all newsgroups and their id
5 : Answer the command using MessageHandler
6 : MessageHandler handles low level transfer

Figur 3: Sequence diagram for List Newsgroups command