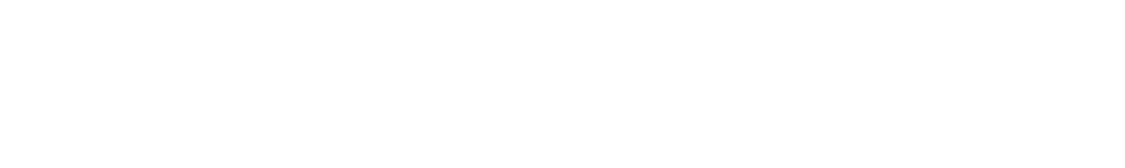
|  |  |  |  |
| --- | --- | --- | --- |
| **Class: FYMCA Div: A Semester: II** | **Course Code: MCA01554**  **Course Name: Java Programming Laboratory** | | **Batch: F1** |
| **Name: Abhijeet Joshi** | | **Roll No: 51023** | |
| **CO No: CO515.3** | | **Assignment No:** 7 | |

**PRACTICAL SUBMISSION RECORD- A.Y. 2024-25**



Progressive Education Society’s

**MODERN COLLEGE OF ENGINEERING, Pune -05.**

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

**MCA Department**

**Title**: Write a Java Socket program for client server chatting application.

# Code:

1. **Client Side code** import java.io.\*; import java.net.\*;

public class ChatClient {

public static void main(String[] args) { String serverAddress = "localhost"; int serverPort = 1234;

Socket socket = null; try {

socket = new Socket(serverAddress, serverPort); System.out.println("Connected to the server...");

Thread sendThread = new Thread(new MessageSender(socket)); Thread receiveThread = new Thread(new MessageReceiver(socket)); sendThread.start();

receiveThread.start(); sendThread.join(); receiveThread.join();

} catch (IOException | InterruptedException e) { e.printStackTrace();

} finally { try {

if (socket != null && !socket.isClosed()) { socket.close();

System.out.println("Disconnected from server.");

}

} catch (IOException e) { e.printStackTrace();

}

}

}

}

class MessageSender implements Runnable { private final Socket socket;

public MessageSender(Socket socket) { this.socket = socket;

}

@Override

public void run() { try (

BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())); BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.in))

) {

String message;

while ((message = consoleReader.readLine()) != null) { writer.write(message);

writer.newLine(); writer.flush();

if (message.equalsIgnoreCase("exit")) { socket.close();

break;

}

}

} catch (IOException e) {

System.out.println("Sender stopped: " + e.getMessage());

}

}

}

class MessageReceiver implements Runnable { private final Socket socket;

public MessageReceiver(Socket socket) { this.socket = socket;

}

@Override

public void run() { try (

BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()))

) {

String message;

while ((message = reader.readLine()) != null) { System.out.println("Server: " + message);

}

} catch (IOException e) {

System.out.println("Receiver stopped: " + e.getMessage());

}

}

}

1. **Server Side Code** import java.io.\*; import java.net.\*;

public class ChatServer {

public static void main(String[] args) { int port = 1234;

try (ServerSocket serverSocket = new ServerSocket(port)) { System.out.println("Server is waiting for client connection..."); Socket clientSocket = serverSocket.accept();

System.out.println("Client connected: " + clientSocket.getInetAddress().getHostAddress()); Thread sendThread = new Thread(new MessageSender(clientSocket));

Thread receiveThread = new Thread(new MessageReceiver(clientSocket)); sendThread.start();

receiveThread.start();

} catch (IOException e) { e.printStackTrace();

}

}

}

class MessageSender implements Runnable { private Socket socket;

public MessageSender(Socket socket) { this.socket = socket;

}

@Override

public void run() {

try (BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()))) {

BufferedReader consoleReader = new BufferedReader(new InputStreamReader(System.in)); String message;

while ((message = consoleReader.readLine()) != null) { writer.write(message);

writer.newLine(); writer.flush();

}

} catch (IOException e) { e.printStackTrace();

}

}

}

class MessageReceiver implements Runnable { private Socket socket;

public MessageReceiver(Socket socket) { this.socket = socket;

}

@Override

public void run() {

try (BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream())))

{

String message;

while ((message = reader.readLine()) != null) { System.out.println("Client: " + message);

}

} catch (IOException e) { e.printStackTrace();

}

}

}

# Output:

