1. Document version

Name	Version	Date
Amadeusz Misiak	1.1	09-04-2018

2. Objective

To create a java server which listens for a TCP connection and the instructions that are received along with the connection, in this case a /getStudents or /getStudents=”[course type]” HTML suffix. The request is to send back a response with a table of Students and their information, or page not found message if incorrect request sent. The students’ data is to be firstly retrieved from a MySQL database server at city.ac.telecom2.net:63006, then formatted into HTML 1.0 and forwarded to the client.

3. Solution overview

To achieve this firstly a listening class called SJWS will be created, this class will be responsible for initiating threads which can make sure multiple connections can be processed at the same time. Each thread will be created as soon as a connection is made with the client. This listening function, depending on the server’s current thread capacity, will initiate a different type of response for a free and busy server. For a busy server, the database access and other handling should be simplified to initiating streams and sending a short 503 response. The threads will implement the usual thread information (status, ID and no. of threads running).

Before the server begins to ‘listen’ the SJWS class needs to establish a connection with the database server first. To simply and abstract some of the functionality that comes with sql handling a database class will have to be implemented. It is responsible for all required database information, like driver location, host, port, url, database and login information. Initialisation of the database object will load the sql driver right away as it’s the first necessary component. Next, methods like connect and send statement are all that’s needed. Connect only used once initially, to connect to the database, and the send statement to be used as many times as possible with thread safety included to avoid race conditions from connection threads. Send statement only will require a String statement input and will abstract statement creation and query execution to return a ResultSet.

Next a connection thread interface class that requires a socket object and the currently connected database to initiate, or socket and server busy true flag to instruct the thread to disable database handling and only send busy response. Input and Output streams will firstly be setup to receive the request or send a response. Afterwards, depending on server busy flag, if false, proceed to retrieve client’s request and query the database accordingly. Once the data has been retrieved, the thread will have a method which takes the ResultSet received and formats the records into a HTML 1.0 table. However, depending on the client’s request the thread will either respond with a 404 Not found, if an invalid request is made, all student data or data filtered by course type only.

A simple time management abstract class is also added which all the classes will extend so that each use the same formats and methods for time.

All exceptions caught, and necessary libraries imported.

4. Conclusions

A lot of the functionality had to be abstracted into methods due to the complexity of their nature.

I have added colour coding to the console messages to easier distinguish between each action.

I have considered importing sql only into Database class(currently imported in Database and ConnectionThreadInterface), that way formatting to a HTML String from ResultSet would have to be done by the Database class. But this doesn’t align with the abstraction the database is responsible for.

To manage traffic on the server the amount of connections can be defined to allow only that number of connections open available. Meanwhile sending Server Busy back to the client.

Also no closing commands or functions implemented due to running on NetBeans I considered the extra feature unnecessary.