**Assignment 1, Hangman**

**Network Programming, ID1212**

Peter Svensson, peters3@kth.se

2018-11-10

**1 Introduction**

For the first assignment we were given the task of developing a distributed application in Java. This

application is supposed to handle interaction between the client, which is the player, and the server,

where the actual game is running, via socket communication and blocking threads.

I choose to implement the game “Hangman” for my distributed application. This task came with

some requirements, which are.

1. Your solution must have an acceptable layered architecture and be well designed.

This means it must follow the guidelines of the lecture on architecture, and of the

programming examples on the course web. You are, however, not required to use

exactly the same layers as in those examples.

1. Client and server must communicate by sending messages over a TCP connection,

using blocking TCP sockets.

1. The client shall only provide a user interface. It must not store any state, e.g.,

number of letters in the word, correctly guessed letters, number of remaining attempts,

total score, etc. All data entered by the user must be sent to the server for

processing, and all data displayed to the user must be received from the server.

1. The server is only allowed to send state, e.g., number of letters in the word, correctly

guessed letters, number of remaining attempts, total score, etc. The server is not

allowed to send any part of the view, like for example a string saying “You have 2

attempts left”, instead the server must only send the number of remaining attempts,

and the client shall insert it in the user interface.

1. The client must have a responsive user interface, which means it must be multithreaded.

The user must be able to give commands, for example to quit the

program, even if the client is waiting for a message from the server.

1. The server must be able to handle multiple clients playing concurrently, which

means it must be multithreaded.

1. The user interface must be informative. The current state of the program must be

clear to the user, and the user must understand what to do next.

There is also an optional task for higher grade which I have tried to accomplish:

1. Use a length header for all messages; each message sent must include the number of bytes

the message consists of. This means a sender must always calculate the message length

and prepend the number of bytes to the message. A receiver must always extract this

message length and check that it received the specified number of bytes. If it did not, it

must continue reading until the correct number of bytes have been received.

I will centre this report around the requirements of this task and show how I have completed them. I will also, throughout this report, refer to these requirements as “Req-*n”,* where “*n*” denotes the requirement entry at position n in this list.

**2 Literature Study**

Before I started with the given task, I studied all the videos for this assignment posted on YouTube by our lecturer. These videos gave example implementations of different distributed solutions and how to handle different output and input streams.

The video about the chat program gave me the most relevant information for completing the task of implementing the game “Hangman” for this assignment. I showed how to handle the different streams and it also gave a good example of the architecture. Once the architecture has presented, the logical functionality of the application made sense to me and I understood how the dependencies must be handled to achieve the correct functionality. The function of the server, and what tasks are to be performed on the server, contra what is handled on the client side of the application gave me a good base to start my work from. I always feel that I need to grasp the complete concept of a method before I will be able to implement it.

From the course, Concurrent Processes, I already had a good understanding of the concept of multithreading in Java. I also quickly understood the motivation for running a distributed applicating on several threads. This allows the server to serve many clients at the same time, without blocking any client, the same goes for the client side.

A concept which I had to study further was the class “CompletableFuture”, how it works and why it is necessary. For this I watched some videos on YouTube which explained the concept of CompletableFuture, placeholders, the common thread pool and more. I also studied the JavaDocs for more information about this class.

**3 Method**

To solve this assignment, I had to work incremental, where the first task was to create a client and a server. When these packages were created, I had to set up all the basic code for communication between the client and the sever. This included assigning sockets, port number and setting up “writers” and “readers” to be able to send and receive messages between the two applications.

Once this was setup, I could send a message and check that the communication between the server and client worked and when this was confirmed I cloud start to expand the applications whit case handlers for different commands from the input that were necessary for playing the game “Hangman”.

We had been provided with a flowchart of a simulated game to see what input our application needed to handle and how it should react to it.

I could also, thanks to the architecture explanation in one of the videos, model how an input from the client side had to travel and be handled through the different layers of the MVC-structure of the application. With this model in mind I cloud then start to create basic skeleton code consisting of different classes in the different layers that I knew I would be needing. I did this both the client side and the server side. Of course, some of these classes that were initially created showed to be redundant or more classes need to be created, but this gave me a good starting point of what functionality the completed application needed in order to be able to perform as whished.

While implementing the functionality of the application I also worked incremental, starting with the most general function, like sending a command to the server and then parsing the message for telling the game server what actions to perform in the game.

As soon as this functionality was implemented, I executed the actual command and tried to find different “edge cases” that would generate a runtime error or make the application behave in a manner that deviated from the specified flowchart.

For this I used the built-in debugger which made it easy for me to place breakpoint in the code and determine exactly what happened and what went wrong. Working in this manner made it easy for me to correct any bugs and make sure that all the requirements were fulfilled and that the application behaved as required.

Once I felt that I had meet the requirements of the basic task I cloud then move on to complete the advanced task of the assignment. Since everything else was already in place and tested I cloud exclusively focus on handling the header for the message passing between the applications.

For this assignment I used the IntelliJ IDE and the built-in debugger.

**4 Result**

This section explains the result of what you did.

Present your solution and prove that it meets the requirements listed in the homework

speci\_cation. Make sure to clearly explain how you met each requirement. You can,

for example, write a separate subsection for each requirement, where you mention the

requirement and explain how you met it. You can prove that a requirement is met by, for

example, explaining source code related to the requirement. If so, include a link to that

particular code in the Git repository, or clearly tell which method and class you intend.

You can also prove that a requirement is met by including, and explaining, screenshots

of the user interface.

Also, prove that you participated in writing the program, and that you understand it

in detail. You prove this by explaining essential parts of the source code.

**5 Discussion**

This section analysis the result presented in the previous section.

Summarize the requirements and clearly state which of them you have met. This might

seem like a meaningless repetition of the Result section, but it is not. Here, you shall

not prove that the requirements are met, just briey state what you achieved, that is,

which requirements where met. This summary is very useful when assessing your report.

Also, explain what lessons you learned, what problems you faced, and how the problems

were solved. Should something have been done di\_erently?

**6 Comments About the Course**

Any comment(s) related to this course o\_ering or to coming o\_erings is much appre-

ciated. Please also tell approximately how much time you spent on the assignment,

including lectures and exercises. This is of great help for improving the course.