**Project**: jChat

**Subject**: Object Oriented Programming Laboratory.

**Code**: CS8383.

**Collage**: PSNA Collage of Engineering and

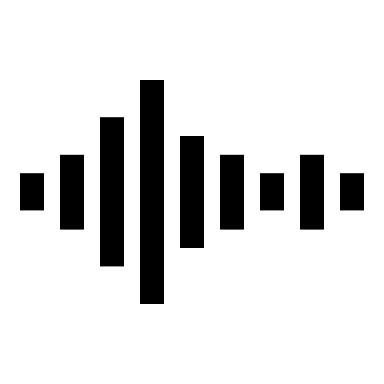
Technology, Dindigul.

**Department**: Information Technology.

**Year of Study**: II

**Section**: B

Java Mini Project

****

Submitted by

Navayuvan SB.

Manikandaprabhu S.

Mahesh Ram K.

**jChat**

**jChat**

**Idea:**

The idea is to provide a chatting interface between client and server using Java programming. This application is designed with GUI(Graphical User Interface) to make it user friendly.

The user can make contact with the server by the IP address of the server and the TCP port number.

The speciality of this application is, both the client and server runs on different thread. Thereby to avoid delay and collision of connection.

**Working:**

This Application is divided into three modules.

1. GUI
2. Client
3. Server

**GUI:**

The GUI of this application is designed with the help of Swing and Abstract Window Toolkit (awt). This app consist of 4 components.

1) Button for sending the message.

2) Text area to display the conversation of client and server.

3) Text area to type the message.

4) Label to display the role of that window (Client or server).

Font size of this application is changed from default for better experience. The size of the components are fixed by their bound’s coordinates, which makes this application look gentle.

The GUI is created, when an object for the class which inherits the thread is created. The module for GUI is written in the constructor of the subclass.

**Client:**

Client block executes a Socket connection with the Server’s IP address and TCP port number. Data output/input stream is employed to send and receive messages from the server.

Programmatically, Client block consist of following operations.

* + Executes the Socket connection.
  + Sends the message using DataOutputStream.
  + Appending the message in Text area.
  + Waiting for the server’s response and display the response in Text area.

The message sending task is performed when the send button is clicked. Client continues it’s process until “end” message is received.

The whole client module is written in the “run” function which invokes when the respective thread starts. A trick is employed here to avoid exception.

The message sending process is written in a separate function and called from ActionListener, because exception cannot be thrown by a overridden method as the code for this process wants a exception handling block with it.

**Server:**

Server block executes two connections. Server socket for the server connection and socket for interacting with client. Client cannot run with the server program in dead state.

It wants the server program to run so that it can make link with server. Server has to accept the socket connection (client) to execute the process.

Similar to the Client’s process, this also have the following operations:

* + Executes the Server Socket connection.
  + Executes the Server Socket connection.
  + Receiving the message using DataInputStream.
  + Appending the message in Text area.
  + Get the response from the Text area.
  + Sends the response using DataOutputStream.

The message sending task is performed when the send button is clicked. Client continues it’s process until “end” message is received.

The whole server module is written in the “run” function which invokes when the respective thread starts. A trick is employed here to avoid exception.

The message sending process is written in a separate function and called from ActionListener, because, exception cannot be thrown by a overridden method as the code for this process wants a exception handling block with it.

**Features:**

* An interacting medium between client and server.
* Can interact with the server only by the knowing their IP address and TCP port number.
* Fast communication between Client and server.

**Dependencies:**

* + Socket programming.
  + Abstract Window Toolkit (awt).
  + Swing.
  + JDK-10.
  + Java.

**Source code:**

Chatserver.java

import javax.swing.\*;

import java.awt.event.\*;

import java.io.\*;

import java.net.\*;

class Chatserver {

public static void main(String[] args) {

try{

MyserverT myservert = new MyserverT();

myservert.start();

}

catch(Exception e){

}

}

}

class MyserverT extends Thread{

public JLabel l1,l2;

public JTextArea area,space;

public JButton b;

public DataInputStream din;

public DataOutputStream dout;

public ServerSocket ss;

public String msgin="",msgout="";

public Socket s;

MyserverT(){

JFrame f = new JFrame("Server");

l1 = new JLabel();

l1.setBounds(220,10,280,30);

l1.setText("Server");

f.add(l1);

area = new JTextArea();

area.setBounds(5,50,490,480);

area.setVisible(true);

area.setFont(area.getFont().deriveFont(20.0f));

area.setEditable(false);

f.add(area);

b = new JButton("Send");

b.setBounds(200,600,100,40);

f.add(b);

space = new JTextArea();

space.setBounds(5,540,490,50);

space.setFont(space.getFont().deriveFont(25.0f));

f.add(space);

f.setSize(500,700);

f.setLayout(null);

f.setVisible(true);

}

public void run(){

try{

ServerSocket ss = new ServerSocket(39521);

Socket s = ss.accept();

din = new DataInputStream(s.getInputStream());

dout = new DataOutputStream(s.getOutputStream());

while(!msgin.equals("end")){

msgin = din.readUTF();

area.append("Client: "+msgin+"\n");

b.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

msgout = space.getText().toString();

call(msgout,dout);

b = new JButton("Send");

}

});

}

}

catch(Exception e){

space.setText("Oops...!!! Problem in starting Server Socket");

space.setEditable(false);

b.setEnabled(false);

}

}

public void call(String a,DataOutputStream dout){

try{

area.append("Server: "+a+"\n");

dout.writeUTF(a);

dout.flush();

space.setText("");

}

catch(Exception e){

space.setText("Error in sending message");

space.setEditable(false);

b.setEnabled(false);

}

}

}

Chatclient.java

import javax.swing.\*;

import java.awt.event.\*;

import java.io.\*;

import java.net.\*;

class Chatclient {

public static void main(String[] args) {

try{

MyclientT myclientt = new MyclientT();

myclientt.start();

}

catch(Exception e){

}

}

}

class MyclientT extends Thread{

public JLabel l1,l2;

public JTextArea area;

public JTextField space;

public JButton b;

public DataInputStream din;

public DataOutputStream dout;

public String msgin="",msgout="";

public Socket s;

MyclientT(){

JFrame f = new JFrame("Client");

l1 = new JLabel();

l1.setBounds(220,10,280,30);

l1.setText("Client");

f.add(l1);

area = new JTextArea();

area.setBounds(5,50,490,480);

area.setVisible(true);

area.setFont(area.getFont().deriveFont(20.0f));

area.setEditable(false);

f.add(area);

b = new JButton("Send");

b.setBounds(200,600,100,40);

f.add(b);

space = new JTextField(20);

space.setBounds(5,540,490,50);

space.setFont(space.getFont().deriveFont(25.0f));

f.add(space);

f.setSize(500,700);

f.setLayout(null);

f.setVisible(true);

}

public void run(){

try{

Socket s = new Socket("localhost",39521);

din = new DataInputStream(s.getInputStream());

dout = new DataOutputStream(s.getOutputStream());

while(!msgin.equals("end")){

b.addActionListener(new ActionListener(){

public void actionPerformed(ActionEvent e){

msgout = space.getText();

call(msgout,dout);

//space.setText("");

b = new JButton("Send");

}

});

msgin = din.readUTF();

area.append("Server: "+msgin+"\n");

}

}

catch(Exception e){

space.setText("Oops...!!! Problem in starting Client Socket");

space.setEditable(false);

b.setEnabled(false);

}

}

public void call(String a,DataOutputStream dout){

try{

area.append("You: "+a+"\n");

dout.writeUTF(a);

space.setText("");

}

catch(Exception e){

space.setText("Error in sending message");

space.setEditable(false);

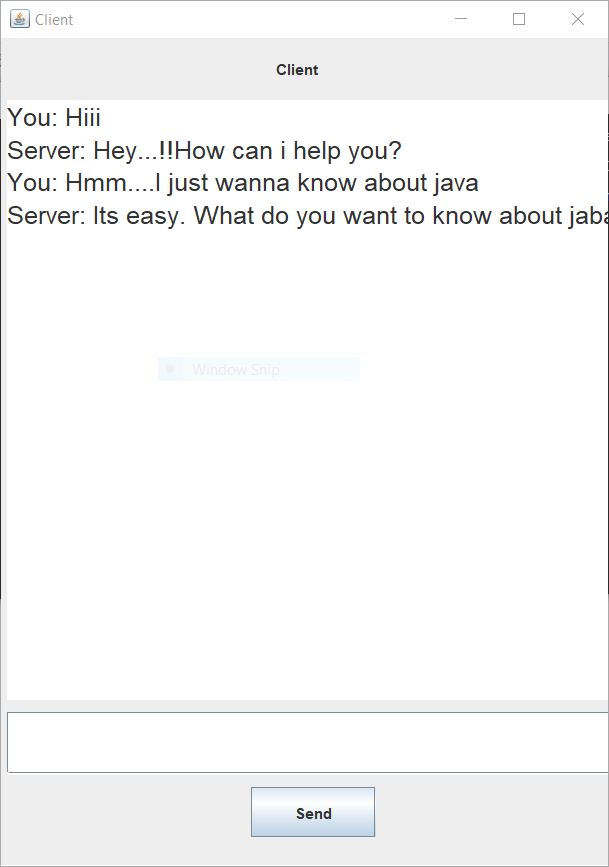
b.setEnabled(false);

}

}

}

**Output:**

****



**Team members:**

* + Navayuvan SB.
  + Manikandaprabhu S.
  + Mahesh Ram K.