**Chapter (1)**

**Introduction**

* 1. **Tic Tac Toe Multiplayer Game**

This project is implemented by using the idea of Remote Method Invocation (RMI) in Java for tic tac toe multiplayer game. This project describes the basic feature of RMI. This project also presents simple example of tic tac toe multiplayer game that is implemented by Java RMI. Tic-tac-toe (American English), noughts and crosses (British English), or Xs and Os is a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row is the winner.

* 1. **Objectives**

The main objectives of our system are:

* To play the Tic Tac Toe game with 2 player or multiplayer or players around the world
* To request different available users
* To replay current user
  1. **System Development Environment**

Setup the following software on your PC for the system development.

* JDK(1.6) or Above
* Eclipse (Indigo, etc.)

This system is implemented by Java Language using the idea of Remote Method Invocation (RMI).

**Chapter (2)**

**Theory Background**

**2.1 Remote Method Invocation (RMI)**

Remote Method Invocation (RMI) is a way that a programmer, using the Java programming language, can write object-oriented programming in which objects on different computers can interact in distributed network. RMI is the Java version of what is generally known as a remote procedure call (RPC), but with the ability to pass one or more objects along the request. The object can include information that will change the service that is performed in the remote computer.

The following two fundamental concepts are at the heart of distributed object model:

Remote object reference: Other objects can invoke the method of a remote object if they have access to its remote object reference. For example, a remote object reference for B in Figure 1 must be available to A.

Remote interface: Every remote object has a remote interface that specifies which of its methods can be invoked remotely. For example, the objects B and F in Figure 1 must have remote interface.

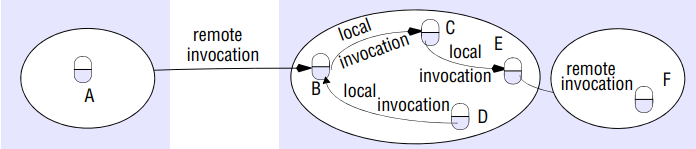


Figure 1 : Remote and local method invocation

The role of each of the components shown in Figure 2, dealing first with the communication and remote reference modules and with the RMI software that runs over them. The following related topics are the generation of proxies, the binding of names to their remote objects references, the activation and passivation of objects from their remote object references.

Communication module: the two cooperating communication modules carry out the request-reply protocol, which transmits request and reply messages between the client and server.

Remote reference module: A remote reference module is responsible for translating between local and remote object references and for creating remote object references.

Servants: A servant is an instance of a class that provides the body of a remote object. Servants live within a server process.

The RMI software: This consists of a layer of software between the application-level objects and the communication and remote reference modules. The role of middleware objects shown in Figure 2 are as follows.

Proxy: The role of the proxy is to make remote method invocation transparent to clients by behaving like a local object to the invoker. But instead of executing an invocation, it forwards it in a message to a remote object. It hides the details of the remote object reference, the marshalling of arguments, unmarshalling of results and sending and receiving of messages from the client. There is one proxy for each remote object for which a process holds a remote object reference.

Dispatcher: A server has one dispatcher and one skeleton for each class representing a remote object. The dispatcher receives request messages from the communication module.

Skeleton: A skeleton method unmarshals the arguments in the request message and invokes the corresponding method in the servant. It waits for the invocation to complete and then marshals the result, together with any exceptions, in a reply message to the sending proxy’s method.

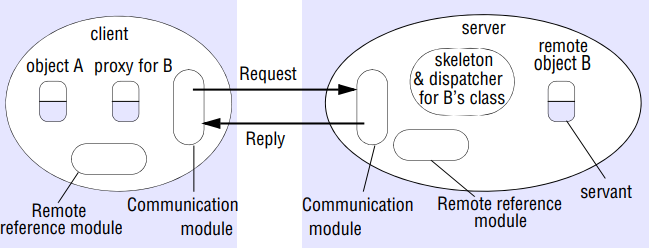


Figure 2 : The role of proxy and skeleton in remote method invocation

**Chapter (3)**

**Implementation**

## **Starting the Server**

Before starting the convert engine, you need to start the RMI registry. The RMI registry is a simple server-side bootstrap naming facility that enables remote clients to obtain a reference to an initial remote object. It can be started with the rmiregistry command. Before you execute rmiregistry, you must make sure that the shell or window in which you will run rmiregistry either has no CLASSPATH environment variable set or has a CLASSPATH environment variable that does not include the path to any classes that you want downloaded to clients of your remote objects.

To start the registry on the server, execute the rmiregistry command. This command produces no output and is typically run in the background. For this example, the registry is started on the host my computer.

**. start rmiregristry**

Once the registry is started, you can start the server. You need to make sure that both the ClientTTTIF.java file and the remote object implementation class are in your class path. When you start the compute engine, you need to specify, using the java.rmi.server.codebase property, where the server’s classes are network accessible. In this example, the server-side classes to be made available for downloading are the Compute and Task interfaces, which are available in the TTTServiceIF.java file and ClientTTTIF.java file. The compute engine server is started on the host mycomputer, the same host on which the registry was started.

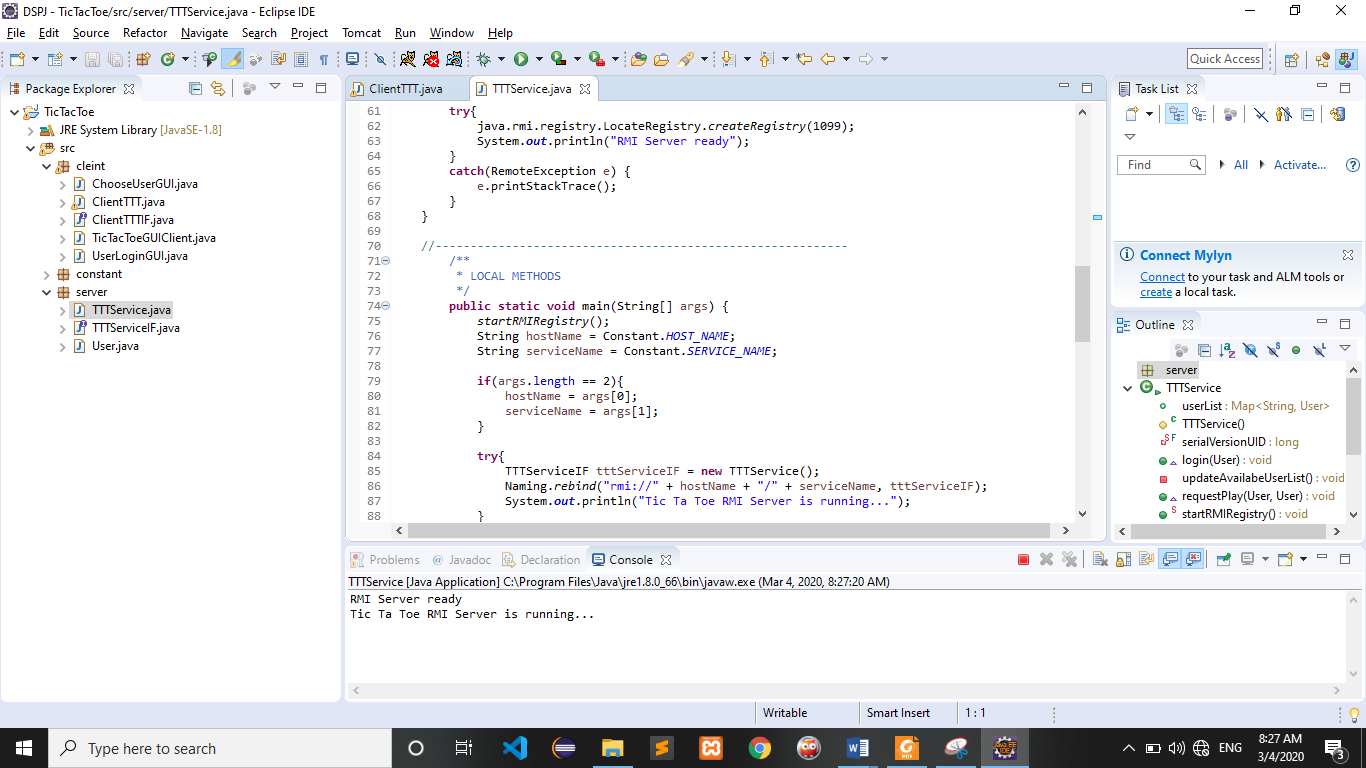


Figure 3 : Starting the Server From eclipse

## **Starting the Client**

Once the registry and the compute engine are running, you can start the client, specifying the following:

* The location where the client servers its classes (the Client class) by using the java.rmi.server.codebase property
* The java.security.policy property, which is used to specify the security policy file that contains the permissions you intend to grand to various pieces of code start the client on another host as follows:

First, user must input name to login tic tac toe multiplayer game application. Name is username. Figure 4 shows Tic Tac Toe Game Client Side in User View.

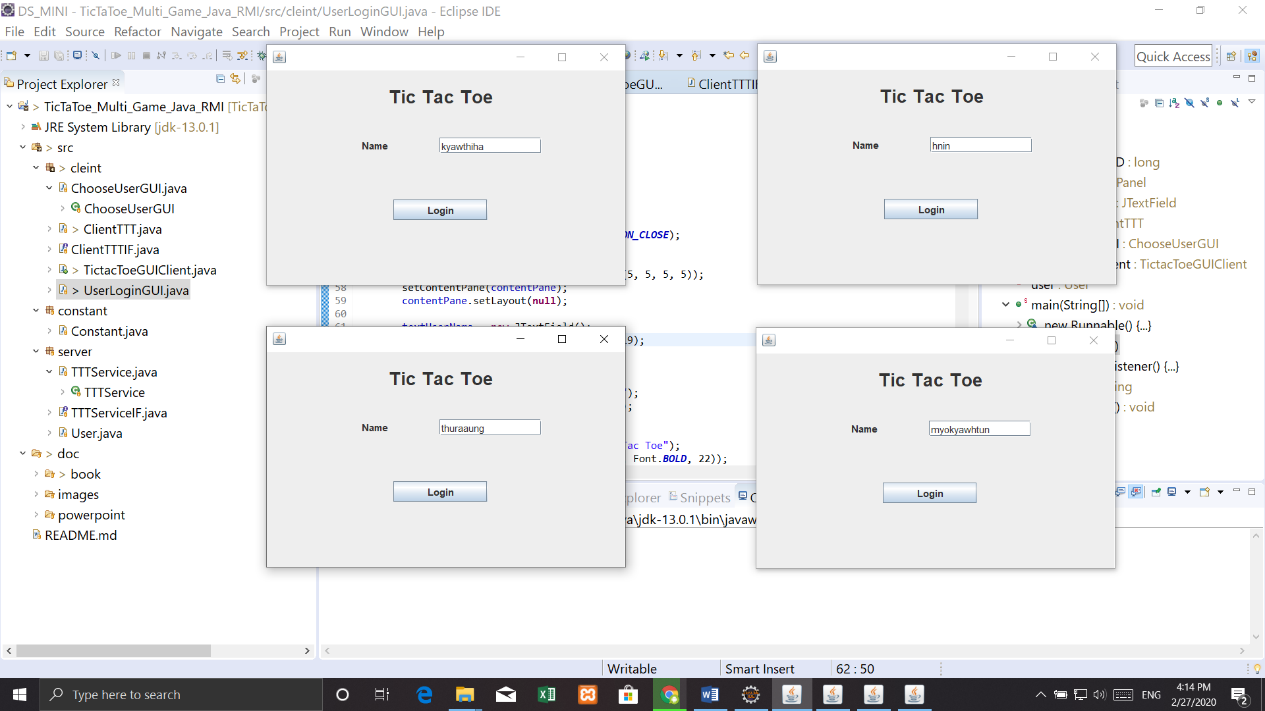


Figure 4 : Tic Tac Toe Game Client Side in User View

If you enter username, login to the server. After client sent a user object to the server side, the server side receives these values. And it check these values by using login function. Finally, it returns the user view to the client as shown in Figure 5.

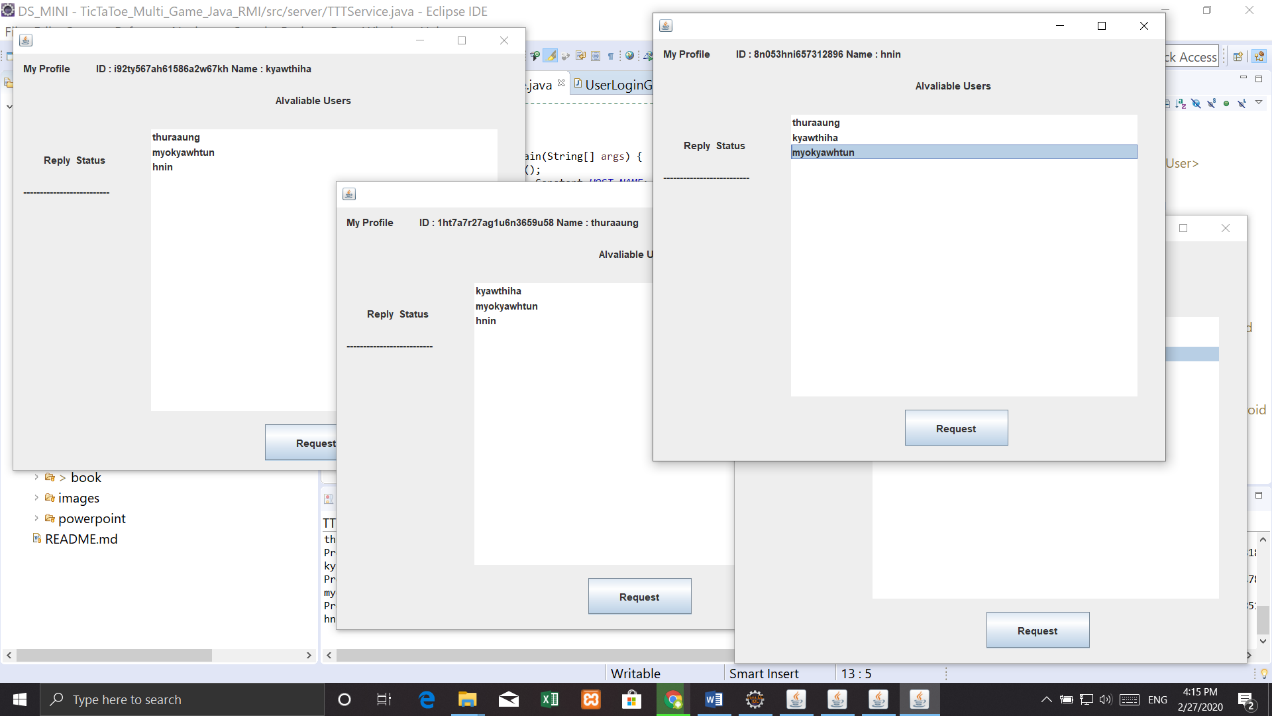


Figure 5 : Available User List in User View

If user choose another user in available user list and request another user, this request dialog will show in another users. This figure as shown in Figure 6.

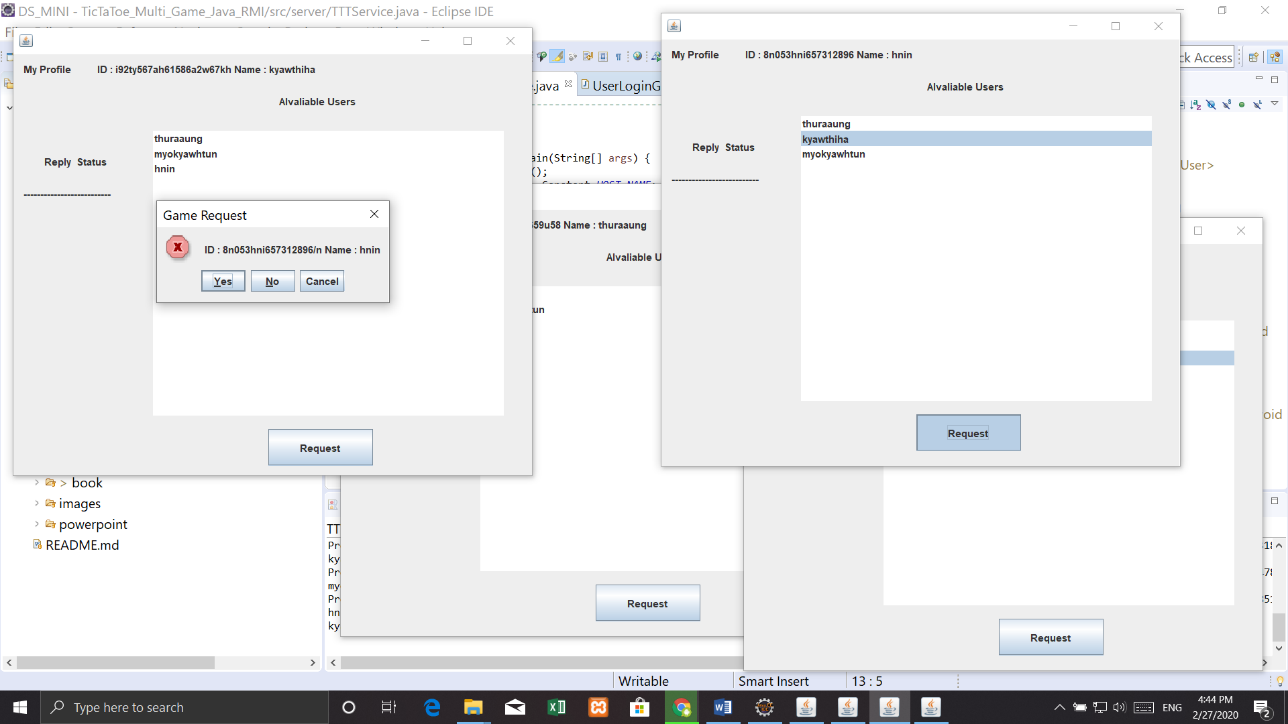


Figure 6 : User requests other user

If user accepts another user’s request, the game is played. This figure as shown in Figure 7 and Figure 8.

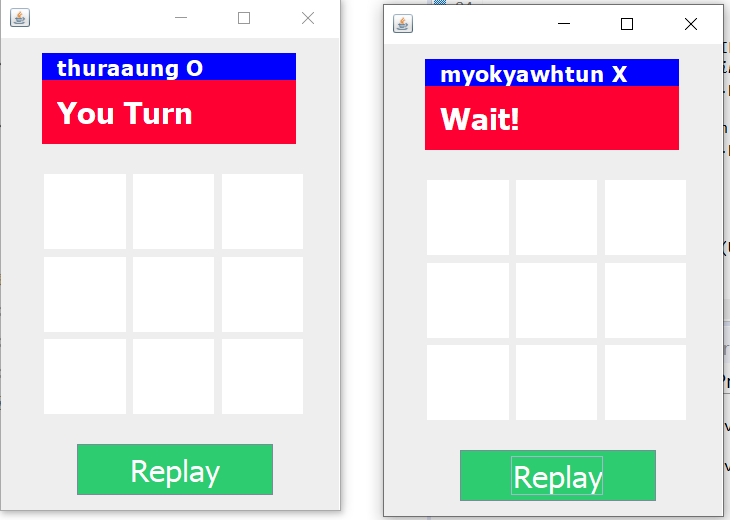


Figure 7 : Game is played ready

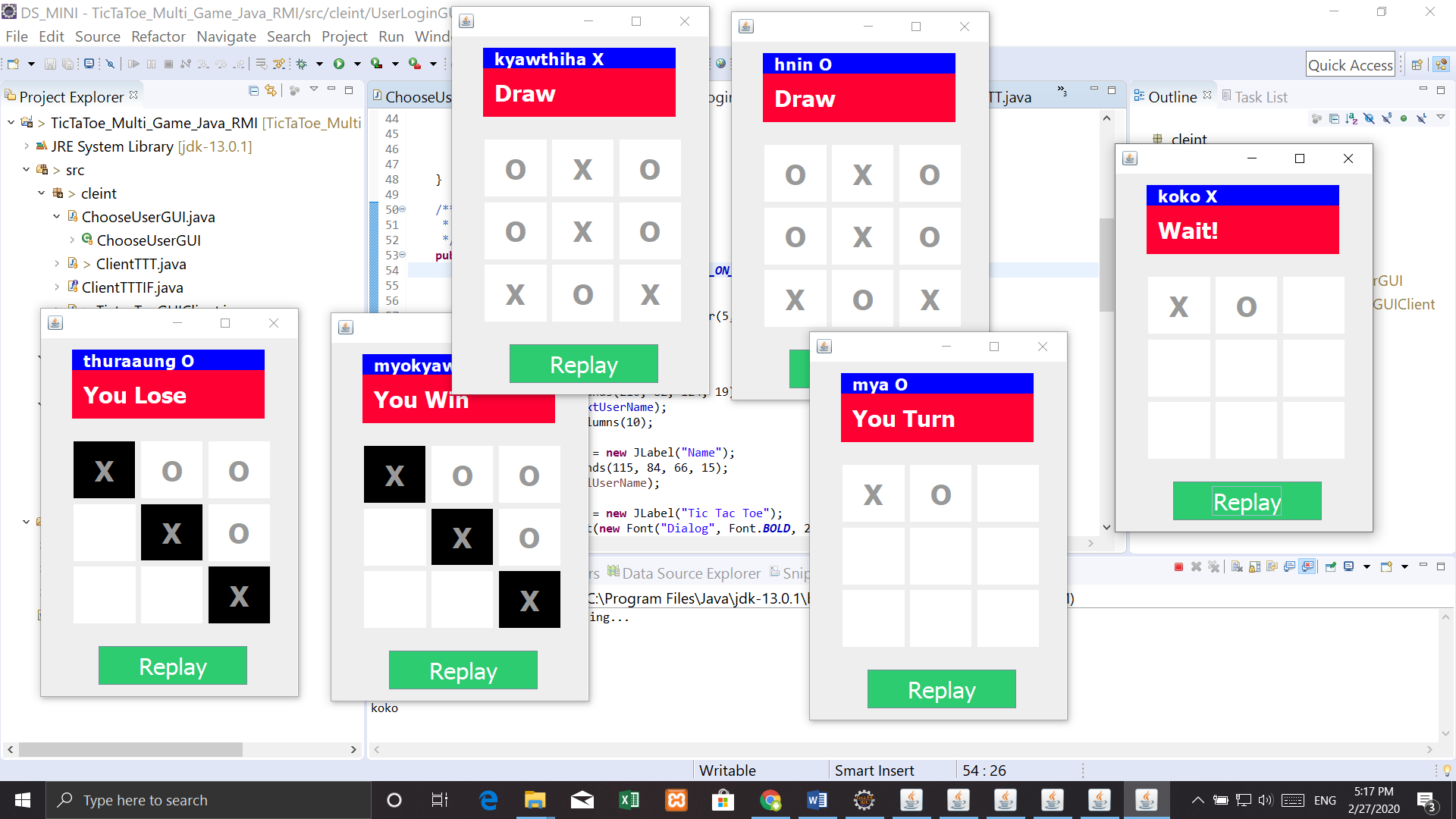


Figure 8 : Some user win, lose and draw

If user request replay game, this request dialog will show in another users. This figure as shown in Figure 9.

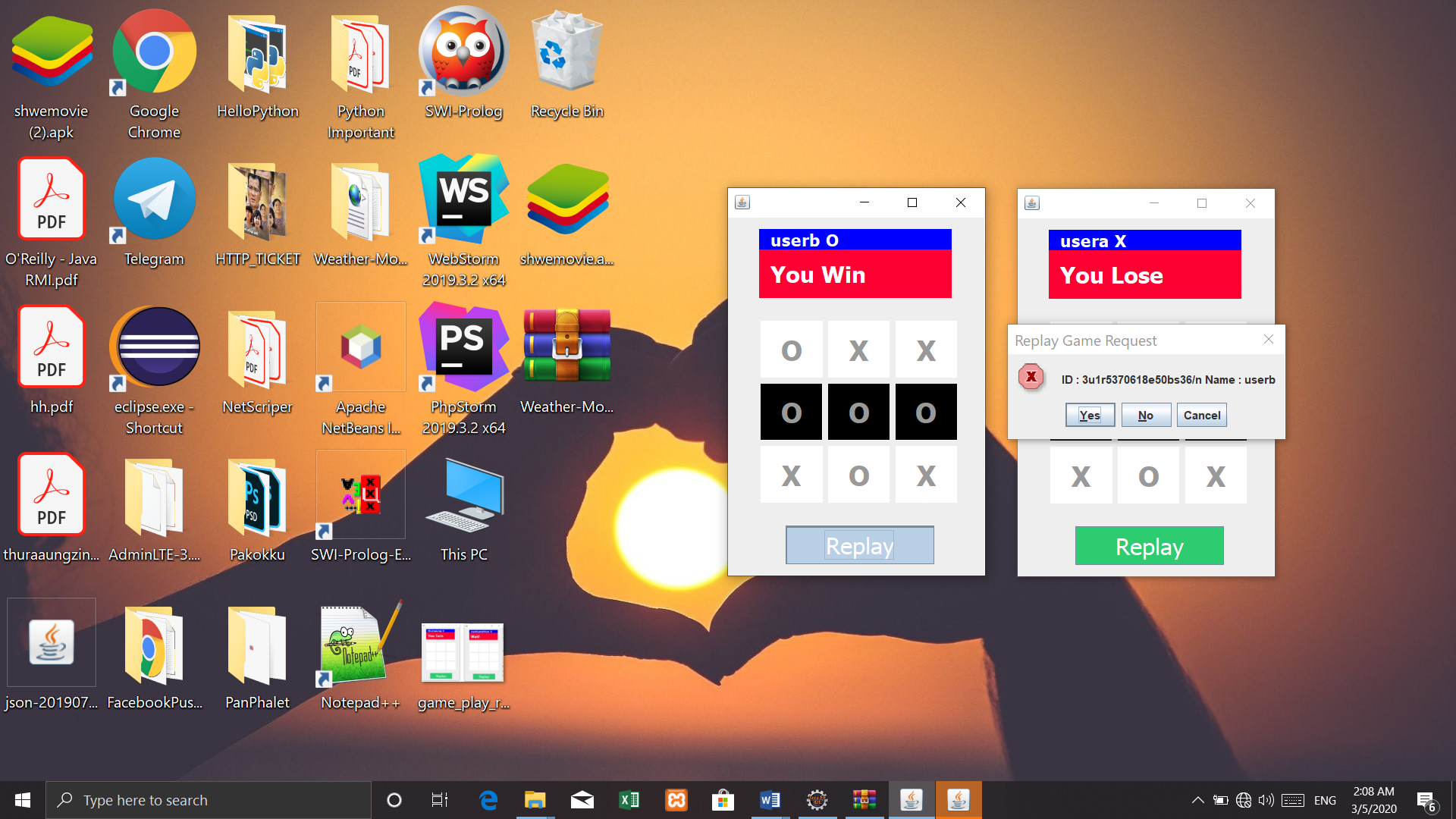


Figure 9 : User request another user to replay game

**Chapter (4)**

**Appendix**

### Constant.java

**package** constant;

**public** **class** Constant {

**public** **static** String *HOST\_NAME* = "localhost";

**public** **static** String *SERVICE\_NAME* = "TTTGameServer";

}

### User.java

**package** server;

**import** java.io.Serializable;

**import** cleint.ClientTTTIF;

**public** **class** User **implements** Serializable {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** String userid;

**private** String userName;

**private** String userHost;

**private** ClientTTTIF userClient;

**private** **boolean** availableStatus = **true**;

**public** **boolean** isAvailableStatus() {

**return** availableStatus;

}

**public** **void** setAvailableStatus(**boolean** availableStatus) {

**this**.availableStatus = availableStatus;

}

**public** **static** **long** getSerialversionuid() {

**return** ***serialVersionUID***;

}

**public** String getUserid() {

**return** userid;

}

**public** **void** setUserid(String userid) {

**this**.userid = userid;

}

**public** String getUserName() {

**return** userName;

}

**public** **void** setUserName(String userName) {

**this**.userName = userName;

}

**public** String getUserHost() {

**return** userHost;

}

**public** **void** setUserHost(String userHost) {

**this**.userHost = userHost;

}

**public** String getUserRemoteObjName() {

**return** "rmi://" + userHost + "/" + userid;

}

**public** ClientTTTIF getUserClient() {

**return** userClient;

}

**public** **void** setUserClient(ClientTTTIF userClient) {

**this**.userClient = userClient;

}

@Override

**public** String toString() {

**return** userName;

}

}

### TTTServiceIF.java

**package** server;

**import** java.rmi.\*;

**public** **interface** TTTServiceIF **extends** Remote {

**public** **void** login(User user) **throws** RemoteException;

**public** **void** requestPlay(User user, User otheruser) **throws** RemoteException;

**public** **void** acceptPlay(User user, User otheruser, **boolean** status) **throws** RemoteException;

**public** **void** playOnOff(User user, User otheruser, **int** pos) **throws** RemoteException;

**public** **void** leaveGame(User user) **throws** RemoteException;

**public** **void** requestRePlay(User user, User otheruser) **throws** RemoteException;

**public** **void** acceptRePlay(User user, User otheruser, **boolean** status) **throws** RemoteException;

}

### TTTService.java

**package** server;

**import** java.rmi.Naming;

**import** java.rmi.RemoteException;

**import** java.rmi.server.UnicastRemoteObject;

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Map;

**import** constant.Constant;

**public** **class** TTTService **extends** UnicastRemoteObject **implements** TTTServiceIF{

**public** Map<String, User> userList;

**protected** TTTService() **throws** RemoteException {

**super**();

userList = **new** HashMap<String, User>();

}

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Override

**public** **void** login(User user) **throws** RemoteException {

userList.put(user.getUserid(), user);

System.***out***.println(userList.toString());

updateAvailabeUserList();

}

**private** **void** updateAvailabeUserList(){

userList.forEach((k,v)->{

System.***out***.println(v.getUserClient());

**try** {

**if**(v.isAvailableStatus()) {

v.getUserClient().updateAvailableUserList(**new** ArrayList<User>(userList.values()));

System.***out***.println(v.getUserName());

}

} **catch** (RemoteException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

});

}

@Override

**public** **void** requestPlay(User user,User otheruser) **throws** RemoteException {

System.***out***.println(otheruser);

otheruser.getUserClient().request(user);

}

/\*\*

\* Start the RMI Registry

\*/

**public** **static** **void** startRMIRegistry() {

**try**{

java.rmi.registry.LocateRegistry.*createRegistry*(1099);

System.***out***.println("RMI Server ready");

}

**catch**(RemoteException e) {

e.printStackTrace();

}

}

//-----------------------------------------------------------

/\*\*

\* LOCAL METHODS

\*/

**public** **static** **void** main(String[] args) {

*startRMIRegistry*();

String hostName = Constant.*HOST\_NAME*;

String serviceName = Constant.*SERVICE\_NAME*;

**if**(args.length == 2){

hostName = args[0];

serviceName = args[1];

}

**try**{

TTTServiceIF tttServiceIF = **new** TTTService();

Naming.*rebind*("rmi://" + hostName + "/" + serviceName, tttServiceIF);

System.***out***.println("Tic Ta Toe RMI Server is running...");

}

**catch**(Exception e){

System.***out***.println("Server had problems starting");

}

}

@Override

**public** **void** acceptPlay(User user, User otheruser, **boolean** status) **throws** RemoteException {

**if**(status) {

System.***out***.println("Game connect");

userList.remove(user.getUserid());

userList.remove(otheruser.getUserid());

**if**(!userList.isEmpty())

updateAvailabeUserList();

otheruser.getUserClient().accept(user, **true**,**true**);

user.getUserClient().accept(otheruser, **true**,**false**);

}**else** {

otheruser.getUserClient().accept(user, **false**,**true**);

}

}

@Override

**public** **void** playOnOff(User user, User otheruser,**int** pos) **throws** RemoteException {

otheruser.getUserClient().playOnOFF(otheruser,pos);

}

@Override

**public** **void** leaveGame(User user) **throws** RemoteException {

userList.remove(user.getUserid());

**if**(!userList.isEmpty())

updateAvailabeUserList();

}

@Override

**public** **void** requestRePlay(User user, User otheruser) **throws** RemoteException {

otheruser.getUserClient().requestRePlay(user);

}

@Override

**public** **void** acceptRePlay(User user, User otheruser, **boolean** status) **throws** RemoteException {

**if**(status) {

System.***out***.println("Game connect");

otheruser.getUserClient().acceptRePlay(user, **true**,**true**);

user.getUserClient().acceptRePlay(otheruser, **true**,**false**);

}**else** {

otheruser.getUserClient().acceptRePlay(user, **false**,**true**);

user.getUserClient().acceptRePlay(otheruser, **false**,**true**);

}

}

}

### ClinetTTTIF.java

**package** cleint;

**import** java.rmi.\*;

**import** java.util.ArrayList;

**import** server.User;

**public** **interface** ClientTTTIF **extends** Remote {

**public** **void** playStatus(**boolean** status) **throws** RemoteException;

**public** **void** updateAvailableUserList(ArrayList<User> users) **throws** RemoteException;

**public** **void** request(User otherUser) **throws** RemoteException;

**public** **void** accept(User otherUser, **boolean** status, **boolean** xo) **throws** RemoteException;

**public** **void** playOnOFF(User otherUser, **int** pos) **throws** RemoteException;

**public** **void** requestRePlay(User otherUser) **throws** RemoteException;

**public** **void** acceptRePlay(User otherUser, **boolean** status, **boolean** xo) **throws** RemoteException;

}

### ClientTTT.java

**package** cleint;

**import** java.net.MalformedURLException;

**import** java.rmi.ConnectException;

**import** java.rmi.Naming;

**import** java.rmi.NotBoundException;

**import** java.rmi.RemoteException;

**import** java.rmi.server.UnicastRemoteObject;

**import** java.util.ArrayList;

**import** javax.swing.DefaultListModel;

**import** javax.swing.JOptionPane;

**import** constant.Constant;

**import** server.TTTServiceIF;

**import** server.User;

**public** **class** ClientTTT **extends** UnicastRemoteObject **implements** ClientTTTIF {

**public** UserLoginGUI userLoginGUI;

**public** ChooseUserGUI chooseUserGUI;

**public** TictacToeGUIClient tictaToeGUIClient;

**private** String hostName = Constant.*HOST\_NAME*;

**private** String serviceName = Constant.*SERVICE\_NAME*;

**protected** TTTServiceIF tttServiceIF;

**protected** User user;

**protected** **boolean** connectionProblem = **false**;

**protected** ClientTTT(UserLoginGUI userLoginGUI, ChooseUserGUI chooseUserGUI, TictacToeGUIClient tictaToeGUIClient,

User user) **throws** RemoteException {

**super**();

**this**.userLoginGUI = userLoginGUI;

**this**.chooseUserGUI = chooseUserGUI;

**this**.tictaToeGUIClient = tictaToeGUIClient;

**this**.user = user;

}

**protected** ClientTTT(ChooseUserGUI chooseUserGUI, User user) **throws** RemoteException {

**super**();

**this**.chooseUserGUI = chooseUserGUI;

**this**.user = user;

}

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

@Override

**public** **void** playStatus(**boolean** status) **throws** RemoteException {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** updateAvailableUserList(ArrayList<User> users) **throws** RemoteException {

DefaultListModel<User> listModel = **new** DefaultListModel<User>();

System.***out***.println("HELLO" + **this**.user + users.contains(**this**.user) + **this**.user.getUserid());

users.forEach(uu -> System.***out***.println(uu.getUserid()));

users.removeIf(obj -> obj.getUserid().equals(**this**.user.getUserid()));

**if** (users.size() > 0)

listModel.addAll(users);

System.***out***.println(listModel.toString());

**this**.chooseUserGUI.list.setModel(listModel);

}

/\*\*

\* Register our own listening service/interface lookup the server RMI interface,

\* then send our details

\*

\* **@throws** RemoteException

\*/

**public** **void** startClient() **throws** RemoteException {

**try** {

Naming.*rebind*(user.getUserRemoteObjName(), **this**);

tttServiceIF = (TTTServiceIF) Naming.*lookup*("rmi://" + hostName + "/" + serviceName);

} **catch** (ConnectException e) {

JOptionPane.*showMessageDialog*(userLoginGUI, "The server seems to be unavailable\nPlease try later",

"Connection problem", JOptionPane.***ERROR\_MESSAGE***);

connectionProblem = **true**;

e.printStackTrace();

} **catch** (NotBoundException | MalformedURLException me) {

connectionProblem = **true**;

me.printStackTrace();

}

**if** (!connectionProblem) {

// registerWithServer(details);

}

System.***out***.println("Client Listen RMI Server is running...\n");

}

@Override

**public** **void** request(User otherUser) **throws** RemoteException {

**int** result = JOptionPane.*showConfirmDialog*(chooseUserGUI,

"ID : " + otherUser.getUserid() + "/n Name : " + otherUser.getUserName(), "Game Request",

JOptionPane.***INFORMATION\_MESSAGE***, JOptionPane.***YES\_NO\_OPTION***);

**if** (result == JOptionPane.***YES\_OPTION***) {

tttServiceIF.acceptPlay(user, otherUser, **true**);

} **else** {

tttServiceIF.acceptPlay(user, otherUser, **false**);

}

}

@Override

**public** **void** accept(User otherUser, **boolean** status, **boolean** xo) **throws** RemoteException {

**if** (status) {

tictaToeGUIClient.setOtheruser(otherUser);

tictaToeGUIClient.setPlayerstatus(xo);

tictaToeGUIClient.playOnOff(xo);

tictaToeGUIClient.getjLabelUser().setText(user.getUserName() + (xo ? " O" : " X"));

**if** (xo) {

tictaToeGUIClient.getjLabelMSG().setText("You Turn");

} **else** {

tictaToeGUIClient.getjLabelMSG().setText("Wait!");

}

chooseUserGUI.setVisible(**false**);

tictaToeGUIClient.setVisible(**true**);

} **else** {

chooseUserGUI.lblReplyStatus.setText(otherUser.getUserName() + " Deny Request");

System.***out***.println("Deny");

}

}

@Override

**public** **void** playOnOFF(User otherUser, **int** pos) **throws** RemoteException {

tictaToeGUIClient.getjLabelMSG().setText("You Turn");

tictaToeGUIClient.playOnOff(**true**);

tictaToeGUIClient.anotherUserClick(pos);

}

@Override

**public** **void** requestRePlay(User otherUser) **throws** RemoteException {

**int** result = JOptionPane.*showConfirmDialog*(tictaToeGUIClient,

"ID : " + otherUser.getUserid() + "/n Name : " + otherUser.getUserName(), "Replay Game Request",

JOptionPane.***INFORMATION\_MESSAGE***, JOptionPane.***YES\_NO\_OPTION***);

**if** (result == JOptionPane.***YES\_OPTION***) {

tttServiceIF.acceptRePlay(user, otherUser, **true**);

} **else** {

tttServiceIF.acceptRePlay(user, otherUser, **false**);

}

}

@Override

**public** **void** acceptRePlay(User otherUser, **boolean** status, **boolean** xo) **throws** RemoteException {

**if** (status) {

tictaToeGUIClient.clearButtonAlll();

tictaToeGUIClient.setOtheruser(otherUser);

tictaToeGUIClient.setPlayerstatus(xo);

tictaToeGUIClient.playOnOff(xo);

tictaToeGUIClient.getjLabelUser().setText(user.getUserName() + (xo ? " O" : " X"));

**if** (xo) {

tictaToeGUIClient.getjLabelMSG().setText("You Turn");

} **else** {

tictaToeGUIClient.getjLabelMSG().setText("Wait!");

}

} **else** {

tictaToeGUIClient.playOnOff(**false**);

tictaToeGUIClient.getjLabelMSG().setText("Deny Try Again");

System.***out***.println("Deny");

}

}

}

### UserLoginGUI.java

**package** cleint;

**import** java.awt.EventQueue;

**import** javax.swing.JFrame;

**import** javax.swing.JPanel;

**import** javax.swing.border.EmptyBorder;

**import** server.User;

**import** javax.swing.JTextField;

**import** javax.swing.JLabel;

**import** java.awt.Font;

**import** javax.swing.JButton;

**import** java.awt.event.ActionListener;

**import** java.rmi.RemoteException;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.awt.event.ActionEvent;

**public** **class** UserLoginGUI **extends** JFrame {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = -8490581488485793046L;

**private** JPanel contentPane;

**private** JTextField textUserName;

**private** ClientTTT clientTTT;

**private** ChooseUserGUI chooseUserGUI;

**private** TictacToeGUIClient tictaToeGUIClient;

**private** User user;

/\*\*

\* Launch the application.

\*/

**public** **static** **void** main(String[] args) {

EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

**try** {

UserLoginGUI frame = **new** UserLoginGUI();

frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

}

});

}

/\*\*

\* Create the frame.

\*/

**public** UserLoginGUI() {

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setBounds(100, 100, 450, 300);

contentPane = **new** JPanel();

contentPane.setBorder(**new** EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(**null**);

textUserName = **new** JTextField();

textUserName.setBounds(210, 82, 124, 19);

contentPane.add(textUserName);

textUserName.setColumns(10);

JLabel lblUserName = **new** JLabel("Name");

lblUserName.setBounds(115, 84, 66, 15);

contentPane.add(lblUserName);

JLabel lblNewLabel = **new** JLabel("Tic Tac Toe");

lblNewLabel.setFont(**new** Font("Dialog", Font.***BOLD***, 22));

lblNewLabel.setBounds(148, 12, 147, 38);

contentPane.add(lblNewLabel);

JButton btnLogin = **new** JButton("Login");

btnLogin.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent arg0) {

**try** {

getConnected();

setVisible(**false**);

chooseUserGUI.setVisible(**true**);

clientTTT.tttServiceIF.login(user);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

btnLogin.setBounds(154, 157, 114, 25);

contentPane.add(btnLogin);

}

**private** String getUUID() {

ArrayList<String> chars = **new** ArrayList<String>();

String rawString = System.*currentTimeMillis*() + **this**.user.getUserName();

**for** (**int** i = 0; i < rawString.length(); i++) {

chars.add(rawString.charAt(i) + "");

}

Collections.*shuffle*(chars);

StringBuilder sb = **new** StringBuilder();

**for** (String a : chars) {

sb.append(a);

}

**return** sb.toString();

}

/\*\*

\* Make the connection to the chat server

\*

\* **@param** userName

\* **@throws** RemoteException

\*/

**private** **void** getConnected() **throws** RemoteException {

// remove whitespace and non word characters to avoid malformed url

**this**.user = **new** User();

**this**.user.setUserHost("localhost");

**this**.user.setUserName(textUserName.getText());

**this**.user.setUserid(getUUID());

**try** {

**this**.chooseUserGUI = **new** ChooseUserGUI(user);

tictaToeGUIClient = **new** TictacToeGUIClient();

clientTTT = **new** ClientTTT(**this**, chooseUserGUI, tictaToeGUIClient, user);

clientTTT.startClient();

**this**.chooseUserGUI.setClientTTT(clientTTT);

**this**.user.setUserClient(clientTTT);

tictaToeGUIClient.setClientTTT(clientTTT);

tictaToeGUIClient.setUser(user);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

}

### ChooseUserGUI.java

**package** cleint;

**import** javax.swing.JButton;

**import** javax.swing.JFrame;

**import** javax.swing.JLabel;

**import** javax.swing.JList;

**import** javax.swing.JOptionPane;

**import** javax.swing.JPanel;

**import** javax.swing.border.EmptyBorder;

**import** server.User;

**import** java.awt.event.ActionListener;

**import** java.rmi.RemoteException;

**import** java.awt.event.ActionEvent;

**public** **class** ChooseUserGUI **extends** JFrame {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** JPanel contentPane;

**public** JList<User> list;

**private** ClientTTT clientTTT;

**private** User user;

**public** JLabel lblMyProfile;

**private** JLabel lblMyProfileUser;

**public** JLabel lblReplyStatus;

**public** ClientTTT getClientTTT() {

**return** clientTTT;

}

**public** **void** setClientTTT(ClientTTT clientTTT) {

**this**.clientTTT = clientTTT;

}

**public** **void** showDenyMessage() {

JOptionPane.*showMessageDialog*(**null**, "Deny", "Deny Request", JOptionPane.***INFORMATION\_MESSAGE***);

}

/\*\*

\* Create the frame.

\*

\* **@throws** RemoteException

\*/

**public** ChooseUserGUI(User user) **throws** RemoteException {

**this**.setUser(user);

setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

setBounds(100, 100, 633, 548);

contentPane = **new** JPanel();

contentPane.setBorder(**new** EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);

contentPane.setLayout(**null**);

list = **new** JList<User>();

list.setBounds(167, 91, 418, 340);

contentPane.add(list);

JLabel lblAlvaliableUsers = **new** JLabel("Alvaliable Users");

lblAlvaliableUsers.setBounds(317, 43, 134, 27);

contentPane.add(lblAlvaliableUsers);

JButton btnRequest = **new** JButton("Request");

btnRequest.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent arg0) {

System.***out***.println(list.getSelectedValue().getUserName());

**try** {

clientTTT.tttServiceIF.requestPlay(user, list.getSelectedValue());

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

addWindowListener(**new** java.awt.event.WindowAdapter() {

@Override

**public** **void** windowClosing(java.awt.event.WindowEvent windowEvent) {

System.***out***.println("HELLO");

**try** {

clientTTT.tttServiceIF.leaveGame(user);

} **catch** (RemoteException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

System.*exit*(0);

}

});

btnRequest.setBounds(304, 447, 125, 44);

contentPane.add(btnRequest);

lblMyProfile = **new** JLabel("My Profile");

lblMyProfile.setBounds(12, 7, 100, 20);

contentPane.add(lblMyProfile);

lblMyProfileUser = **new** JLabel("ID");

lblMyProfileUser.setBounds(100, 4, 453, 27);

contentPane.add(lblMyProfileUser);

lblMyProfileUser.setText("ID : " + user.getUserid() + " Name : " + user.getUserName());

lblReplyStatus = **new** JLabel("--------------------------");

lblReplyStatus.setBounds(12, 149, 153, 35);

contentPane.add(lblReplyStatus);

JLabel label = **new** JLabel("Reply Status");

label.setBounds(37, 111, 95, 35);

contentPane.add(label);

}

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

}

### TicTacToeGUIClient.java

**package** cleint;

**import** java.awt.Color;

**import** java.awt.Component;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.ActionListener;

**import** java.rmi.RemoteException;

**import** javax.swing.JButton;

**import** server.User;

/\*\*

\*

\* **@author** 1bestcsharp.blogspot.com

\*/

**public** **class** TicTacToeGUIClient **extends** javax.swing.JFrame {

/\*\*

\*

\*/

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

**private** **boolean** playerstatus;

**private** ClientTTT clientTTT;

**private** User otheruser;

**private** User user;

**public** User getOtheruser() {

**return** otheruser;

}

**public** **void** setOtheruser(User otheruser) {

**this**.otheruser = otheruser;

}

**public** User getUser() {

**return** user;

}

**public** **void** setUser(User user) {

**this**.user = user;

}

**public** ClientTTT getClientTTT() {

**return** clientTTT;

}

**public** **void** setClientTTT(ClientTTT clientTTT) {

**this**.clientTTT = clientTTT;

}

**public** **boolean** isPlayerstatus() {

**return** playerstatus;

}

**public** **void** setPlayerstatus(**boolean** playerstatus) {

**this**.playerstatus = playerstatus;

}

/\*\*

\* Creates new form Java\_tic\_tac\_toe

\*/

**public** TicTacToeGUIClient() {

initComponents();

addWindowListener(**new** java.awt.event.WindowAdapter() {

@Override

**public** **void** windowClosing(java.awt.event.WindowEvent windowEvent) {

System.***out***.println("HELLO");

**try** {

clientTTT.tttServiceIF.leaveGame(user);

} **catch** (RemoteException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

System.*exit*(0);

}

});

addEachButtonAction();

}

**public** **void** addEachButtonAction() {

jButton1.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

setButtonClick(jButton1);

disabledAction();

**try** {

playOnOff(**false**);

clientTTT.tttServiceIF.playOnOff(user, otheruser, 1);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

jButton2.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

setButtonClick(jButton2);

disabledAction();

**try** {

playOnOff(**false**);

clientTTT.tttServiceIF.playOnOff(user, otheruser, 2);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

jButton3.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

setButtonClick(jButton3);

disabledAction();

**try** {

playOnOff(**false**);

clientTTT.tttServiceIF.playOnOff(user, otheruser, 3);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

jButton4.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

setButtonClick(jButton4);

disabledAction();

**try** {

playOnOff(**false**);

clientTTT.tttServiceIF.playOnOff(user, otheruser, 4);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

jButton5.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

setButtonClick(jButton5);

disabledAction();

**try** {

playOnOff(**false**);

clientTTT.tttServiceIF.playOnOff(user, otheruser, 5);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

jButton6.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

setButtonClick(jButton6);

disabledAction();

**try** {

playOnOff(**false**);

clientTTT.tttServiceIF.playOnOff(user, otheruser, 6);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

jButton7.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

setButtonClick(jButton7);

disabledAction();

**try** {

playOnOff(**false**);

clientTTT.tttServiceIF.playOnOff(user, otheruser, 7);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

jButton8.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

setButtonClick(jButton8);

disabledAction();

**try** {

playOnOff(**false**);

clientTTT.tttServiceIF.playOnOff(user, otheruser, 8);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

jButton9.addActionListener(**new** ActionListener() {

@Override

**public** **void** actionPerformed(ActionEvent arg0) {

setButtonClick(jButton9);

disabledAction();

**try** {

playOnOff(**false**);

clientTTT.tttServiceIF.playOnOff(user, otheruser, 9);

} **catch** (RemoteException e) {

e.printStackTrace();

}

}

});

}

**public** **void** anotherUserClick(**int** pos) {

**switch** (pos) {

**case** 1:

setOtherButtonClick(jButton1);

**break**;

**case** 2:

setOtherButtonClick(jButton2);

**break**;

**case** 3:

setOtherButtonClick(jButton3);

**break**;

**case** 4:

setOtherButtonClick(jButton4);

**break**;

**case** 5:

setOtherButtonClick(jButton5);

**break**;

**case** 6:

setOtherButtonClick(jButton6);

**break**;

**case** 7:

setOtherButtonClick(jButton7);

**break**;

**case** 8:

setOtherButtonClick(jButton8);

**break**;

**default**:

setOtherButtonClick(jButton9);

**break**;

}

disabledAction();

}

**public** **void** setOtherButtonClick(JButton button) {

**if** (!playerstatus) {

button.setText("O");

button.setForeground(Color.***RED***);

} **else** {

button.setText("X");

button.setForeground(Color.***BLUE***);

}

getTheWinner(**false**);

}

**public** **void** setButtonClick(JButton button) {

**if** (playerstatus) {

button.setText("O");

button.setForeground(Color.***RED***);

} **else** {

button.setText("X");

button.setForeground(Color.***BLUE***);

}

jLabelMSG.setText("Wait!");

getTheWinner(**true**);

}

**public** **void** disabledAction() {

Component[] comps = jPanel1.getComponents();

**for** (Component comp : comps) {

**if** (comp **instanceof** JButton) {

JButton button = (JButton) comp;

**if** (!button.getText().equals(""))

button.setEnabled(**false**);

}

}

}

**public** **void** winEffect(JButton b1, JButton b2, JButton b3, **boolean** status) {

b1.setBackground(Color.***BLACK***);

b2.setBackground(Color.***BLACK***);

b3.setBackground(Color.***BLACK***);

b1.setForeground(Color.***WHITE***);

b2.setForeground(Color.***WHITE***);

b3.setForeground(Color.***WHITE***);

**if** (status) {

jLabelMSG.setText("You Win");

} **else** {

jLabelMSG.setText("You Lose");

}

playOnOff(**false**);

}

**boolean** win = **false**;

**public** **void** getTheWinner(**boolean** status) {

**if** (!jButton1.getText().equals("") && jButton1.getText().equals(jButton2.getText())

&& jButton1.getText().equals(jButton3.getText())) {

winEffect(jButton1, jButton2, jButton3, status);

win = **true**;

}

**if** (!jButton4.getText().equals("") && jButton4.getText().equals(jButton5.getText())

&& jButton4.getText().equals(jButton6.getText())) {

winEffect(jButton4, jButton5, jButton6, status);

win = **true**;

}

**if** (!jButton7.getText().equals("") && jButton7.getText().equals(jButton8.getText())

&& jButton7.getText().equals(jButton9.getText())) {

winEffect(jButton7, jButton8, jButton9, status);

win = **true**;

}

**if** (!jButton1.getText().equals("") && jButton1.getText().equals(jButton4.getText())

&& jButton1.getText().equals(jButton7.getText())) {

winEffect(jButton1, jButton4, jButton7, status);

win = **true**;

}

**if** (!jButton2.getText().equals("") && jButton2.getText().equals(jButton5.getText())

&& jButton2.getText().equals(jButton8.getText())) {

winEffect(jButton2, jButton5, jButton8, status);

win = **true**;

}

**if** (!jButton3.getText().equals("") && jButton3.getText().equals(jButton6.getText())

&& jButton3.getText().equals(jButton9.getText())) {

winEffect(jButton3, jButton6, jButton9, status);

win = **true**;

}

**if** (!jButton1.getText().equals("") && jButton1.getText().equals(jButton5.getText())

&& jButton1.getText().equals(jButton9.getText())) {

winEffect(jButton1, jButton5, jButton9, status);

win = **true**;

}

**if** (!jButton3.getText().equals("") && jButton3.getText().equals(jButton5.getText())

&& jButton3.getText().equals(jButton7.getText())) {

winEffect(jButton3, jButton5, jButton7, status);

win = **true**;

}

**else** **if** (allButtonsTextLength() == 9 && win == **false**) {

jLabelMSG.setText("Draw");

}

}

// if no one win

**public** **int** allButtonsTextLength() {

// all buttons text length mean if

// all buttons text length = 9

// all buttons with text on them

String txt = "";

Component[] comps = jPanel1.getComponents();

**for** (Component comp : comps) {

**if** (comp **instanceof** JButton) {

JButton button = (JButton) comp;

txt = txt + button.getText();

}

}

**return** txt.length();

}

// create action

// x\_or\_o => to display X or O

**int** x\_or\_o = 0;

**public** **void** playOnOff(**boolean** status) {

Component[] comps = jPanel1.getComponents();

**for** (Component comp : comps) {

**if** (comp **instanceof** JButton) {

JButton button = (JButton) comp;

**if** (!status)

button.setEnabled(**false**);

**else** {

**if** (button.getText().equals("")) {

button.setEnabled(**true**);

}

}

}

}

}

**private** **void** initComponents() {

jPanel1 = **new** javax.swing.JPanel();

jButton1 = **new** javax.swing.JButton();

jButton2 = **new** javax.swing.JButton();

jButton3 = **new** javax.swing.JButton();

jButton4 = **new** javax.swing.JButton();

jButton5 = **new** javax.swing.JButton();

jButton6 = **new** javax.swing.JButton();

jButton7 = **new** javax.swing.JButton();

jButton8 = **new** javax.swing.JButton();

jButton9 = **new** javax.swing.JButton();

jPanel2 = **new** javax.swing.JPanel();

jPanel3 = **new** javax.swing.JPanel();

jLabelMSG = **new** javax.swing.JLabel();

jLabelUser = **new** javax.swing.JLabel();

jButtonReplay = **new** javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.***EXIT\_ON\_CLOSE***);

jButton1.setBackground(**new** java.awt.Color(255, 255, 255));

jButton1.setFont(**new** java.awt.Font("Tahoma", 1, 30)); // NOI18N

jButton1.setBorder(**null**);

jButton2.setBackground(**new** java.awt.Color(255, 255, 255));

jButton2.setFont(**new** java.awt.Font("Tahoma", 1, 30)); // NOI18N

jButton2.setBorder(**null**);

jButton3.setBackground(**new** java.awt.Color(255, 255, 255));

jButton3.setFont(**new** java.awt.Font("Tahoma", 1, 30)); // NOI18N

jButton3.setBorder(**null**);

jButton4.setBackground(**new** java.awt.Color(255, 255, 255));

jButton4.setFont(**new** java.awt.Font("Tahoma", 1, 30)); // NOI18N

jButton4.setBorder(**null**);

jButton5.setBackground(**new** java.awt.Color(255, 255, 255));

jButton5.setFont(**new** java.awt.Font("Tahoma", 1, 30)); // NOI18N

jButton5.setBorder(**null**);

jButton6.setBackground(**new** java.awt.Color(255, 255, 255));

jButton6.setFont(**new** java.awt.Font("Tahoma", 1, 30)); // NOI18N

jButton6.setBorder(**null**);

jButton7.setBackground(**new** java.awt.Color(255, 255, 255));

jButton7.setFont(**new** java.awt.Font("Tahoma", 1, 30)); // NOI18N

jButton7.setBorder(**null**);

jButton8.setBackground(**new** java.awt.Color(255, 255, 255));

jButton8.setFont(**new** java.awt.Font("Tahoma", 1, 30)); // NOI18N

jButton8.setBorder(**null**);

jButton9.setBackground(**new** java.awt.Color(255, 255, 255));

jButton9.setFont(**new** java.awt.Font("Tahoma", 1, 30)); // NOI18N

jButton9.setBorder(**null**);

javax.swing.GroupLayout jPanel1Layout = **new** javax.swing.GroupLayout(jPanel1);

jPanel1.setLayout(jPanel1Layout);

jPanel1Layout.setHorizontalGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***LEADING***)

.addGroup(jPanel1Layout.createSequentialGroup().addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***LEADING***)

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jButton1, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 65,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.***RELATED***)

.addComponent(jButton2, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 65,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.***RELATED***)

.addComponent(jButton3, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 65,

javax.swing.GroupLayout.***PREFERRED\_SIZE***))

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jButton4, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 65,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.***RELATED***)

.addComponent(jButton5, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 65,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.***RELATED***)

.addComponent(jButton6, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 65,

javax.swing.GroupLayout.***PREFERRED\_SIZE***))

.addGroup(jPanel1Layout.createSequentialGroup()

.addComponent(jButton7, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 65,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.***RELATED***)

.addComponent(jButton8, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 65,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.***RELATED***)

.addComponent(jButton9, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 65,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)))

.addContainerGap(javax.swing.GroupLayout.***DEFAULT\_SIZE***, Short.***MAX\_VALUE***)));

jPanel1Layout.setVerticalGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***LEADING***)

.addGroup(jPanel1Layout.createSequentialGroup().addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***TRAILING***)

.addComponent(jButton3, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 60,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addComponent(jButton2, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 60,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addComponent(jButton1, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 60,

javax.swing.GroupLayout.***PREFERRED\_SIZE***))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.***RELATED***)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***TRAILING***)

.addComponent(jButton6, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 60,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addComponent(jButton5, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 60,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addComponent(jButton4, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 60,

javax.swing.GroupLayout.***PREFERRED\_SIZE***))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.***RELATED***)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***TRAILING***)

.addComponent(jButton9, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 60,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addComponent(jButton8, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 60,

javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addComponent(jButton7, javax.swing.GroupLayout.***PREFERRED\_SIZE***, 60,

javax.swing.GroupLayout.***PREFERRED\_SIZE***))

.addContainerGap(javax.swing.GroupLayout.***DEFAULT\_SIZE***, Short.***MAX\_VALUE***)));

jPanel2.setBackground(**new** java.awt.Color(255, 0, 51));

jLabelMSG.setFont(**new** java.awt.Font("Tahoma", 1, 24)); // NOI18N

jLabelMSG.setForeground(**new** java.awt.Color(255, 255, 255));

jLabelMSG.setText("Play");

jPanel3.setBackground(Color.***BLUE***);

jLabelUser.setFont(**new** java.awt.Font("Tahoma", 1, 18)); // NOI18N

jLabelUser.setForeground(**new** java.awt.Color(255, 255, 255));

jLabelUser.setText("HELLO USER");

javax.swing.GroupLayout jPanel3Layout = **new** javax.swing.GroupLayout(jPanel3);

jPanel3.setLayout(jPanel3Layout);

jPanel3Layout.setHorizontalGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***LEADING***)

.addGroup(jPanel3Layout.createSequentialGroup().addContainerGap().addComponent(jLabelUser,

javax.swing.GroupLayout.***DEFAULT\_SIZE***, 191, Short.***MAX\_VALUE***)));

jPanel3Layout.setVerticalGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***LEADING***)

.addComponent(jLabelUser, javax.swing.GroupLayout.Alignment.***TRAILING***,

javax.swing.GroupLayout.***DEFAULT\_SIZE***, 12, Short.***MAX\_VALUE***));

javax.swing.GroupLayout jPanel2Layout = **new** javax.swing.GroupLayout(jPanel2);

jPanel2.setLayout(jPanel2Layout);

jPanel2Layout.setHorizontalGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***LEADING***)

.addGroup(jPanel2Layout.createSequentialGroup().addContainerGap().addComponent(jLabelMSG,

javax.swing.GroupLayout.***DEFAULT\_SIZE***, 191, Short.***MAX\_VALUE***)));

jPanel2Layout.setVerticalGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***LEADING***)

.addComponent(jLabelMSG, javax.swing.GroupLayout.Alignment.***TRAILING***,

javax.swing.GroupLayout.***DEFAULT\_SIZE***, 51, Short.***MAX\_VALUE***));

jButtonReplay.setBackground(**new** java.awt.Color(46, 204, 113));

jButtonReplay.setFont(**new** java.awt.Font("Tahoma", 0, 24)); // NOI18N

jButtonReplay.setForeground(**new** java.awt.Color(255, 255, 255));

jButtonReplay.setText("Replay");

jButtonReplay.addActionListener(**new** java.awt.event.ActionListener() {

**public** **void** actionPerformed(java.awt.event.ActionEvent evt) {

**try** {

jButtonReplayActionPerformed(evt);

} **catch** (RemoteException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

});

javax.swing.GroupLayout layout = **new** javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***LEADING***)

.addGroup(javax.swing.GroupLayout.Alignment.***TRAILING***,

layout.createSequentialGroup().addContainerGap(23, Short.***MAX\_VALUE***)

.addComponent(jPanel1, javax.swing.GroupLayout.***PREFERRED\_SIZE***,

javax.swing.GroupLayout.***DEFAULT\_SIZE***, javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addGap(18, 18, 18))

.addGroup(layout.createSequentialGroup().addGroup(layout

.createParallelGroup(javax.swing.GroupLayout.Alignment.***LEADING***)

.addGroup(layout.createSequentialGroup().addGap(33, 33, 33).addComponent(jPanel3,

javax.swing.GroupLayout.***PREFERRED\_SIZE***, javax.swing.GroupLayout.***DEFAULT\_SIZE***,

javax.swing.GroupLayout.***PREFERRED\_SIZE***))

.addGroup(layout.createSequentialGroup().addGap(33, 33, 33).addComponent(jPanel2,

javax.swing.GroupLayout.***PREFERRED\_SIZE***, javax.swing.GroupLayout.***DEFAULT\_SIZE***,

javax.swing.GroupLayout.***PREFERRED\_SIZE***))

.addGroup(layout.createSequentialGroup().addGap(61, 61, 61).addComponent(jButtonReplay,

javax.swing.GroupLayout.***PREFERRED\_SIZE***, 157, javax.swing.GroupLayout.***PREFERRED\_SIZE***)))

.addContainerGap(javax.swing.GroupLayout.***DEFAULT\_SIZE***, Short.***MAX\_VALUE***)));

layout.setVerticalGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.***LEADING***)

.addGroup(layout.createSequentialGroup().addContainerGap()

.addComponent(jPanel3, javax.swing.GroupLayout.***PREFERRED\_SIZE***,

javax.swing.GroupLayout.***DEFAULT\_SIZE***, javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addComponent(jPanel2, javax.swing.GroupLayout.***PREFERRED\_SIZE***,

javax.swing.GroupLayout.***DEFAULT\_SIZE***, javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.***UNRELATED***)

.addComponent(jPanel1, javax.swing.GroupLayout.***PREFERRED\_SIZE***,

javax.swing.GroupLayout.***DEFAULT\_SIZE***, javax.swing.GroupLayout.***PREFERRED\_SIZE***)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.***UNRELATED***)

.addComponent(jButtonReplay, javax.swing.GroupLayout.***DEFAULT\_SIZE***, 41, Short.***MAX\_VALUE***)

.addContainerGap()));

pack();

}// </editor-fold>

**private** **void** jButtonReplayActionPerformed(java.awt.event.ActionEvent evt) **throws** RemoteException {

win = **false**;

clientTTT.tttServiceIF.requestRePlay(user, otheruser);

}

**public** **void** clearButtonAlll() {

Component[] comps = jPanel1.getComponents();

**for** (Component comp : comps) {

**if** (comp **instanceof** JButton) {

JButton button = (JButton) comp;

button.setText("");

button.setBackground(Color.***WHITE***);

}

}

}

/\*\*

\* **@param** args the command line arguments

\*/

**public** **static** **void** main(String args[]) {

/\* Set the Nimbus look and feel \*/

// <editor-fold defaultstate="collapsed" desc=" Look and feel setting code

// (optional) ">

/\*

\* If Nimbus (introduced in Java SE 6) is not available, stay with the default

\* look and feel. For details see

\* http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

**try** {

**for** (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.*getInstalledLookAndFeels*()) {

**if** ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.*setLookAndFeel*(info.getClassName());

**break**;

}

}

} **catch** (ClassNotFoundException ex) {

java.util.logging.Logger.*getLogger*(TicTacToeGUIClient.**class**.getName()).log(java.util.logging.Level.***SEVERE***,

**null**, ex);

} **catch** (InstantiationException ex) {

java.util.logging.Logger.*getLogger*(TicTacToeGUIClient.**class**.getName()).log(java.util.logging.Level.***SEVERE***,

**null**, ex);

} **catch** (IllegalAccessException ex) {

java.util.logging.Logger.*getLogger*(TicTacToeGUIClient.**class**.getName()).log(java.util.logging.Level.***SEVERE***,

**null**, ex);

} **catch** (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.*getLogger*(TicTacToeGUIClient.**class**.getName()).log(java.util.logging.Level.***SEVERE***,

**null**, ex);

}

// </editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.*invokeLater*(**new** Runnable() {

**public** **void** run() {

TicTacToeGUIClient tictaToeGUIClient = **new** TicTacToeGUIClient();

tictaToeGUIClient.setPlayerstatus(**false**);

tictaToeGUIClient.setVisible(**true**);

}

});

}

// Variables declaration - do not modify

**private** javax.swing.JButton jButton1;

**private** javax.swing.JButton jButton2;

**private** javax.swing.JButton jButton3;

**private** javax.swing.JButton jButton4;

**private** javax.swing.JButton jButton5;

**private** javax.swing.JButton jButton6;

**private** javax.swing.JButton jButton7;

**private** javax.swing.JButton jButton8;

**private** javax.swing.JButton jButton9;

**private** javax.swing.JButton jButtonReplay;

**private** javax.swing.JLabel jLabelMSG;

**private** javax.swing.JPanel jPanel1;

**private** javax.swing.JPanel jPanel2;

**private** javax.swing.JPanel jPanel3;

**private** javax.swing.JLabel jLabelUser;

**public** javax.swing.JLabel getjLabelMSG() {

**return** jLabelMSG;

}

**public** **void** setjLabelMSG(javax.swing.JLabel jLabelMSG) {

**this**.jLabelMSG = jLabelMSG;

}

**public** javax.swing.JLabel getjLabelUser() {

**return** jLabelUser;

}

**public** **void** setjLabelUser(javax.swing.JLabel jLabelUser) {

**this**.jLabelUser = jLabelUser;

}

// End of variables declaration

}