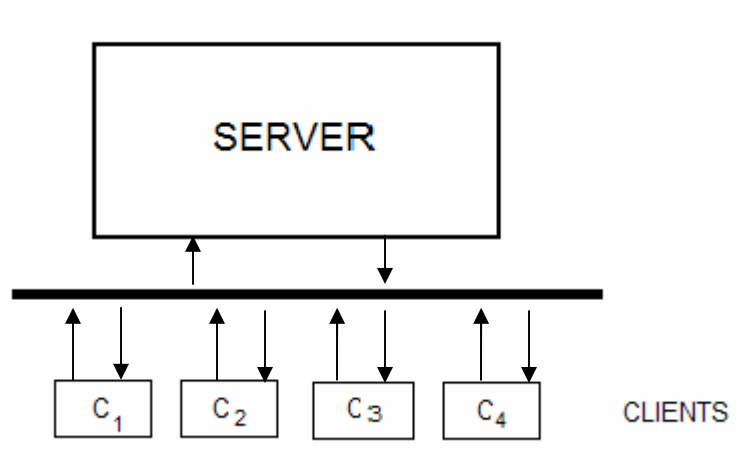
HomeWork 4

In this homework, a C2 application that allows one to query one's scores in a networked environment is presented.

[1] Architecture Description

The architecture that has been implemented using C2 has been documented here. The following few pages illustrate the description of the architecture.



The diagram above shows two components Server and Client. C_1, C_2, C_3, ..., C_n are several clients connected to the Server via the horizontal bar called the connector. Each of the clients interact with the server by sending a request and receiving the confirmation of the request through notification.

Client

The internal architecture of a client is shown below. However there are two such possibilities. At first , let us discuss about the various components of the Client architecture.

There are 3 components, namely, Message interface, Validator and Graphical User interface. Message interface component usage is to send request from the client to the server and receive response in the form of notification from the server. Validator is the component which is responsible for verifying the client request prior sending it to the server. And the graphical user interface is the component that interfaces with the user.

The two possible scenarios of such an architecture are shown below.

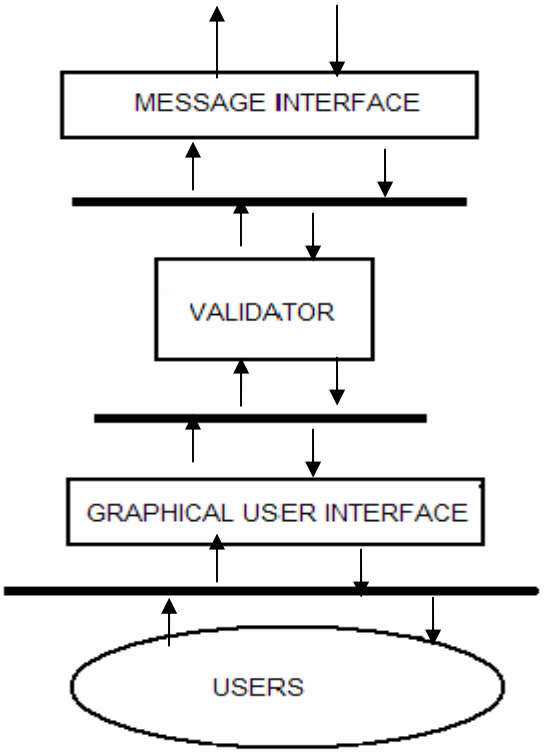


Fig. 1

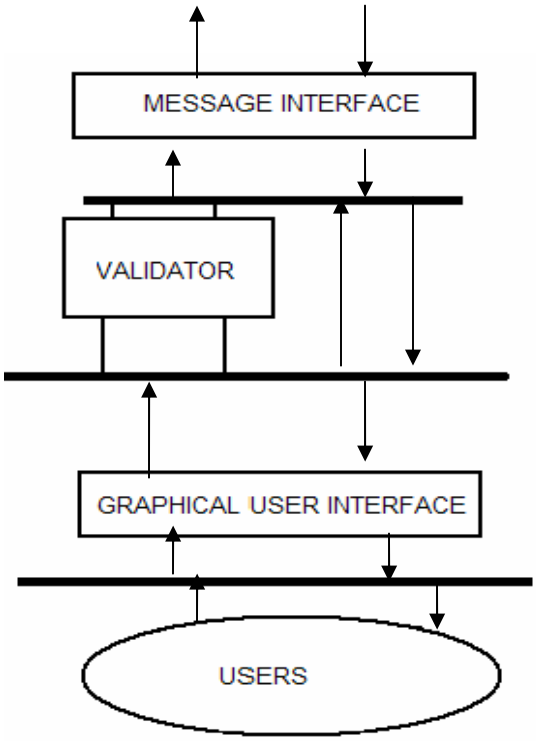


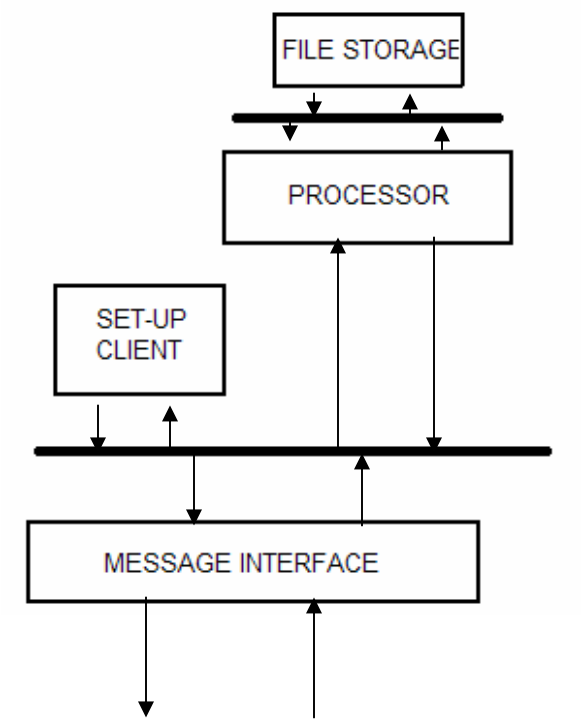
Fig.2.

The only difference between the two types of architecture is that there is a direct connection between two connectors as shown in fig.2. The reason is because when information flows from the server to the GUI there is no need to flow through the Validator component.

However the two connectors represent that they are interconnected. Hence they are not independent to each other. Thus they are not scalable. Hence we would implement the architecture depicted in fig.1.

Server

The internal architecture of the Server is depicted below:



The above diagram of the server architecture is self-explanatory. The message interface is a component that parses request from the client to the upper layers. The set-up client is the component that sets up every client to this server. The processor is the component that queries every request and sends the response to the client. The file storage component is a storage place from where the files are passed.

Set of Messages for each component

Messages are depicted as a triplet $< \alpha, \beta, \lambda >$.

α :- Component id, here id stands for the sender component that sends the request.

β :- Message id

λ :- The message itself that is to be sent through the pathway.

$\alpha \subseteq C_k, C_k$ are the components in the system.
$\beta \geq 1$
λ is a valid string.

Client

The precise set of messages of every component in the Client architecture is discussed below:

1. Message interface component:

This component denoted as C_1 is the bottom most layer of the client. The top port is directly connected to the main connector denoted as B_1. It sends request to and receives notification from B_1. The bottom port is connected to B_2. The set of messages for this component is formally defined below.

Request:

$\alpha \subseteq \{C_1\}$

$\beta \in \{M_1, M_2\}$; where M_1 is the connection setup message and M_2 is the query from the client.

λ is a valid string.

Notification:

$\alpha \subseteq \{C_2, C_3\}$

$\beta \in \{N_1, N_2\}$; where N_1 is the connection setup notification and N_2 is the notification for query from the client.

λ is a valid string.

2. Validator:

This component denoted as C_2 is the middle layer of the client. The top port is directly connected to B_2. It sends request to and receives notification from B_2. The bottom port is connected to B_3. The set of messages for this component is formally defined below.

Request:

$\alpha \subseteq \{C_2\}$

$\beta \in \{M_1, M_2\}$; where M_1 is the connection setup message and M_2 is the query from the client.

λ is a valid string.

Notification:

$\alpha \subseteq \{C_3\}$

$\beta \in \{N_1, N_2\}$; where N_1 is the connection setup notification and N_2 is the notification for query from the client.

λ is a valid string.

3. Graphical User interface:

This component denoted as C_3 is the lowermost layer of the client. The top port is directly connected to B_3. It sends request to and receives notification from B_3. The bottom port is connected to B_4. The set of messages for this component is formally defined below.

Request:

$\alpha \subseteq \{C_3\}$

$\beta \in \{M_1, M_2\}$; where M_1 is the connection setup message and M_2 is the query from the client.

λ is a valid string.

Notification:

$\alpha \subseteq \{C_4\}$
$\beta \in \{N_1, N_2\}$; where N_1 is the connection setup notification and N_2 is the notification for query from the client.
λ is a valid string.

4. Users:
This component is the I/O device.The message format for this component is given below:

Request:
$\alpha \subseteq \{C_4\}$
$\beta \in \{M_1, M_2\}$; where M_1 is the connection setup message and M_2 is the query from the client.
λ is a valid string.

Server

The precise set of messages of every component in the Server architecture is discussed below:

5. Message interface component:
This component denoted as C_5 is the topmost layer of the client. The top port is directly connected to B_5. It sends request to and receives notification from B_5. The bottom port is connected to the main connector B_1. The set of messages for this component is formally defined below.

Request:
$\alpha \subseteq \{C_5\}$
$\beta \in \{M_1, M_2\}$; where M_1 is the connection setup message and M_2 is the query from the client.
λ is a valid string.

Notification:
$\alpha \subseteq \{C_1\}$
$\beta \in \{N_1, N_2\}$; where N_1 is the connection setup notification and N_2 is the notification for query from the client.
λ is a valid string.

6. Set Up Client:
This component denoted as C_6 is the middle layer of the client. The top port is not connected to any connector. It acts as a sink. The bottom port is connected to B_5. The set of messages for this component is formally defined below.

Request:
None

Notification:
$\alpha \subseteq \{C_5\}$
$\beta \in \{N_1\}$; where N_1 is the connection setup notification.
λ is a valid string.

7. Processor:
This component denoted as C_7 is another layer of the client. The top port is directly connected to B_6. It sends request to and receives notification from B_6. The bottom port is connected to B_5. The set of messages for this component is formally defined below.

Request:
$\alpha \subseteq \{C_7\}$
$\beta \in \{M_2\}$; where M_2 is the request for query from the client.
λ is a valid string.

Notification:
$\alpha \subseteq \{C_5\}$
$\beta \in \{N_3\}$; where N_3 is the notification based on the search request send from the client
λ is a valid string.

8. File Locator:
This component is a storage device.The message format for this component is given below:

Request:
None

Notification
$\alpha \subseteq \{C_7\}$
$\beta \in \{M_2\}$; where M_2 is the notification for query from the client.
λ is a valid string.

Behaviour

The behaviour of messages that each component parses is given below:

Client

As seen from the above formal definition that messages are generally of two types:

1. Messages
2. Notification.

The messages are generally depicted as requests from each component depicted as M_k and the notification is the response to each request depicted as N_k.

In this domain, there are two types of messages.
M_1 :- This is a connection message request where every client sends this request for setting up with the server.
M_2 :- This is a query message from the client to the server.

There are three types of notification in this architecture.
N_1 :- This is a notification message of the connection request
N_2 :- This is a notification message from the server to the client of the query that it had requested
N_3 :- This is a notification message based on the search request send from the server File Storage to the Processor component of the server.

Set of Messages for each connector

$\{B_1,B_2,B_3,B_4,B_5\}$ are the set of connectors that share the same message types namely $\{M_1,M_2,N_1,N_2\}$. This is evident from the architectural diagram that was drawn above. However, connector B_6 uses only $\{M_2,N_3\}$ only.

Fornally, it can be shown that:

messages(B_i) $\in$ A where $1 \leq i \leq 5$ and A = $\{M_1,M_2,N_1,N_2\}$.
messages(B_6) $\in$ $\{M_2,N_3\}$

Behaviour

$\{B_2,B_3,B_4,B_6\}$ are the set of connectors that routes messages to the components connectced to it. This is because only one component is connected on the top and another on the bottom of each component. Whereas, $\{B_1,B_5\}$ broadcasts messages to the components Since more than 1 component is connected either on the top or bottom of the connector.

[2] Implementation using C2:

The following is the code for the Interface of Connector:

```
public interface IConnector {

	/* declaring functions for retrieving messages and notifications
*/

	public Object Read_Message();
```

```
	public Object Read_Notificaton();

	public void Put_Message(Object msg);
	public void Put_Notificaton(Object not);

}
```

The following is the code for the Interface of Component:

```
public interface IComponent {

	/* declaring the sub-components */
	public void Dialog_Constraints();
	public void Domain_Translator();
	public void Wrapper();
}
```

The following is the code for Component:

```
public class Connector implements IConnector {

	private Object msg;
	private Object not;

	public Connector(){

		Object msg = new Object();
		Object not = new Object();

	}
	public void Put_Message(Object msg) {
		// TODO Auto-generated method stub
		this.msg = msg;
	}

	public void Put_Notificaton(Object not) {
		// TODO Auto-generated method stub
		this.not = not;

	}

	public Object Read_Message() {
		// TODO Auto-generated method stub
		return this.msg;
	}

	public Object Read_Notificaton() {
		// TODO Auto-generated method stub
		return this.not;
	}

}
```

The following is the code for the Client:

```
/*
Done by: Amritam Sarcar
Date: 02 / 29 / 2008
 */
import java.awt.BorderLayout;
import java.text.NumberFormat;
import javax.swing.JButton;
import javax.swing.JFormattedTextField;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.*;

import java.io.*;
import java.net.*;
import java.awt.event.*;
import java.awt.*;

public class Client extends Connector{

	public static final Connector b3 = new Connector();
	public static Connector b2 = new Connector();

	/* Defining three components and instantiating them */
	public static final UserInterface c3 = new UserInterface(); // GUI
	public static final Validator c2 = new Validator(); // Validator
	public static final MessageInterface c1 = new MessageInterface(); // Method
interface

	public void run() {
		/* Calling the function run of component c3 */
		c3.run();
	}

	public static void main(String[] args){
		/* Instantiating a client object and calling the run method */
		new Client().run();
	}
}

class Validator implements IComponent{
	public void Wrapper(){
		// reads from the connector b3
		Object query = Client.b3.Read_Message();
		if(query!=null){
```

```
                        String sq = query.toString();
                        // parses the message to identify the message id
                        String[] msg = sq.split(";");
                        if(msg[0].equals("C3")){
                                if(msg[1].equals("M2")){
                                        // checks or consistency
                                        // sends a request to the connector b2
                                        Client.b2.Put_Message(query);
                                        Client.c1.Wrapper();
                                }
                                else if(msg[1].equals("M1")){
                                        Client.b2.Put_Message(query);
                                        Client.c1.Wrapper();
                                }
                        }
                }
        }

        public void Domain_Translator(){
                Object query = Client.b2.Read_Notificaton();
                if(query!=null){
                        String sq = query.toString();
                        String[] msg = sq.split(";");
                        if(msg[0].equals("C3")){
                                if(msg[1].equals("M2")){
                                        Client.b3.Put_Notificaton(query);
                                        Client.c3.Domain_Translator();
                                }
                                else if(msg[1].equals("M1")){
                                        Client.b3.Put_Notificaton(query);
                                        Client.c3.Domain_Translator();
                                }
                        }
                }
        }
        public void Dialog_Constraints(){
                // no implementation as such
        }
}
/* Definition of component class UserInterface */
class UserInterface extends JFrame implements ActionListener,IComponent{

        /* Defining variables to be used */
        JFormattedTextField tf;
        JButton button;
        JTextField pintf,nmetf,mrk1,mrk2,mrk3,mrk4,mrk5;;
```

```
JPanel panel,panel1,panel2,panel3;
JLabel label1,label2,label3,label4,label5,label6,label7;

/* Function for displaying the dialog box */
private void dialogBox(String msg, String header, int type){

        JOptionPane.showMessageDialog(this, msg, header, type);
}

public void Wrapper(){
        // does not send any notifications to anyone since it is the bottom most
        // layer
}
/* Disabling all the elements */
private void setDisable(){

        pintf.disable();
        nmetf.disable();
        mrk1.disable();
        mrk2.disable();
        mrk3.disable();
        mrk4.disable();
        mrk5.disable();

        label1.disable();
        label2.disable();
        label3.disable();
        label4.disable();
        label5.disable();
        label6.disable();
        label7.disable();
}

/* Enabling all the elements */
private void setEnable(){

        label1.enable();
        label2.enable();
        label3.enable();
        label4.enable();
        label5.enable();
        label6.enable();
        label7.enable();

        pintf.enable();
        nmetf.enable();
```

```
                mrk1.enable();
                mrk2.enable();
                mrk3.enable();
                mrk4.enable();
                mrk5.enable();
        }

        /* Displaying the Result set */
        private void displayResult(String ans[]){

                pintf.setText(ans[1]);
                nmetf.setText(ans[0]);
                mrk1.setText(ans[2]);
                mrk2.setText(ans[3]);
                mrk3.setText(ans[4]);
                mrk4.setText(ans[5]);
                mrk5.setText(ans[6]);
        }
        /* The run method */
        public void run(){
                /* Setting the properties */
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                this.setVisible(true);

                /* Checking if connection is done properly */
                Object ansquery = Client.b3.Read_Notificaton();
                if(ansquery!=null){
                        String sq = ansquery.toString();
                        String[] msg = sq.split(";");
                        if(msg[0].equals("C3")){
                                String[] connected
=Client.b3.Read_Notificaton().toString().split(";");


                                /* If there is connection failure */
                                if(!connected[2].equals("OK")){
                                        dialogBox(connected[2],"Server-Client Connection
Error", JOptionPane.ERROR_MESSAGE);
                                        System.exit(0);
                                }
                        }
                }
        }
        /* Setting the layout */
        private void setTheLayout() {
```

```
getContentPane().setLayout(new GridLayout(4,1));

/* Setting the layout */
panel = new JPanel();
panel1 = new JPanel();
panel2 = new JPanel();
panel3 = new JPanel();
JLabel label = new JLabel("Enter the PIN :");

pintf = new JTextField(3);
label1 = new JLabel("PIN :");

nmetf = new JTextField(10);
label2 = new JLabel("NAME :");

mrk1 = new JTextField(3);
mrk2 = new JTextField(3);
mrk3 = new JTextField(3);
mrk4 = new JTextField(3);
mrk5 = new JTextField(3);
label3 = new JLabel("PHY :");
label4 = new JLabel("CHEM :");
label5 = new JLabel("BIO :");
label6 = new JLabel("MATHS :");
label7 = new JLabel("TOTAL :");
tf = new JFormattedTextField(NumberFormat.getIntegerInstance());
tf.setColumns(10);
panel.add(label);
panel.add(tf);
button = new JButton();
button.setLabel("Search");

button.addActionListener(this);
panel.add(button);

getContentPane().add(panel, BorderLayout.WEST);

panel1.add(label1);
panel1.add(pintf);
getContentPane().add(panel1, BorderLayout.WEST);

panel2.add(label2);
panel2.add(nmetf);
getContentPane().add(panel2, BorderLayout.WEST);

panel3.add(label3);
```

```
		panel3.add(mrk1);
		panel3.add(label4);
		panel3.add(mrk2);
		panel3.add(label5);
		panel3.add(mrk3);
		panel3.add(label6);
		panel3.add(mrk4);
		panel3.add(label7);
		panel3.add(mrk5);
		getContentPane().add(panel3, BorderLayout.WEST);

		setDisable();
}

/* Default constructor */
public UserInterface(){
		setSize(600,300);
		setTitle("Server Client Interaction - Amritam Sarcar");
		setTheLayout();
		String query="C3;M1;";
}

/* Action Listener */
public void actionPerformed(ActionEvent ae) {

		/* Accessing the sendQuery method through port out */
		String query = "C3;M2;"+tf.getText();
		Client.b3.Put_Message(query);//
		Client.c2.Wrapper();
}
public void Dialog_Constraints(){
		String query = "C3;M1;"+tf.getText();
		Client.b3.Put_Message(query);//

}

/* Function definition of Domain_Translatory. */
public void Domain_Translator(){

		Object ansquery = Client.b3.Read_Notificaton();
		if(ansquery!=null){
			String sq = ansquery.toString();
			String[] msg = sq.split(";");
			if(msg[0].equals("C3")){

				/* Splitting the answer into different strings */
```

```
                                String ans[] = msg[2].split(",");

                                /* If the answer is NULL then dialogBox is shown */
                                if(ans[0].equals("Null")){
                                        dialogBox("PIN value does not match. Please verify
and re-enter","Database Error",
JOptionPane.WARNING_MESSAGE);
                                        setDisable();
                                        repaint();
                                }

                                /* if the answer is FALSE meaning a server side error */
                                else if (msg[3].equals("false")){
                                        dialogBox(ans[0],"Server- Client Error",
JOptionPane.ERROR_MESSAGE);
                                        System.exit(0);
                                }

                                /* if the file to be searched is not found */
                                else if (msg[2].equals("Database connectivity problem.
Please try again, after some time")){
                                        dialogBox(msg[2],"Database Connectivity Error",
JOptionPane.WARNING_MESSAGE);
                                        setDisable();
                                        repaint();
                                }

                                /* if the server returns the right data set */
                                else{
                                        setEnable();
                                        displayResult(ans);
                                }
                        }
                }
        }
}

class MessageInterface implements IComponent{
        public void Dialog_Constraints(){

        }
        public void Domain_Translator(){

        }
        //Defining the temporary values
        private BufferedReader br;
```

```
private PrintStream ps;
private Socket socket = null;
private String error = "OK";
private String answer = null;

// Defining the default constructor
public MessageInterface(){
	try{
		//Establishing the TCP/IP protocol connection
		byte[] ipAddr = new byte[]{127, 0, 0, 1};
		socket = new Socket();
		InetAddress addr = InetAddress.getByAddress(ipAddr);
		socket.connect(new InetSocketAddress(addr, 6000));
		br = new BufferedReader(
				new InputStreamReader(socket.getInputStream()));
		ps = new PrintStream(socket.getOutputStream());

	}
	catch(Exception e) {
		error = new String();
		error = "C3;M1;";
		error += e.getMessage()+";false";
		Client.b3.Put_Notificaton(error);
		Client.c2.Wrapper();
	}
}

public void Wrapper(){
	Object ansquery = Client.b2.Read_Message();
	if(ansquery!=null){
		String sq = ansquery.toString();
		String[] msg = sq.split(";");
		if(msg[0].equals("C3")){
			try{
				ps.println(sq);
				ps.flush();
				answer= br.readLine();
				answer+=";true";
				Client.b2.Put_Notificaton(answer);

				answer = null;
			}
			catch(Exception e) {
				answer="C3;N1;" +e.getMessage()+";false";
			}
			finally {
```

```
                                    Client.c2.Domain_Translator();
                              }
                        }
                  }
            }
```

The following is the code for the Server:

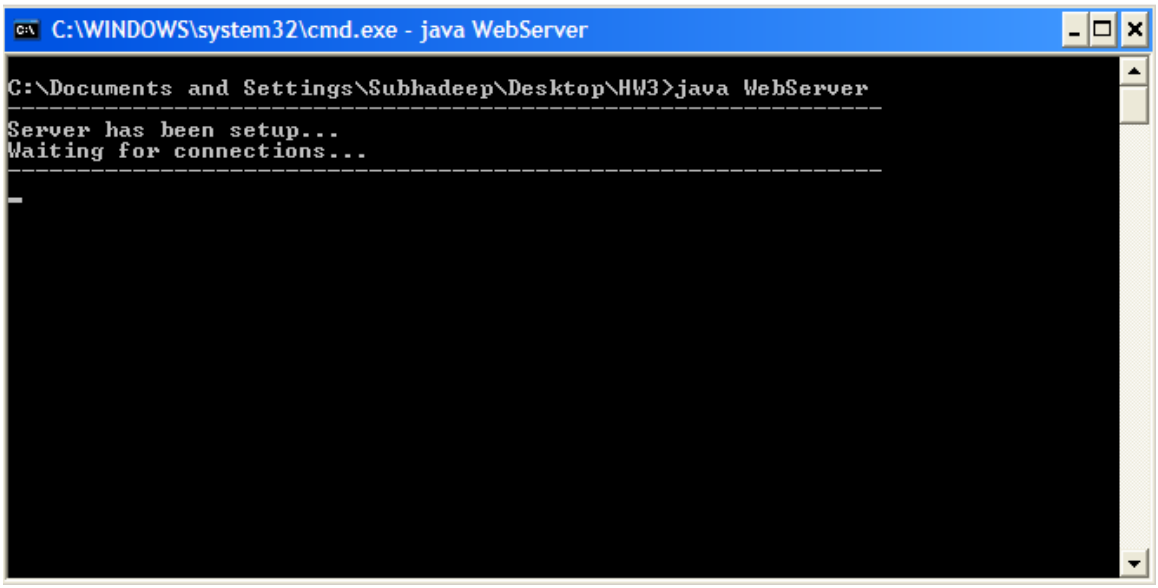
```
/*
Done by: Amritam Sarcar
Date: 02 / 29 / 2008
 */

import java.io.*;
import java.net.*;
import java.util.Scanner;

public class Server {

      private int countWorker = 0;
      private final SetupClient r = new SetupClient(this);

      public Server(){
            System.out.println("----------------------------------------
------------------------");
            System.out.println("Server has been setup...");
            System.out.println("Waiting for connections...");
            System.out.println("----------------------------------------
------------------------");
      }
      public static void main(String[] args){
            new Server().run();
      }

      public void run() { r.listen(); }

      public Processor requestWorker() {
            final Processor newWorker = new Processor(++countWorker);
            System.out.println("Client " + countWorker + " is
attempting to connect.");
            System.out.println("Attempt Success, connected!!");
            System.out.println("----------------------------------------
------------------------");
            return newWorker;
      }
}

class SetupClient {
      Server s;
      public SetupClient(Server s){
            this.s = s;
      }

      Processor workers;
```

```
	public void listen() {
		try{
			ServerSocket server = new ServerSocket(6000);
			while (true) {
				Socket sock = server.accept();
				workers = s.requestWorker();


	workers.httpRequest(sock.getInputStream(),sock.getOutputStream())
;
			}
		}
		catch(Exception e) {
			System.out.println(e.getMessage());
		}
	}
}

class Processor extends Thread {

	private int workerNo;
	private int noOfQuery = 0;
	private BufferedReader pBr;
	private PrintStream pPs;
	private String query, answer;
	private String[] q;

	void httpRequest(InputStream in,OutputStream out) {
		pBr = new BufferedReader(new InputStreamReader(in));
		pPs = new PrintStream(out);
		start();
	}
	public Processor(){}
	public Processor(int workerNo){
		this.workerNo = workerNo;
	}

	public void run() {

		try{
			while(true)
			{
				String[] q = pBr.readLine().split(";");
				query = q[2];
				System.out.println("Servicing Client " +
workerNo);
				System.out.println("No. of times it has
accessed Server :: " + (++noOfQuery));
				System.out.println("Received request. PIN no: "
+ query);
				processLineByLine();
				System.out.println("Sending response.");
				System.out.println("Response is :: " + answer);
				System.out.println("-------------------------
-----------------------------------");
				pPs.println(answer);
				pPs.flush();
```

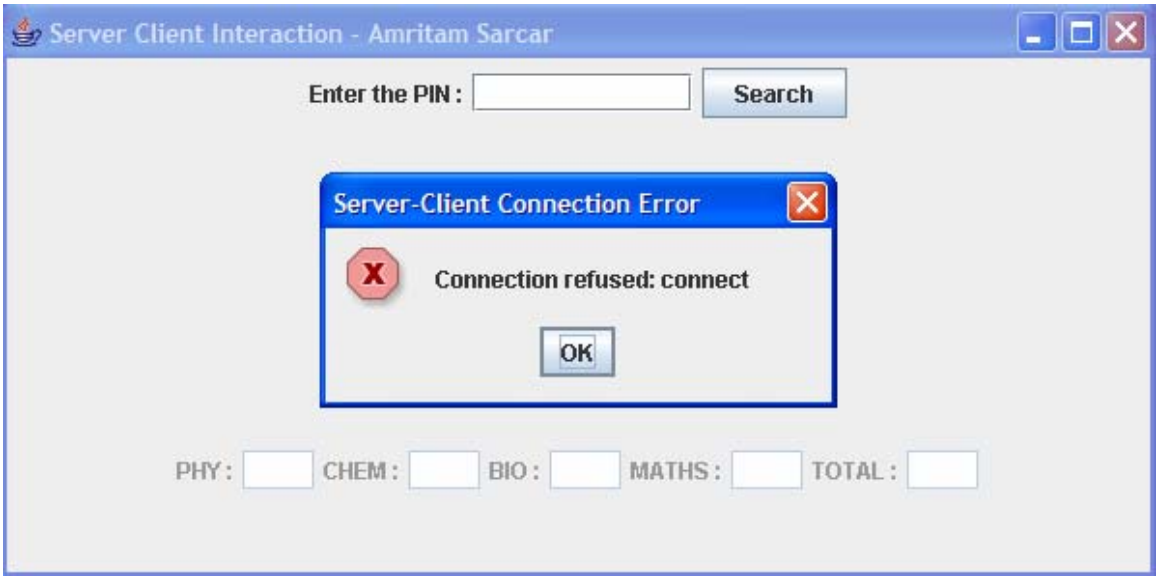
```
                        answer = null;
                  }
            }
            catch(Exception e) {
                  System.out.println(e.getMessage() + " Client No :: "
+ workerNo + " has exited");
                  System.out.println("----------------------------------
------------------------------");

            }

      }

      public final void processLineByLine(){
            try {
                  //first use a Scanner to get each line
                  Scanner scanner = new Scanner( new File(
"C:\\text.txt" ));
                  while ( scanner.hasNextLine() ){
                        if (processLine( scanner.nextLine() ) == 1)
return;
                  }
                  scanner.close();
                  answer="C3;M1;Null";
            }
            catch (IOException ex){
                  System.out.println(ex.getMessage());
                  answer = "C3;M1;Database connectivity problem. Please
try again, after some time";
            }
      }

      protected int processLine(String aLine){
            Scanner scanner = new Scanner(aLine);
            scanner.useDelimiter(",");
            if ( scanner.hasNext() ){
                  String name = scanner.next();
                  String pin = scanner.next();
                  String marks = scanner.next();
                  if( pin.equals(query) ){
                        answer = "C3;M2;" + aLine;
                        return 1;
                  }
            }
            else {
                  answer = "C3;M2;Empty or invalid line. Unable to
process.";
            }
            scanner.close();
            return 0;
      }
}
```

Output

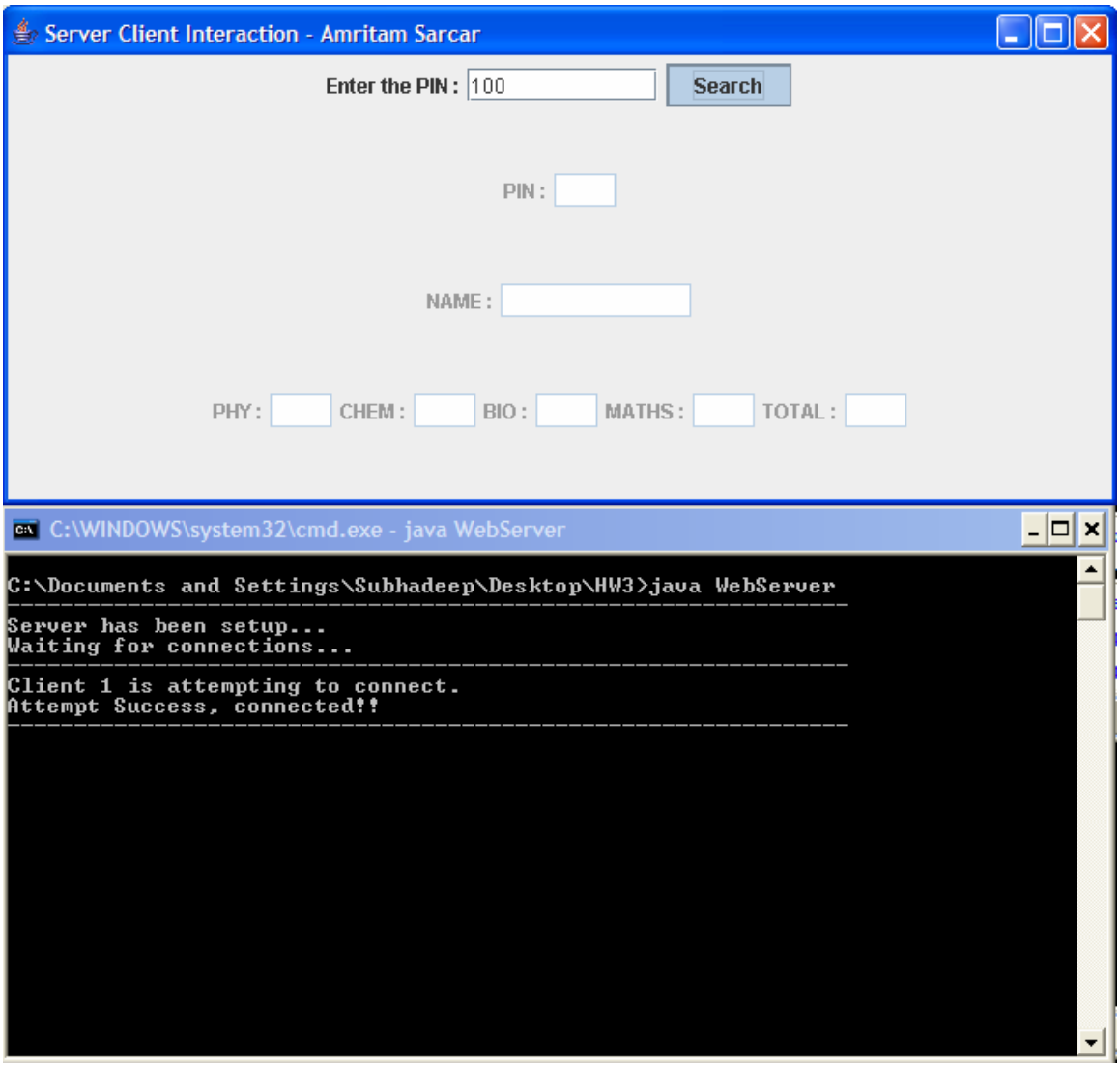
The following snapshots were taken in running the server and the client.



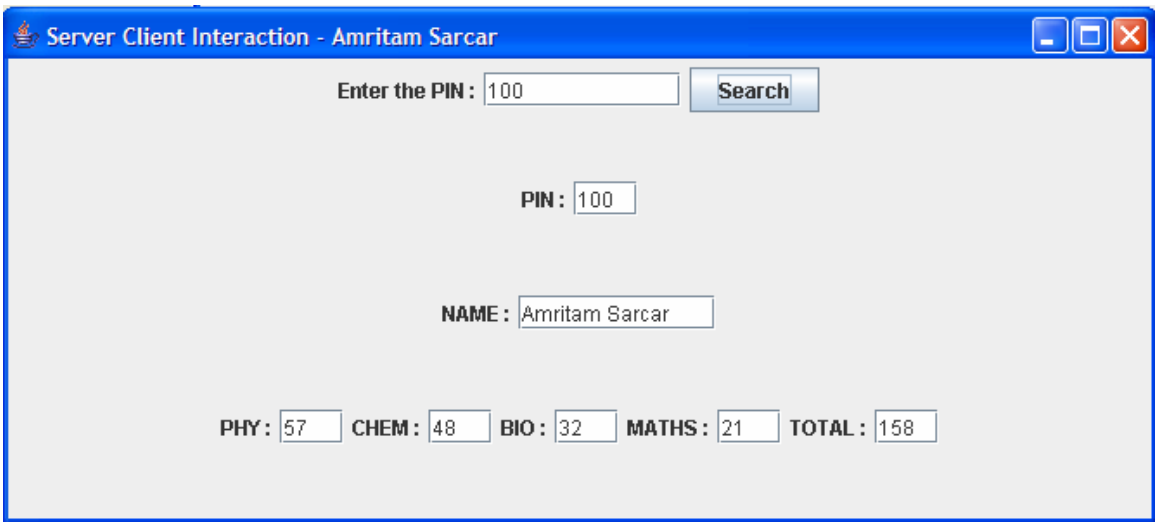
Running the WebServer without starting the client program



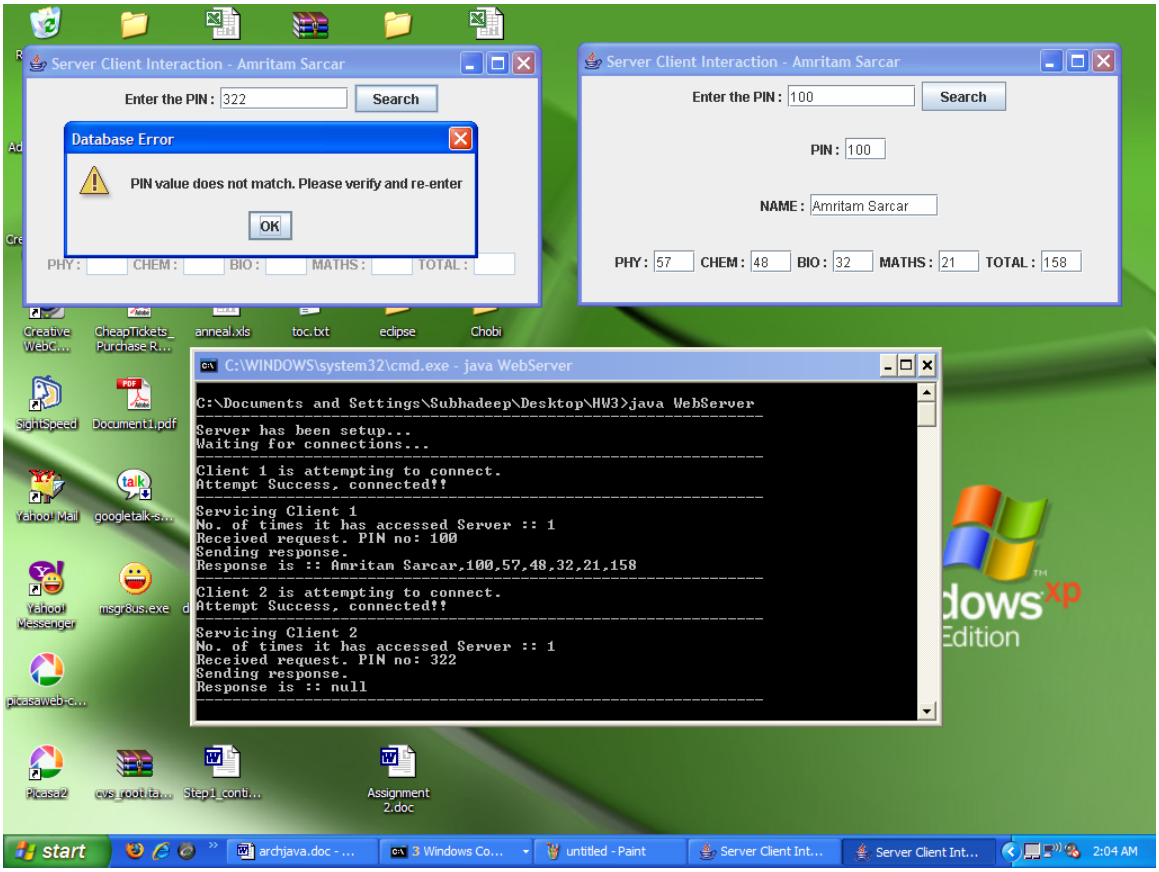
Running the WebClient without starting the server program



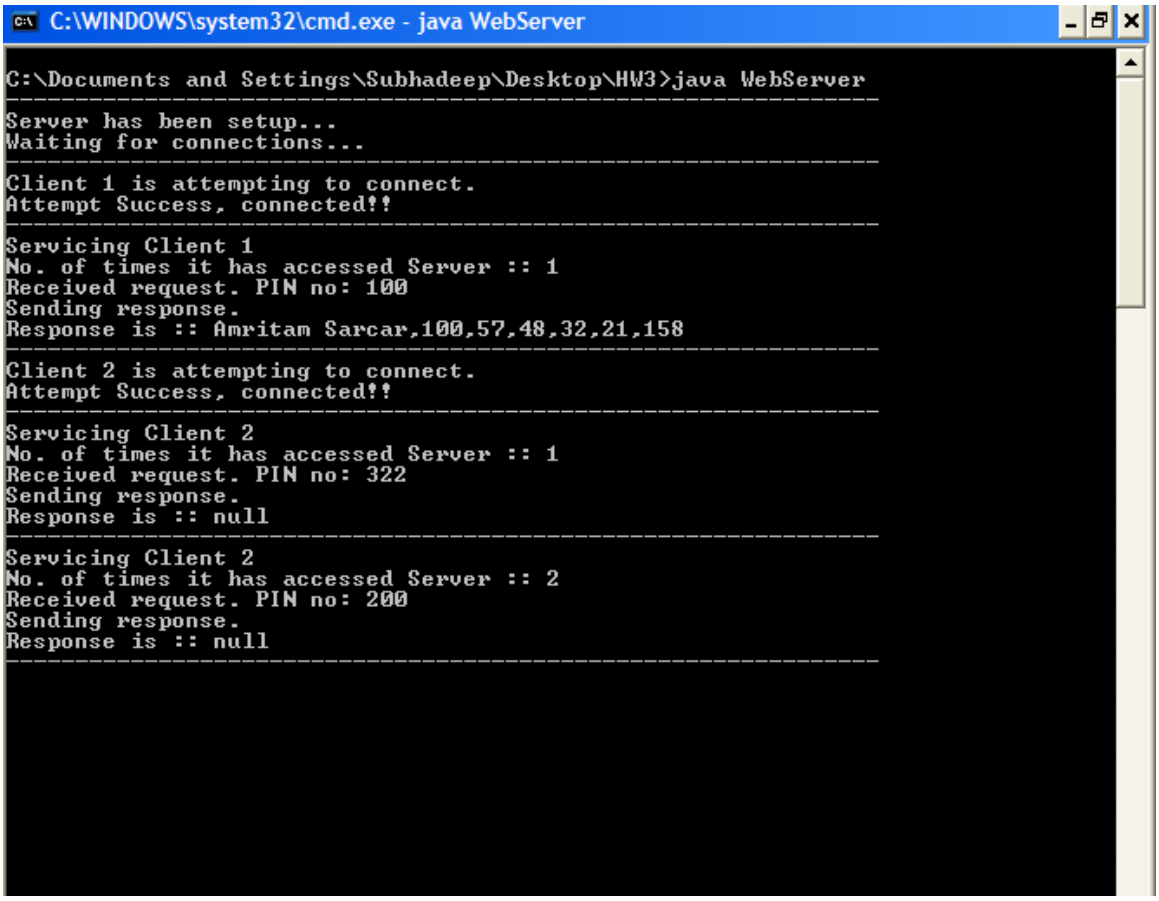
Client and Server interacting with each other.



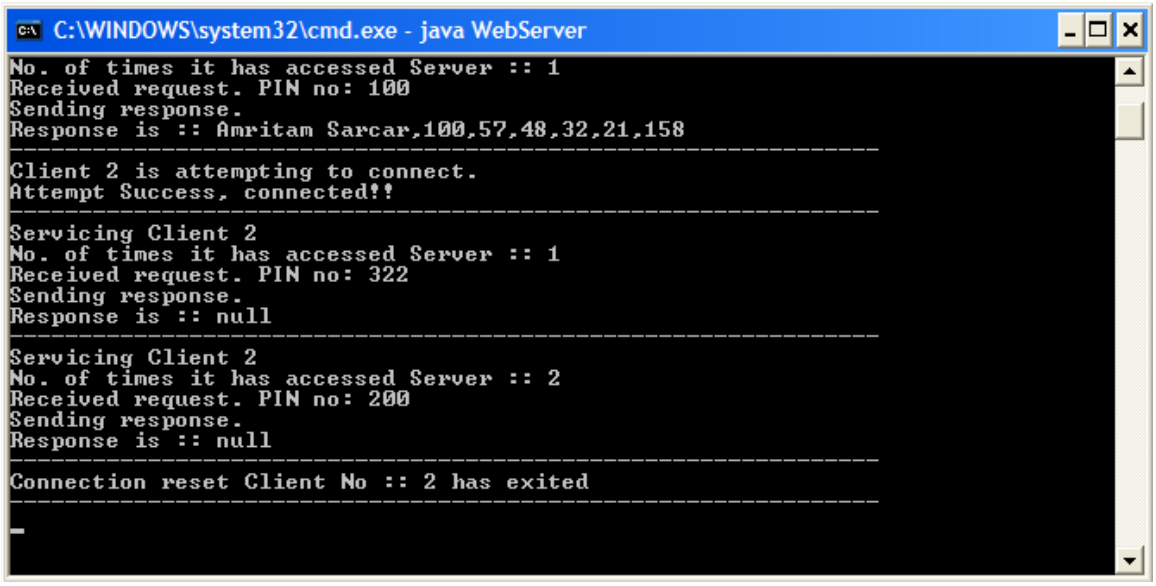
Corresponding marks and name for pin 100



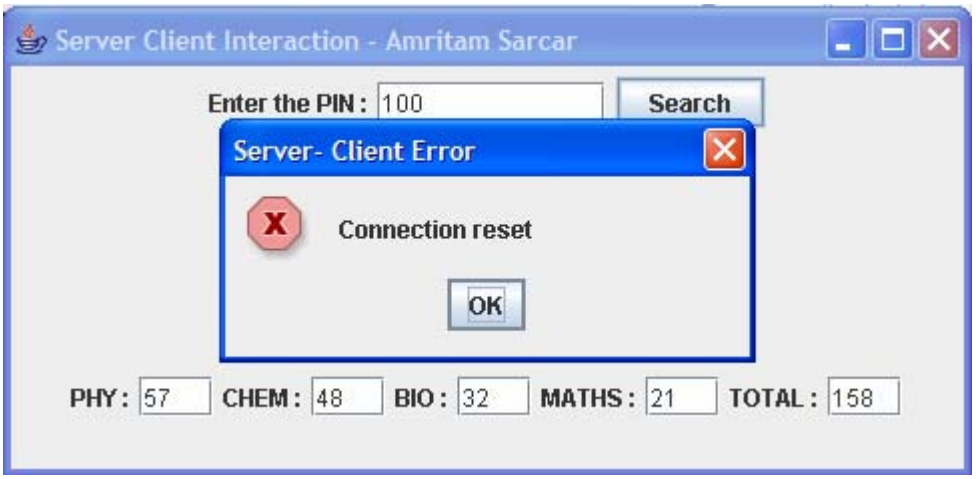
Two clients interacting and one client enters the wrong pin



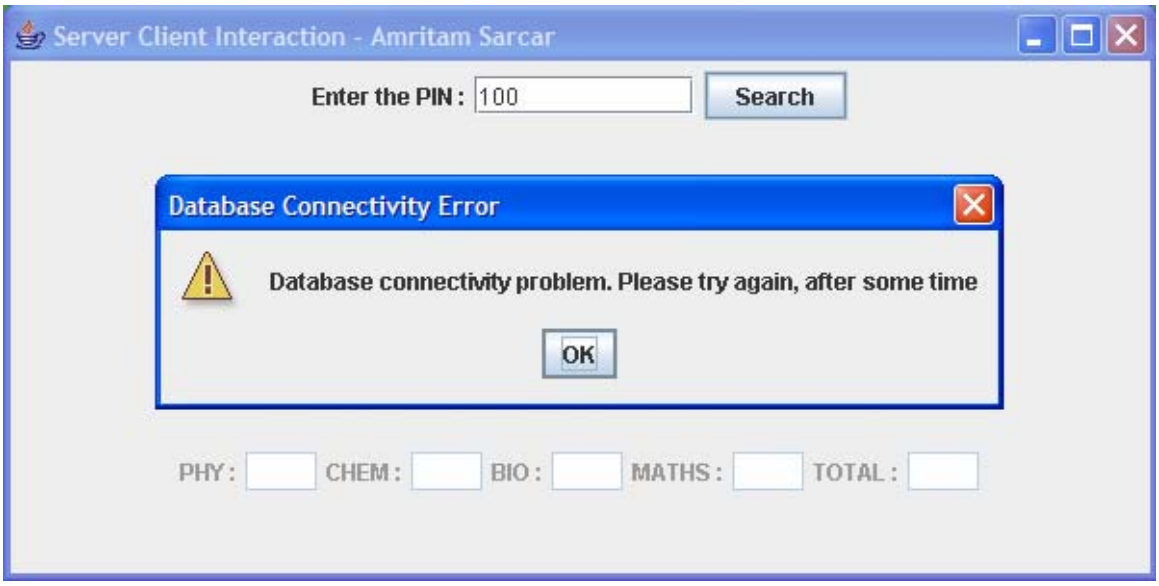
Keeping count of multiple clients and their individual access counts.



One of the connected clients being disconnected.



After successful connection, the server disconnects… client trying to access the server



The file/ database to be accessed cannot be found.

Reference:

1. http://www.cs.utep.edu/~cheon/cs5381/homework/network.ppt#1
2. http://java.sun.com/docs/books/tutorial/uiswing/