NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA, SURATHKAL

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE – Paradigms Of Programming
COURSE CODE - IT206

A Project Report
on

BROADCASTING CHAT SERVER

Submitted By :

Ganesh P Nischay	**16IT220**
Rahul A R	**16IT239**
Supreeth G	**16IT246**
Sai Kumar	**16IT241**

Faculty of Information Technology , Mrs.Priyadarshini
2017

2.INTRODUCTION

Implementing chat server application provides a good opportunity for a beginner to design and implement a network-based system.The design is very simple.It is implemented in Java,since is easy to program in ,it precludes the need to deal with low-level memory management and includes powerful libraries for sockets and threads.A very simple cross-platform client-server chat application has been implementedin Java. Its design is described, limitations are discussed, improvements are proposed and a user manual is included.

3.PROBLEM STATEMENT

- This project is to create a chat application with a server and clients to enable the clients to chat with many other clients in the same common chat group.
- This software can be used on any system within the same server connection
- To create a chat window and display the sent and received messages
- The main purpose of this project is to provide chatting functionality through network

-

4.OBJECTIVE

- To enable data exchange in text format between two computers in connection
- To create easy interface for data exchange
- To receive notifications when new messages arrives
- This whole process take through sockets
- To clear the chat data according to the user

5.SYSTEM SPECIFICATION

5.1 Software Requirement

Language : Java
Platform : Netbeans IDE
Tool : JDK
Client : Own Client designed Using Java Server Socket
Server : Server designed using Java Server Socket

5.2 Hardware Requirement

RAM : 128MB(min)
Processor : Pentium 2 and Above
Processor speed : Above 500MHz

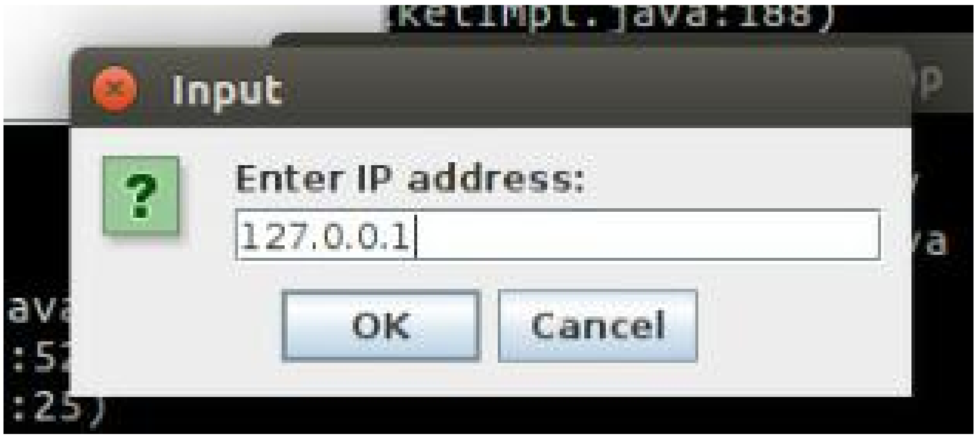
Fuctional Requirment :

- The data will be valid until the server is valid
- Transmitting data between Client and Server is has been developed using IP address
- We are running it as a Server Client in PC itself
- Usage of SQL server is will not be needed for sending Messages and Attachments

6.WORK DONE (with screenshots)

6.1 Hosting Server

This window will pop up. Then you have submit valid info to successfully Register

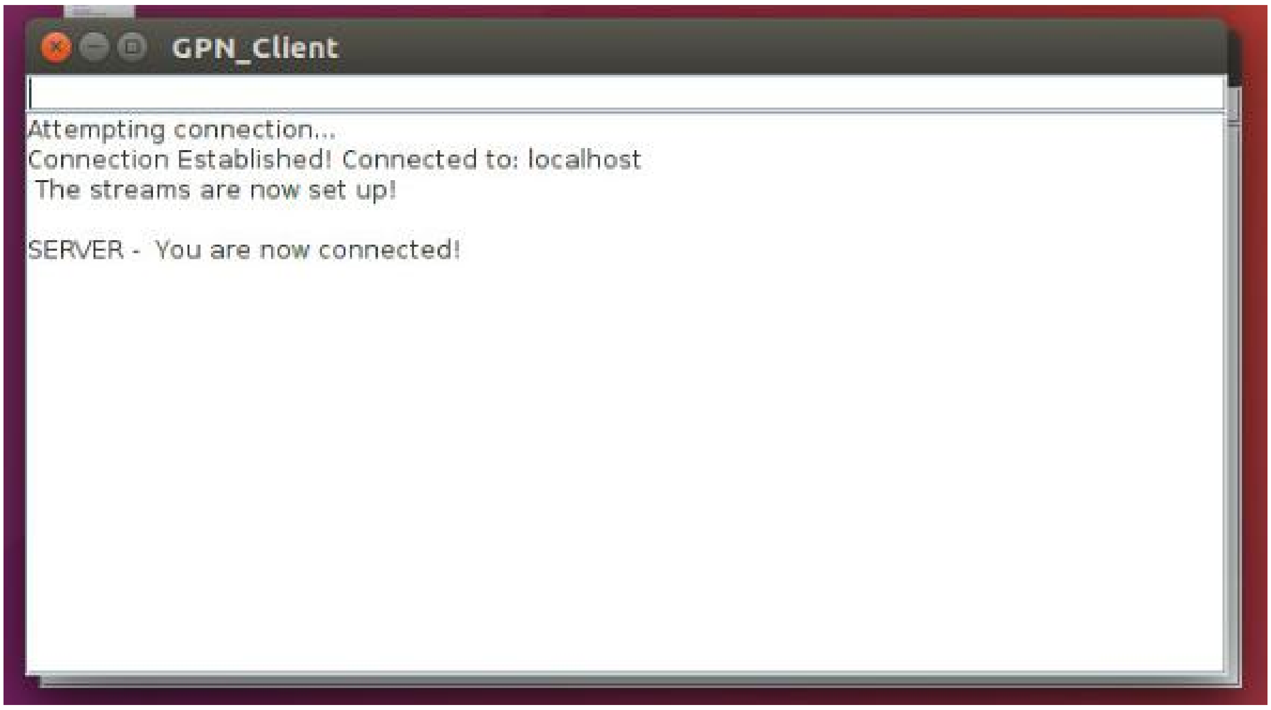


Enter IP address:

127.0.0.1

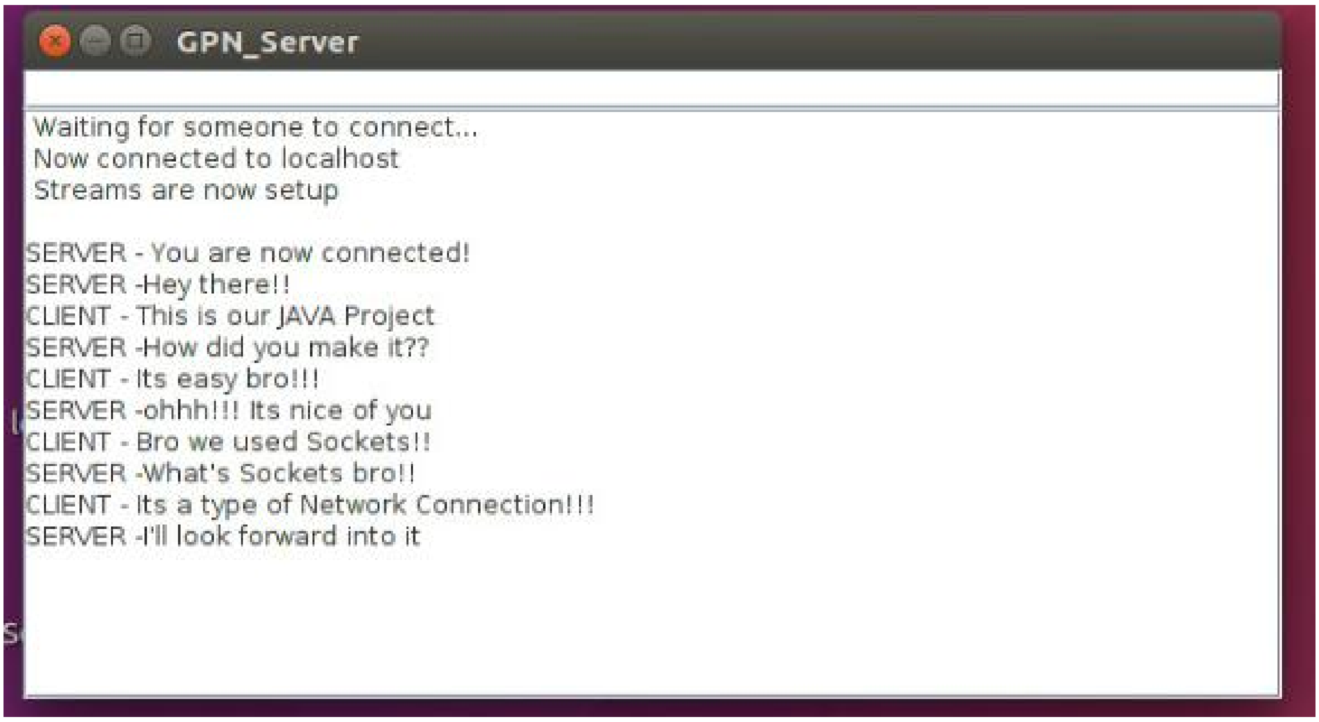
6.2 Client connection

This window will pop up. Then again submit info to establish connection

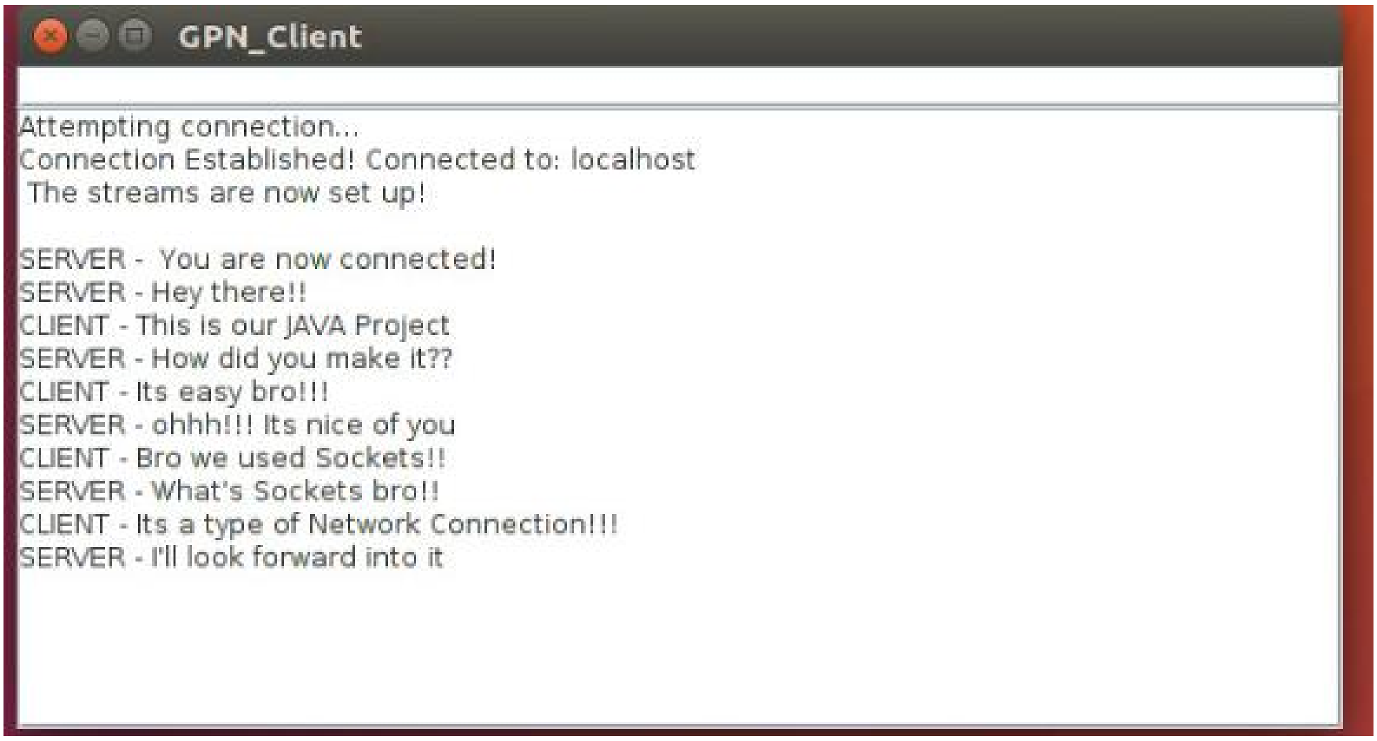


```
Attempting connection...
Connection Established! Connected to: localhost
 The streams are now set up!

SERVER -  You are now connected!
```

6.3 Private Chatting



```
Waiting for someone to connect...
Now connected to localhost
Streams are now setup

SERVER - You are now connected!
SERVER -Hey there!!
CLIENT - This is our JAVA Project
SERVER -How did you make it??
CLIENT - Its easy bro!!!
SERVER -ohhh!!! Its nice of you
CLIENT - Bro we used Sockets!!
SERVER -What's Sockets bro!!
CLIENT - Its a type of Network Connection!!!
SERVER -I'll look forward into it
```



```
Attempting connection...
Connection Established! Connected to: localhost
 The streams are now set up!

SERVER -  You are now connected!
SERVER - Hey there!!
CLIENT - This is our JAVA Project
SERVER - How did you make it??
CLIENT - Its easy bro!!!
SERVER - ohhh!!! Its nice of you
CLIENT - Bro we used Sockets!!
SERVER - What's Sockets bro!!
CLIENT - Its a type of Network Connection!!!
SERVER - I'll look forward into it
```

7.FUTURE ENHANCEMENT CONCLUSION

There is always a room for improvements in any software package, however good and efficient may be done.

- File transfer : this will enable the userr to send files of different forms via chat application
- Voice chat : this will enhance the application to a higher great level where communication will be possible via aliing as in telephone
- An improved version an include multiple servers,serving different geographical locations while talking to each other.

8.CONCLUSION

There is always a room for improvement . Right now we are dealing with text comunication . There are several projects/apps which serve similar as this project . A positive first impression is available in human relationship and interactions . This project hopes to develop a chat sevice.

-.---

9.REFERENCE

[1] Open Source Chat Servers in Java http://java-source.net/open-source/chat-servers

[2] The Singleton Design Pattern - Brian D Foy
http://www.theperlreview.com/Articles/v0i1/singletons.pdf

[3] Internet Relay Chat
http://en.wikipedia.org/wiki/Internet_Relay_Chat

[4]
javaTpoint

APPENDIX [Source Code]

Server.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package servertest;


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

/**
 *
 * @author gpn
 */
public class Server extends JFrame{

    private JTextField userText;
        private JTextArea chatWindow;
        private ObjectOutputStream output;
        private ObjectInputStream input;
        private ServerSocket server;
        private Socket connection;

        //constructor
        public Server(){
                super("Server");
                userText = new JTextField();
                userText.setEditable(false);
                userText.addActionListener(
                        new ActionListener(){
                                public void actionPerformed(ActionEvent event){
```

```
                                sendMessage(event.getActionCommand());
                                userText.setText("");
                        }
                }
        );
        add(userText, BorderLayout.NORTH);
        chatWindow = new JTextArea();
        add(new JScrollPane(chatWindow));
        setSize(600, 300);
        setVisible(true);
    }

    public void startRunning(){
        try{
            server = new ServerSocket(6789, 100);
            while(true){
                try{

                    waitForConnection();
                    setupStreams();
                    whileChatting();
                }catch(EOFException eofException){
                    showMessage("\n Server ended the 
connection! ");
                } finally{
                    closeConnection();
                }
            }
        } catch (IOException ioException){
            ioException.printStackTrace();
        }
    }
    private void waitForConnection() throws IOException{
        showMessage(" Waiting for someone to connect... \n");
        connection = server.accept();
        showMessage(" Now connected to " + 
connection.getInetAddress().getHostName());
    }
```

```
	private void setupStreams() throws IOException{
		output = new
ObjectOutputStream(connection.getOutputStream());
		output.flush();

		input = new ObjectInputStream(connection.getInputStream());

		showMessage("\n Streams are now setup \n");
	}

	private void whileChatting() throws IOException{
		String message = " You are now connected! ";
		sendMessage(message);
		ableToType(true);
		do{
			try{
				message = (String) input.readObject();
				showMessage("\n" + message);
			}catch(ClassNotFoundException
classNotFoundException){
				showMessage("The user has sent an unknown
object!");
			}
		}while(!message.equals("CLIENT - END"));
	}

	public void closeConnection(){
		showMessage("\n Closing Connections... \n");
		ableToType(false);
		try{
			output.close();
			input.close();
			connection.close();
		}catch(IOException ioException){
			ioException.printStackTrace();
		}
	}

	private void sendMessage(String message){
		try{
			output.writeObject("SERVER - " + message);
			output.flush();
			showMessage("\nSERVER -" + message);
		}catch(IOException ioException){
			chatWindow.append("\n ERROR: CANNOT SEND 
MESSAGE, PLEASE RETRY");
		}
	}

	private void showMessage(final String text){
		SwingUtilities.invokeLater(
			new Runnable(){
				public void run(){
					chatWindow.append(text);
				}
			}
		);
	}

	private void ableToType(final boolean tof){
		SwingUtilities.invokeLater(
			new Runnable(){
				public void run(){
					userText.setEditable(tof);
				}
			}
		);
	}


}
```

ServerTest.java

```
/*
 * To change this license header, choose License Headers in Project 
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package servertest;
import javax.swing.JFrame;
import javax.swing.JOptionPane;



/**
 *
 * @author gpn
 */
public class ServerTest {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Server server = new Server();
        server.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        server.startRunning();
    }

}
```

Client.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package clienttest;
import java.io.*;
import java.net.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

```
/**
 *
 * @author gpn
 */
public class Client extends JFrame{

        private JTextField userText;
         private JTextArea chatWindow;
         private ObjectOutputStream output;
         private ObjectInputStream input;
         private String message = "";
         private String serverIP;
         private Socket connection;

         //constructor
         public Client(String host){
                 super("Client");
                 serverIP = host;
                 userText = new JTextField();
                 userText.setEditable(false);
                 userText.addActionListener(
                                 new ActionListener(){
                                 public void actionPerformed(ActionEvent event){
                                         sendMessage(event.getActionCommand());
                                         userText.setText("");
                                 }
                         }
                 );
                 add(userText, BorderLayout.NORTH);
                 chatWindow = new JTextArea();
                 add(new JScrollPane(chatWindow));
                 setSize(600, 300);
                 setVisible(true);
         }

         public void startRunning(){
                 try{
                         connectToServer();
```

```
                        setupStreams();
                        whileChatting();
                }catch(EOFException eofException){
                        showMessage("\n Client terminated the connection");
                }catch(IOException ioException){
                        ioException.printStackTrace();
                }finally{
                        closeConnection();
                }
        }

        private void connectToServer() throws IOException{
                showMessage("Attempting connection... \n");
                connection = new Socket(InetAddress.getByName(serverIP),
6789);
                showMessage("Connection Established! Connected to: " +
connection.getInetAddress().getHostName());
        }

        private void setupStreams() throws IOException{
                output = new
ObjectOutputStream(connection.getOutputStream());
                output.flush();
                input = new ObjectInputStream(connection.getInputStream());
                showMessage("\n The streams are now set up! \n");
        }

        private void whileChatting() throws IOException{
                ableToType(true);
                do{
                        try{
                                message = (String) input.readObject();
                                showMessage("\n" + message);
                        }catch(ClassNotFoundException
classNotFoundException){
                                showMessage("Unknown data received!");
                        }
                }while(!message.equals("SERVER - END"));
        }
```

```
private void closeConnection(){
        showMessage("\n Closing the connection!");
        ableToType(false);
        try{
                output.close();
                input.close();
                connection.close();
        }catch(IOException ioException){
                ioException.printStackTrace();
        }
}

private void sendMessage(String message){

   try{
                output.writeObject("CLIENT - " + message);
                output.flush();
                showMessage("\n" + "CLIENT - " + message);
        }catch(IOException ioException){
                chatWindow.append("\n Oops! Something went wrong!");
        }
}

private void showMessage(final String message){
        SwingUtilities.invokeLater(
                new Runnable(){
                        public void run(){
                                chatWindow.append(message);
                        }
                }
        );
}

private void ableToType(final boolean tof){
        SwingUtilities.invokeLater(
                new Runnable(){
                        public void run(){
                                userText.setEditable(tof);
                        }
                    }
                );
        }

}
```

ClientTest.java

```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package clienttest;
import javax.swing.JFrame;
import javax.swing.JOptionPane;

/**
 *
 * @author gpn
 */
public class ClientTest {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Client client;
        String InternetProtocol = JOptionPane.showInputDialog("Enter IP
address: ");
        client = new Client(InternetProtocol);
        client.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        client.startRunning();
    }

}
```

*****THE END*****