SCHOOL OF COMPUTING AND ENGINEERING SCIENCES

Bachelor of Science in Informatics and Computer Science

Bachelor of Science in Telecommunications

BTC4201 / ICS4104 – Distributed Systems

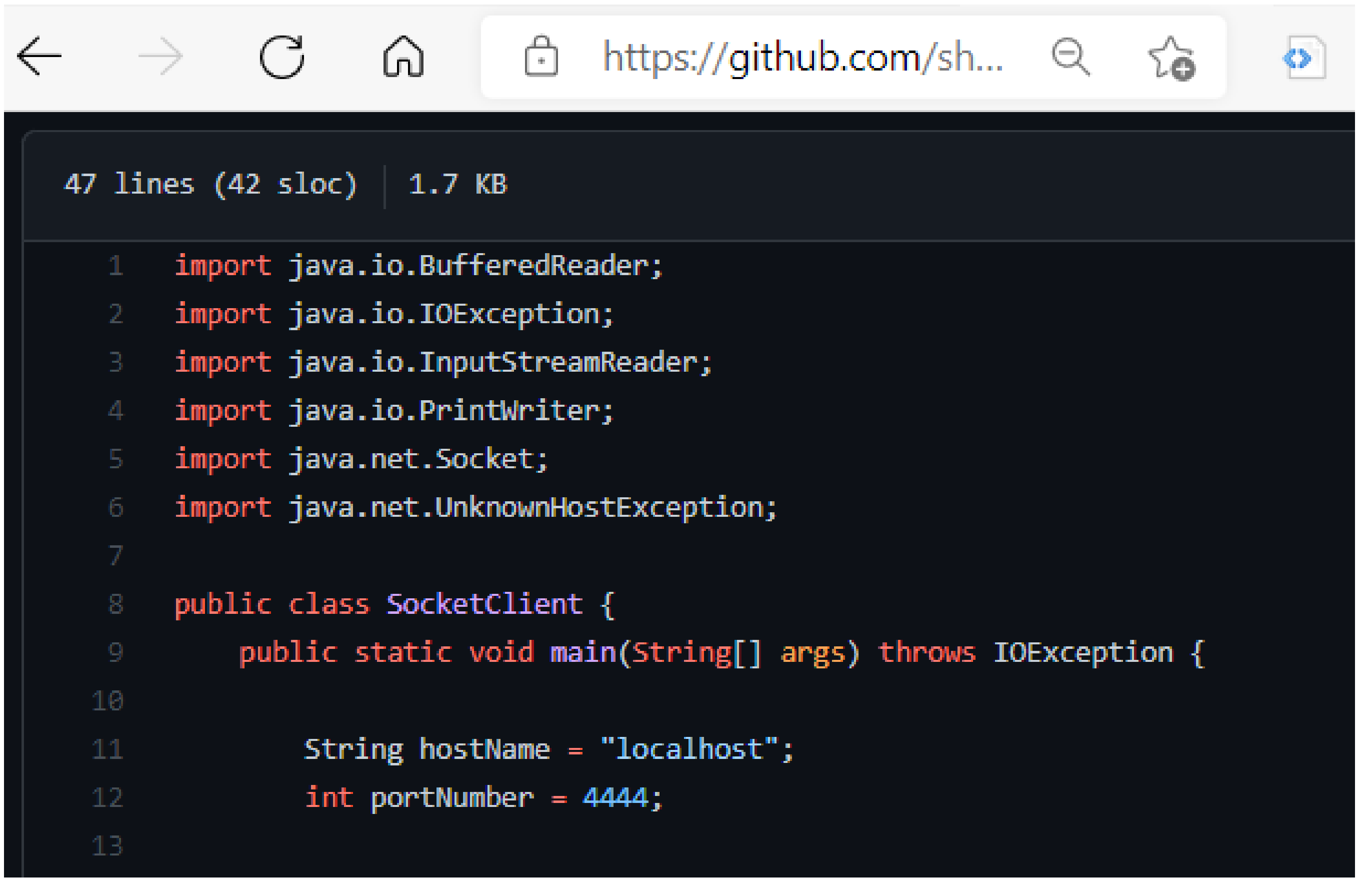
Assignment – Inter-process Communications in Distributed Environment

ICS 4B Group 1 Members

Shalom Orlando	111531
Fridah Nkirote	111033
David Ogalo	110946
Joshua Mochama	106272

A socket can be viewed as a means of connection between two nodes on a network. It is an endpoint of a two-way communication link between two programs running on the network (*What Is a Socket? (The Java^{TM} Tutorials > Custom Networking > All About Sockets)*, n.d.).The socket is bound to a port number hence enabling the TCP layer to identify the application that the data is destined to be sent to. In our implementation of this project, we have implemented various classes such as ClientProtocol.java, ServerProtocol.java, SocketClient.java and Socket Server.java. In addition to that, in creating a socket, we were to use localhost instead of IP address and port number.

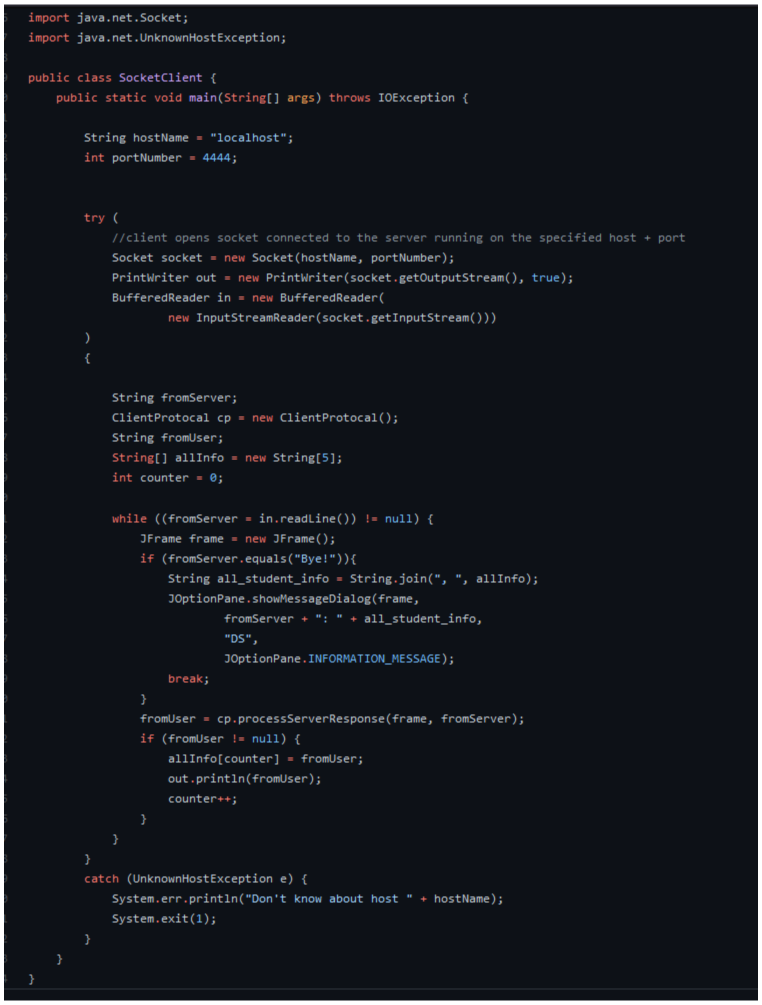
<u>SocketClient.java</u>
This class has been created as a connection point to the server according to the protocol described in ClientProtocol.java. In order to use localhost for the socket connection, we initialized the hostname and portNumber as follows:


https://github.com/sh...

47 lines (42 sloc) | 1.7 KB

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;

public class SocketClient {
    public static void main(String[] args) throws IOException {

        String hostName = "localhost";
        int portNumber = 4444;

```

Thereafter, we configured the client to open the socket connected to the server that was running on the specified host and port number as follows. We enclosed the particular code in a try-catch block as it was foreseen that it might have raised an exception.

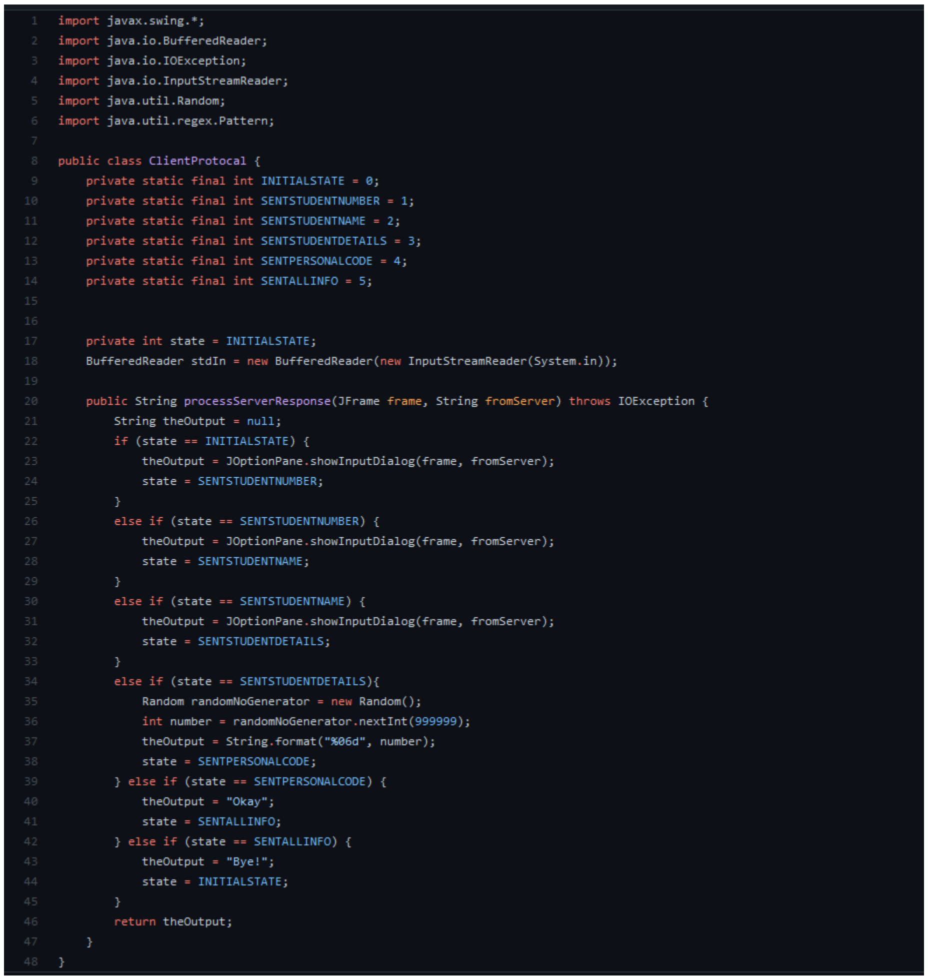


```java
import java.net.Socket;
import java.net.UnknownHostException;

public class SocketClient {
    public static void main(String[] args) throws IOException {

        String hostName = "localhost";
        int portNumber = 4444;


        try (
            //client opens socket connected to the server running on the specified host + port
            Socket socket = new Socket(hostName, portNumber);
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader in = new BufferedReader(
                    new InputStreamReader(socket.getInputStream()))
        )
        {

            String fromServer;
            ClientProtocal cp = new ClientProtocal();
            String fromUser;
            String[] allInfo = new String[5];
            int counter = 0;

            while ((fromServer = in.readLine()) != null) {
                JFrame frame = new JFrame();
                if (fromServer.equals("Bye!")){
                    String all_student_info = String.join(", ", allInfo);
                    JOptionPane.showMessageDialog(frame,
                            fromServer + ": " + all_student_info,
                            "DS",
                            JOptionPane.INFORMATION_MESSAGE);
                    break;
                }
                fromUser = cp.processServerResponse(frame, fromServer);
                if (fromUser != null) {
                    allInfo[counter] = fromUser;
                    out.println(fromUser);
                    counter++;
                }
            }
        }
        catch (UnknownHostException e) {
            System.err.println("Don't know about host " + hostName);
            System.exit(1);
        }
    }
}
```

Within the 'try' block, a new socket object was created, named *socket.* The socket constructor requires the host name and port number that you want to connect to. In our case, we had previously set the hostName as "localhost" and the port number as "4444". The second statement in the block gets the socket's output stream and it opens a PrintWriter on it with the name *out.*The third statement gets the socket's input stream and it opens a BufferedReader on it named *in*. We opt to use the readers and the writers so as to have Unicode characters written over the socket. Within the while block, information is read from and written to the stream in accordance with the server's protocol, the streams are then closed and the socket is closed as well. The server speaks first and the client listens by reading from the input stream that is attached to the socket. Communication between the two ends when the response from the server is "Bye!". After this, the client automatically closes the input and output streams and the socket as they were wrapped up in the try-with-resources statement.

ClientProtocol.java

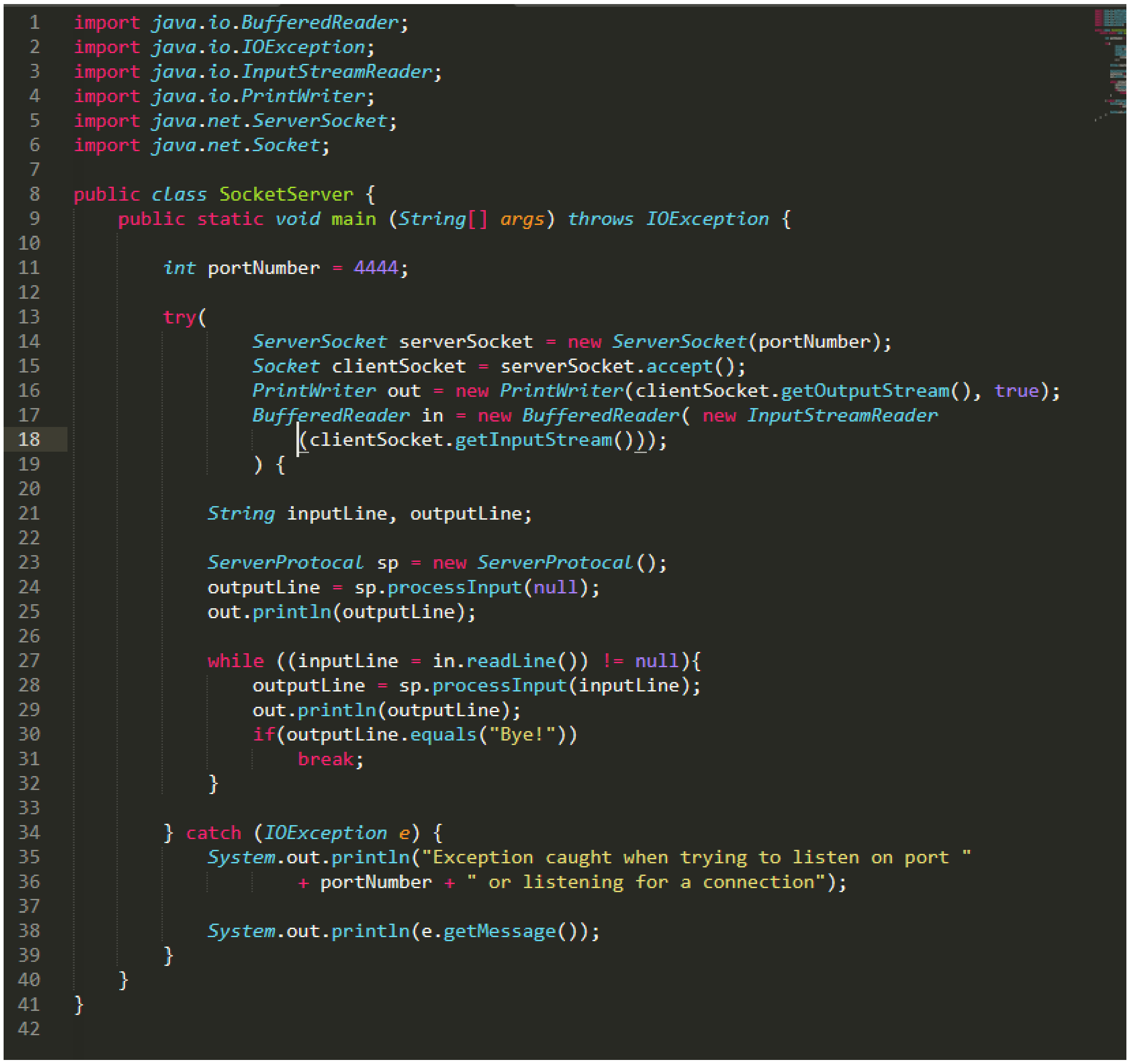
This class defines how the client responds to the servers' messages in that it contains the different client states. It implements the protocol used by the client and server to communicate. In addition to that, this protocol class keeps track of where the client and the server are in their conversation and it serves up the server's response to the client's statements. Using a protocol seemingly makes the data that is being passed back and forth to be meaningful. We use a random number generator to create a personal code.



```java
import javax.swing.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Random;
import java.util.regex.Pattern;

public class ClientProtocal {
    private static final int INITIALSTATE = 0;
    private static final int SENTSTUDENTNUMBER = 1;
    private static final int SENTSTUDENTNAME = 2;
    private static final int SENTSTUDENTDETAILS = 3;
    private static final int SENTPERSONALCODE = 4;
    private static final int SENTALLINFO = 5;


    private int state = INITIALSTATE;
    BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));

    public String processServerResponse(JFrame frame, String fromServer) throws IOException {
        String theOutput = null;
        if (state == INITIALSTATE) {
            theOutput = JOptionPane.showInputDialog(frame, fromServer);
            state = SENTSTUDENTNUMBER;
        }
        else if (state == SENTSTUDENTNUMBER) {
            theOutput = JOptionPane.showInputDialog(frame, fromServer);
            state = SENTSTUDENTNAME;
        }
        else if (state == SENTSTUDENTNAME) {
            theOutput = JOptionPane.showInputDialog(frame, fromServer);
            state = SENTSTUDENTDETAILS;
        }
        else if (state == SENTSTUDENTDETAILS){
            Random randomNoGenerator = new Random();
            int number = randomNoGenerator.nextInt(999999);
            theOutput = String.format("%06d", number);
            state = SENTPERSONALCODE;
        } else if (state == SENTPERSONALCODE) {
            theOutput = "Okay";
            state = SENTALLINFO;
        } else if (state == SENTALLINFO) {
            theOutput = "Bye!";
            state = INITIALSTATE;
        }
        return theOutput;
    }
}
```

<u>SocketServer.java</u>

This class sets a socket that listens to the client connection and enables communication between the server and the client. It receives data from the client and echoes it back.



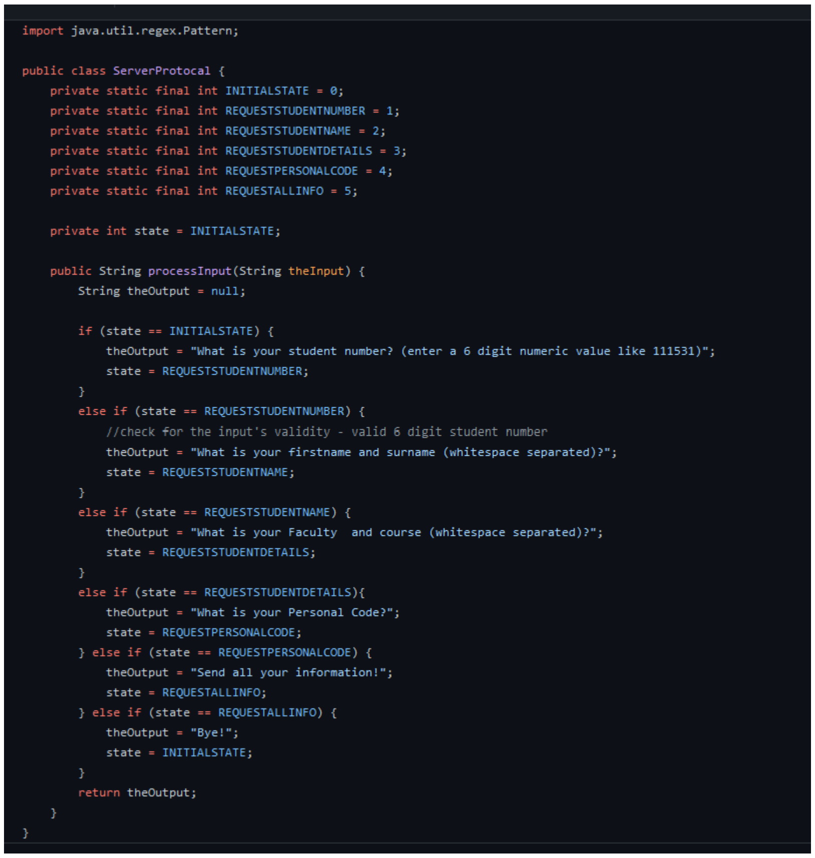
```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class SocketServer {
    public static void main (String[] args) throws IOException {

        int portNumber = 4444;

        try(
                ServerSocket serverSocket = new ServerSocket(portNumber);
                Socket clientSocket = serverSocket.accept();
                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
                BufferedReader in = new BufferedReader( new InputStreamReader
                    (clientSocket.getInputStream()));
                ) {

            String inputLine, outputLine;

            ServerProtocal sp = new ServerProtocal();
            outputLine = sp.processInput(null);
            out.println(outputLine);

            while ((inputLine = in.readLine()) != null){
                outputLine = sp.processInput(inputLine);
                out.println(outputLine);
                if(outputLine.equals("Bye!"))
                    break;
            }

        } catch (IOException e) {
            System.out.println("Exception caught when trying to listen on port "
                    + portNumber + " or listening for a connection");

            System.out.println(e.getMessage());
        }
    }
}
```

First, the port number is specified and accepts the initial communication from the client. It keeps the communication open until it receives a bye message from the client where it closes the communication.

It also throws an error if the client doesn't initiate communication after a while.

<u>ServerProtocol.java</u>

This class describes how your server reacts to the messages of your client. It was created in a methodology similar to the Knock Knock Protocol.

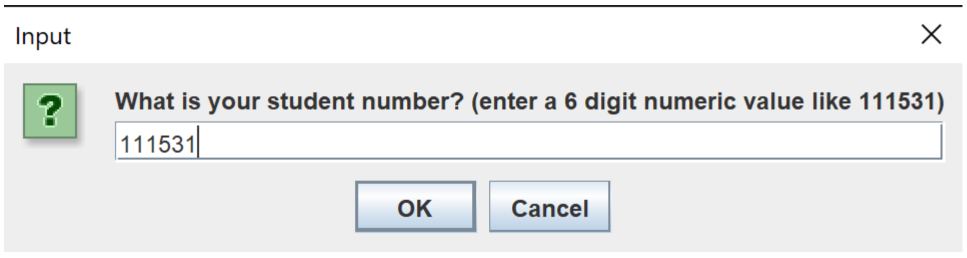


```java
import java.util.regex.Pattern;

public class ServerProtocal {
    private static final int INITIALSTATE = 0;
    private static final int REQUESTSTUDENTNUMBER = 1;
    private static final int REQUESTSTUDENTNAME = 2;
    private static final int REQUESTSTUDENTDETAILS = 3;
    private static final int REQUESTPERSONALCODE = 4;
    private static final int REQUESTALLINFO = 5;

    private int state = INITIALSTATE;

    public String processInput(String theInput) {
        String theOutput = null;

        if (state == INITIALSTATE) {
            theOutput = "What is your student number? (enter a 6 digit numeric value like 111531)";
            state = REQUESTSTUDENTNUMBER;
        }
        else if (state == REQUESTSTUDENTNUMBER) {
            //check for the input's validity - valid 6 digit student number
            theOutput = "What is your firstname and surname (whitespace separated)?";
            state = REQUESTSTUDENTNAME;
        }
        else if (state == REQUESTSTUDENTNAME) {
            theOutput = "What is your Faculty  and course (whitespace separated)?";
            state = REQUESTSTUDENTDETAILS;
        }
        else if (state == REQUESTSTUDENTDETAILS){
            theOutput = "What is your Personal Code?";
            state = REQUESTPERSONALCODE;
        } else if (state == REQUESTPERSONALCODE) {
            theOutput = "Send all your information!";
            state = REQUESTALLINFO;
        } else if (state == REQUESTALLINFO) {
            theOutput = "Bye!";
            state = INITIALSTATE;
        }
        return theOutput;
    }
}
```

Graphical User Interface

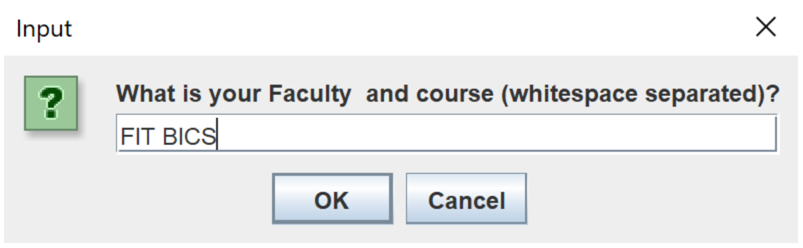
For the graphical interface we used Java Swings JFrame implementing a simple OptionPane.

Graphical User Interface Results



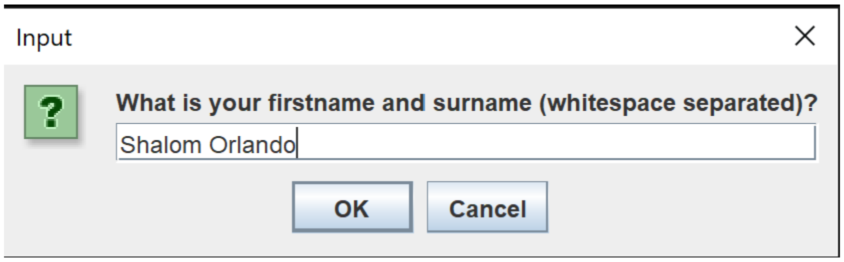
What is your student number? (enter a 6 digit numeric value like 111531)

111531


Input

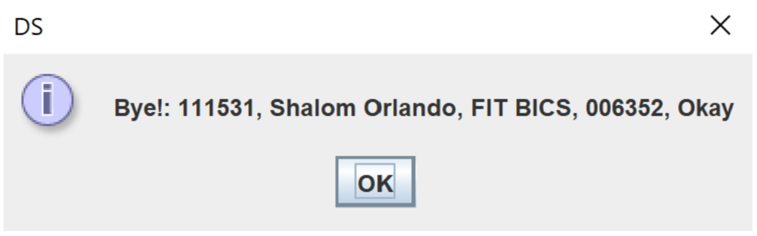
What is your Faculty and course (whitespace separated)?

FIT BICS


Input

What is your firstname and surname (whitespace separated)?

Shalom Orlando


DS

Bye!: 111531, Shalom Orlando, FIT BICS, 006352, Okay

How to run the Program

1. Open a Command Prompt Window/Terminal
2. Clone the repository using the git clone command
3. Move into the cloned directory using the cd command
4. Compile the project files using the javac command.

   "javac ServerProtocol.java SocketClient.java SocketServer.java"
5. Run the following command to start the server, in the current command prompt/terminal window.

   "java SocketServer"
6. In a new command prompt/terminal window, run the following command to start the client.

   "java SocketClient"