EECE 350 – Computer Networks

Section 3 – Dr. Rajai Nasser

Team 3:

Christian Wakim, 201901797

Majd Harake, 201902525

Paul Frangieh, 201902684

Client:

```
import java.io.IOException;
import java.util.*;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class Client{
    @SuppressWarnings("resource")
    public static void main(String args[]) throws Exception{
        try(Socket socket= new Socket("localHost",6000)){
            System.out.println("Enter lines of text");
            try(Scanner scanner = new Scanner(System.in)){
            Scanner in = new Scanner(socket.getInputStream());
            PrintWriter out = new
PrintWriter(socket.getOutputStream(),true);
            while(scanner.hasNextLine()) {
                String s=scanner.nextLine();
                if(s.equals("DONE")) {
                    break;
                }
                out.println(s);
                System.out.println(in.nextLine());
                System.out.println("type DONE if you're Done. If
not, continue normally");
            }
            socket.close();
            System.out.println("Thank you, have a good day");
            }
        }
    }
}
```

Server:

```
import java.io.IOException;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.*;
```

```
public class Server {
    public static void main(String args[]) throws Exception {
        try(ServerSocket listener= new ServerSocket(6000)){
            System.out.println("The statistics server is
running...");
            ExecutorService
pool=Executors.newFixedThreadPool(20);
            while(true) {
                pool.execute(new StatThread(listener.accept()));
            }
        }
    }

    public static class StatThread implements Runnable {
        private Socket socket;

        StatThread(Socket socket){
            this.socket=socket;
        }
        @Override
        public void run() {
            System.out.println("Connected: "+socket);
            try {
                Scanner in = new
Scanner(socket.getInputStream());
                PrintWriter out= new
PrintWriter(socket.getOutputStream(),true);
                while(in.hasNextLine()) {
                    out.println("Hello from the server");
                }
            }
            catch (IOException e) {
                e.printStackTrace();
            }
            finally {
                try {
                    socket.close();}catch(IOException e) {}
                System.out.println("Closed: "+socket);
            }
        }
    }
}
```

File Database:

```
import java.io.FileWriter;   // Import the FileWriter class
import java.io.IOException;  // Import the IOException class to
handle errors
import java.util.Scanner;
import java.io.BufferedWriter;
import java.io.Writer;
import java.io.File;
import java.io.BufferedReader;
import java.util.Vector;

public class UserPass {

    private static Scanner x;
  public static void main(String[] args) {

      String LogUsername, LogPassword;
      Scanner Log = new Scanner(System.in);
      System.out.print("Enter your username: ");
      LogUsername = Log.nextLine();
      System.out.print("Enter your password: ");
      LogPassword = Log.nextLine();
      String filepath = "UserPass.txt";
      verifyLogin(LogUsername, LogPassword, filepath);
    }




  public static void verifyLogin(String LogUsername, String
LogPassword, String filepath)
      {
          boolean found = false;
          String tempUsername = "";
          String tempPassword = "";
       try {
           x = new Scanner(new File(filepath));
           x.useDelimiter("[;\n]");

           while(x.hasNext() && !found)
           {
               tempUsername = x.next();
               tempPassword = x.next();
               if
(tempUsername.trim().equals(LogUsername.trim()) &&
tempPassword.trim().equals(LogPassword.trim()))
```

```
                {
                    found = true;
                    break;
                }
            }
                x.close();
                System.out.println(found);

        }
        catch (Exception e)
        {
            System.out.println("Error");
        }

    try {

      if (found == false) {

      String User, Pass;
      Scanner input = new Scanner(System.in);
      FileWriter myWriter = new FileWriter("UserPass.txt",
true);
      myWriter.write(LogUsername+";"+LogPassword+"\n");
      System.out.println("Successfully wrote to the file.");
      myWriter.close();

    }
}
    catch (IOException e) {

      System.out.println("An error occurred.");
      e.printStackTrace();

    }

  }

}
```

Task Division:

In the beginning, we re-watched the Java introductions and recordings posted by Dr. Jad Matta on Moodle individually, reviewing the concepts concerning Java coding, GUI sessions, and MySQL databases. After that, our team read the instructions of the project's overview to grasp a general idea on how to advance in our work.

Each one of us handled one of the links under "Useful References" (JAVA docs: http://java.sun.com/j2se/1.5.0/docs/api/ , JAVA tutorial: http://java.sun.com/docs/books/tutorial/, and JAVA sockets: http://java.sun.com/docs/books/tutorial/networking/sockets/) and explained the most important notes and findings to the other team members over online Zoom meetings and how we can use them to complete the first phase of the project.

After a lot of brainstorming and thought gathering, we worked for hand in hand and implemented the given prototype. The tasks were divided as follows: Server-side by Majd Harake, Client-side by Paul Frangieh, and creation of a database on text files by Christian Wakim. However, everything was put together by the three of us and re-checked several times so that everyone was aware of the others' work.

Problems we faced:

Throughout the project, we had to cope with a totally new programming language, which is Java. Since no one had any exposure to this language previously, we had to take it online and went to learn on Codecademy, TutorialsPoint, W3Schools, etc. to learn some basics in Java for us to be more familiar with the syntax. This by itself was a little bit complicated, especially to manage it alongside other courses' material and assignments. Still, we ended up grasping the basic syntax of this language.

We then made use of some of the material posted on Moodle by Dr. Jad Matta. We also went online to learn additionally and got a more definite conception of client servers and their implementation in Java.

Some of the issues we also faced were handling and compiling Files and Projects, which is a bit new. Still, this issue was quickly overcome after asking a few questions.

Besides, there was also the issue of time conflicts among us: All three of us are registered in different courses. When we had a common course, we were in different sections, so we always had different deadlines and exam times. This issue made it also harder for us to be able to have group meetings to coordinate our efforts.

One major issue was also the miscommunication and unclear guidelines that we had to abide by: We knew very late that we had to also work on a login and database.

Plans for phase 2:

Concerning phase 2, which is due May 2nd, we are considering tackling it as soon as possible since it is the major part of our project. The description is really detailed and needs a specific knowledge in Java, MySQL, and GUI. We will be working on the structure of the game with its different specifications. That would be step one. For step 2, we will be learning the implementation of a UI and linking it to our game code.

All our meetings are taking place on Zoom. The work isn't divided between the group members. However, it is done at the same time all together. As for the suggested schedule, we are trying as much as possible to meet on a weekly basis, 2 to 3 times. During these meetings, we will be searching online for more resources or contacting Dr. Matta to solve some small problems we could be facing.

Communications Protocol:

In our socket program prototype, we worked on to keep the code simple, to test our sockets and streams in addition to our client-server that all run on the same machine and communicate on the localhost.

The server had to be running first, thus enabling new clients to initiate connections. All messages sent by clients to the server were logged in a single .txt file.

Regarding the server, it was kept on the lookout for any new clients, and when a new client connects to the server, the latter creates a dedicated thread and socket for the client who gets assigned a unique credential that lasts the entire server session. Also, for each message received by the server, the latter responds with a confirmation and stores it in the .txt file specific to the

client in question. It is also to be noted that the multithreaded server we built can store and deliver messages to various clients and deliver their confirmations in parallel.

On the other hand, the client creates a socket to communicate with the server and gets notified on the console with the IP address and hostname where the server is located at the time of initiation of the connection. The role of the client is sending messages and receiving a receipt that it was delivered from the server to the Console of the client.