**Java Programming (22412)**

**Micro-project proposal-JPR**

***Peer to Peer(P2P) Chat Application***

**1.0 Rationale**

Peer-to-peer computing or networking is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.

The project consists of the chat application in which multiple users can chat simultaneously. User has to enter his/her name to continue the chatting. The name entered by user will be seen by all the remaining users who are chatting with the that user. The program code consists of 4 classes, Peer , PeerThread, ServerThread, ServerThreadThread. The code consists of properties of Java with proper syntaxes.

Peer to Peer network is type of network architecture used to create a network. In it all systems are connected to each other. We can use different consoles as different peers when we will execute the program.

**2.0 Aim of the project**

This micro-project aims were

1. To analyze working of Peer to Peer network.
2. To learn about the Java projects existing in real world.
3. To implement the Java commands.

We have achieved all the aims which were decided at the start of the project. We have completed the project with achieving all the aims at the completion.

**3.0 Course Outcomes Achieved**

1. Develop programs using Object Oriented methodology in java.
2. Develop programs using Multithreading.
3. Implement Exception Handling.

**4.0 Actual procedure followed**

Actual procedure followed during the project is mentioned in the table below with the planned start date and completed finish date.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.** | **Details of Activity** | **Planned Start Date** | **Completed Finish Date** | **Name of responsible Team Members** |
| 1. | Collecting information about Peer to Peer network | 10/01/2019 | 15/01/2019 | Patil Pratik |
| 2. | Collecting the information about Java language | 16/01/2019 | 26/01/2019 | Chavan Shubham |
| 3. | Designing the overall logic of project | 27/01/2019 | 01/02/2019 | Kare Abhilash |
| 4. | Designing the actual commands to be used in program | 02/02/2019 | 15/02/2019 | Chavan Shubham |
| 5. | Actual Coding of the project | 16/02/2019 | 22/02/2019 | Patil Pratik |
| 6. | Observing Output | 23/02/2019 | 01/03/2019 | Kare Abhilash  Chavan Shubham |
| 7. | Making Report | 02/03/2019 | 06/03/2019 | Patil Pratik |

* **Code**

**1.Peer class**

import java.io.BufferedReader;

import javax.json.Json;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

import java.io.StringWriter;

public class Peer {

public static void main(String[] args) throws Exception {

BufferedReader bufferedReader= new BufferedReader(new InputStreamReader(System.in));

System.out.println("Enter username & port no for this peer");

String[] setupValues=bufferedReader.readLine().split(" ");

ServerThread serverThread=new ServerThread(setupValues[1]);

serverThread.start();

new Peer().updateListenToPeers(bufferedReader, setupValues[0], serverThread);

}

public void updateListenToPeers(BufferedReader bufferedReader, String username, ServerThread serverThread) throws Exception

{

System.out.println("> Enter (space separated hostname:portno");

System.out.println(" peers to receive Messages from (s to skip):");

String input=bufferedReader.readLine();

String[] inputValues=input.split(" ");

if (!input.equals("s"))

for(int i=0;i<inputValues.length;i++)

{

String[] address = inputValues[i].split(":");

Socket socket = null;

try

{

socket = new Socket(address[0],Integer.valueOf(address[1]));

new PeerThread(socket).start();

}

catch(Exception e)

{

if(socket != null)

socket.close();

else

System.out.println("invalid input. skipping to next step. ");

}

communicate(bufferedReader, username, serverThread);

}

}

public void communicate(BufferedReader bufferedReader, String username, ServerThread serverThread)

{

try

{

System.out.println("> you can now communicate (e to exit, c to change)");

boolean flag = true;

while(flag)

{

String message = bufferedReader.readLine();

if(message.equals("e"))

{

flag = false;

break;

}

else if (message.equals("c"))

{

updateListenToPeers(bufferedReader, username, serverThread);

}

else

{

StringWriter stringWriter = new StringWriter();

Json.createWriter(stringWriter).writeObject(Json.createObjectBuilder() .add("username", username)

.add("message", message)

.build());

serverThread.sendMessage(stringWriter.toString());

}

System.exit(0);

}

}

catch(Exception e) { }

}

}

**2. PeerThread class**

import java.io.BufferedReader;

import java.io.IOException;

import java.json.Json;

import java.json.JsonObject;

import java.io.InputStreamReader;

import java.net.Socket;

public class PeerThread extends Thread{

private BufferedReader bufferedReader;

public PeerThread(Socket socket) throws IOException{

bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

}

public void run()

{

boolean flag = true;

while(flag)

{

try

{

JsonObject jsonObject = Json.createReader(bufferedReader).readObject();

if (jsonObject.containskey("username"))

System.out.println("[" + jsonObject.getString("username") + "]: " + jsonObject.getString("message"));

}

catch(Exception e)

{

flag=false;

interrupt();

}

}

}

}

**3.ServerThread class**

import java.net.ServerSocket;

import java.util.HashSet;

import java.util.Set;

import java.io.IOException;

public class ServerThread extends Thread{

private ServerSocket serverSocket;

private Set<ServerThreadThread> serverThreadThreads = new HashSet<ServerThreadThread>();

public ServerThread(String portNumb) throws IOException {

serverSocket = new ServerSocket(Integer.valueOf(portNumb));

}

public void run()

{

try

{

ServerThreadThread serverThreadThread = new ServerThreadThread(serverSocket.accept(), this);

serverThreadThreads.add(serverThreadThread);

serverThreadThread.start();

}

catch(Exception e)

{

e.printStackTrace();

}

}

void sendMessage(String message)

{

try

{

serverThreadThreads.forEach(t-> t.getPrintWriter().println(message));

}

catch(Exception e)

{

e.printStackTrace();

}

}

public Set<ServerThreadThread> getServerThreadThreads()

{

return serverThreadThreads;

}

}

**4.ServerThreadThread class**

import java.net.Socket;

import java.io.BufferedReader;

import java.io.PrintWriter;

import java.io.InputStreamReader;

public class ServerThreadThread extends Thread{

private ServerThread serverThread;

private Socket socket;

private PrintWriter printWriter;

public ServerThreadThread(Socket socket, ServerThread serverThread)

{

this.serverThread = serverThread;

this.socket = socket;

}

public void run()

{

try

{

BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));

this.printWriter = new PrintWriter(socket.getOutputStream(), true);

while(true)

serverThread.sendMessage(bufferedReader.readLine());

}

catch(Exception e)

{

serverThread.getServerThreadThreads().remove(this);

}

}

public PrintWriter getPrintWriter()

{

return printWriter;

}

}

**5.0 Actual Resources Required**

The resources used during the completion of project are mentioned in the below table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. No.** | **Name of resources material** | **Specifications** | **Quantity** | **Remarks** |
| 1. | Internet | Network of Networks | 1 |  |
| 2. | YouTube | MP4-file format, 640 x 360 pixels | 1 |  |
| 3. | Microsoft Word | 2010 version | 1 |  |
| 4. | Ecllipse | Version 1.8.0 | 1 |  |
| 5. | Desktop PC | RAM 2 GB, Intel dual core processor | 1 |  |
| 6. | Laptop | Dell, RAM 4 GB, Intel (r) core processor | 1 |  |
| 7. | Printer | LaserJet | 1 |  |

**7.0 Skill Developed/ learning out of this Micro-Project**

Various skills got developed in us by doing this project. This skills are

1. Creating a Java program .
2. Using Java packages and classes.
3. Creating the program for peer to peer chat application.
4. Debugging the errors.
5. Using the concept of Java Programming.
6. Efficient communication skills.

This were skills developed by us during the project.

Learning’s were about the debugging the errors. A single error was troubling our program. We understood that because off a single error the program will not run. So, debugging errors is very important.

**8.0 Applications of Project**

1. The project can be used as social networking site in peer to peer network.

2. The project can be used to understand concept of Object Oriented Programming, Multithreading, Exception Handling.

3. The project can be used to study real world applications of Java Programming.

**Submitted By:**

**1.Patil Pratik Kumar 856**

**Subject Teacher**

**Mrs. Jadhav K.P.**