```
package rmiClient;

/**
 *
 * @author Patrick and Tenzin
 * @date 25 nov 2008
 * @version 1.0
 */



import java.io.*;
import java.applet.*;
import java.awt.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.*;
import java.rmi.*;



public class Client extends Applet
{

        private static final long serialVersionUID = 1L;
        private TextArea  display = new TextArea();
        private TextField input   = new TextField();
        private Button    send    = new Button("Send"), connect = new Button("Connect"),
                          quit    = new Button("Bye");
        private String    serverName;
        private String nickname = "";

        // new objects for supporting RMI functionallity
        ChatObjectInterface chat;
        rmiServer.ServerInterface server;


        /*
         * method for initializing the applet, it paints the window, sets the buttons
         *
         */
   public void init() {
      Panel keys = new Panel();
      keys.setLayout(new GridLayout(1,2));
      keys.add(quit);
      keys.add(connect);
      Panel south = new Panel();
      south.setLayout(new BorderLayout());
      south.add("West", keys);
      south.add("Center", input);
      south.add("East", send);
      Label title = new Label("Simple Chat Client Applet", Label.CENTER);
      title.setFont(new Font("Helvetica", Font.BOLD, 14));
      setLayout(new BorderLayout());
      add("North", title);
      add("Center", display);
      add("South",  south);
      quit.setEnabled(false);
      send.setEnabled(false);

      // first get the parameters (do the user request) and then establish the connection to server
      getParameters();
      try {
        connect();
      } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }

    /*
     *
     * method that handles the actions performed on the client side inside the applet.
     * it handles the events for connecting to the server and sending messages.
     * by clicking the bye button it handles the quitting.
     *
     */
    public boolean action(Event e, Object o) {

        if (e.target == quit) {
            input.setText("/exit");
            quit.setEnabled(false);
            send.setEnabled(false);
            connect.setEnabled(false);
            handle("/exit");
        }

        else if (e.target == connect) {
            getParameters(); // request parameters newly from user by showing input dialogs
            try {
                connect(); // connect again
            } catch (NotBoundException ex) {
                Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

        else if (e.target == send) {
            send();
            input.requestFocus();
        }
       return true;
    }

   /*
    * this method is called when clicking the connect button,
    * it establishes the connection to the server by first looking up for
    * RMICHAT on the serverside.
    */
    private void connect() throws NotBoundException {
        println("Establishing connection. Please wait ...");
       try {
           chat = new ChatObject(this);
          server = (rmiServer.ServerInterface)Naming.lookup("//"+ serverName +"/RMICHAT");
          server.connect(nickname, chat);
          send.setEnabled(true);
          connect.setEnabled(false);
          quit.setEnabled(true);
          join();
       } catch(java.rmi.UnknownHostException uhe) {
           println("Host unknown: " + uhe.getMessage());
           getParameters();
       } catch(IOException ioe) {
           println("Unexpected exception: " + ioe.getMessage());
           getParameters();
       }
    }

    /*
     * send method is called when clicking the send button in the applet,
     * it sends the nickname with his message to the server which then
     * broadcasts the message to all clients connected (done at server side).
     */
```

```
    private void send() {
        try {
            String message = input.getText();
            server.send(nickname + ": " + message); input.setText("");
        } catch(Exception e) {
            println("Sending error: " + e.getMessage());
            println("Server may be down at the time, program gets terminated.");
            handle("/exit");
        }
    }

    /* sending out the join message to the server
     */
    private void join() {
        try {
            server.send(nickname + " joined the chat!"); input.setText("");
        } catch(Exception e) {
            println("Sending error: " + e.getMessage());
            println("Server may be down at the time, program gets terminated.");
            handle("/exit");
        }
    }

    /* sending out the leave message to the server
     */
    private void leave() {
        try {
            server.send(nickname + " left the chat!"); input.setText("");
        } catch(Exception e) {
            println("Sending error: " + e.getMessage());
            println("Server may be down at the time, program gets terminated.");
            handle("/exit");
        }
    }


    /* modified method to handle RMI (public method header now)
     stands for SEND
     */
    public void handle(String msg) {
        if (msg.equals("/exit")) {
             try {
                 leave();
                 server.disconnect(chat);
             } catch (RemoteException ex) {
                 Logger.getLogger(Client.class.getName()).log(Level.SEVERE, null, ex);
             }
            println("Good bye. You can now close the window ...");
            quit.setEnabled(false);
            send.setEnabled(false);
            connect.setEnabled(false);
        } else println(msg);
    }


    /*modified method to handle RMI (public method header now)
     stands for RECEIVE
     */
    public void println(String msg) {
        display.append(msg + "\n");
    }

    public void getParameters() {
```

```
        serverName = JOptionPane.showInputDialog(null, "Enter server ip : ");
        System.out.println(serverName);

        nickname = JOptionPane.showInputDialog(null, "Enter nickname : ");
        System.out.println(nickname);

    }


}
```