**Experiment: 6**

**To implement chat server using socket programming (using UDP & TCP)**

**Aim:** Write a program to implement Chat server using socket programming

**Introduction:**

Java Socket programming is used for communication between the applications running on different JRE. Java Socket programming can be connection-oriented or connection-less. ‘Socket’ and ‘ServerSocket’ classes are used for connection-oriented socket programming and ‘DatagramSocket’ and ‘DatagramPacket’ classes are used for connection-less socket programming. The client in socket programming must know two information:

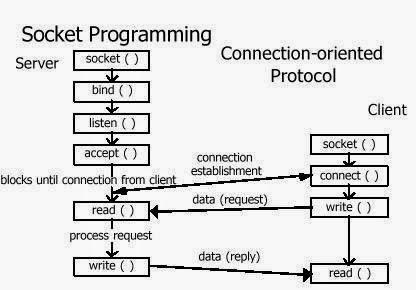
1. IP Address of Server, and
2. Port number.

The java.net package of the J2SE APIs contains a collection of classes and interfaces that provide the low-level communication details.

The java.net package provides support for the two common network protocols,

* **TCP** − TCP stands for Transmission Control Protocol, which allows **reliable, connection oriented** communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
* **UDP** − UDP stands for User Datagram Protocol, a **fast, connection-less protocol** that allows for packets of data to be transmitted between applications.

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server. The flow chart of server-client chat is as shown below,



**Procedure:** *(Ref: 1)* [*https://www.javatpoint.com/socket-programming*](https://www.javatpoint.com/socket-programming)

2) *https://www.geeksforgeeks.org/socket-programming-in-java/ )*

**Algorithm at Server side**

1. Start the program

2. Create a socket in server to client

3. The server establishes a connection to the client.

4. The server accept the connection and to send the data from server to client and vice versa

5. The server communicate the client to send the end of the message

6. Stop the program.

**Algorithm at Client Side**

1. Start the program

2. Create a socket in client to server.

3. The client establishes a connection to the server.

4. The client accept the connection and to send the data from client to server and vice versa

5. The client communicate the server to send the end of the message

6. Stop the program.

//Server side for chat application (save the file as Senddata.java)

import java.io.\*;

import java.net.\*;

class Senddata{

public static void main(String[] args)throws IOException

{

DatagramSocket ds=new DatagramSocket();

String s="";

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

s=br.readLine();

InetAddress ip=InetAddress.getByName("localhost");

DatagramPacket dp=new DatagramPacket(s.getBytes(),s.length(),ip,8888);

ds.send(dp);

ds.close();

}

}

//Client side for chat application (save the file as Receivedata.java)

import java.io.\*;

import java.net.\*;

class Receivedata{

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

{

DatagramSocket ds=new DatagramSocket(8888);

byte[] b=new byte[1024];

DatagramPacket dp=new DatagramPacket(b,1024);

ds.receive(dp);

String s=new String(dp.getData(),0,dp.getLength());

System.out.println(s);

ds.close();

}

}

Open two command prompts. First execute server program and then the client program at respective command prompts. After running the client application, a message will be displayed on the server console.

**Implementation:**

Implement socket programming in java to read and write on both sides (client and server) using TCP socket.

**Output:**

Attach print of following as output:

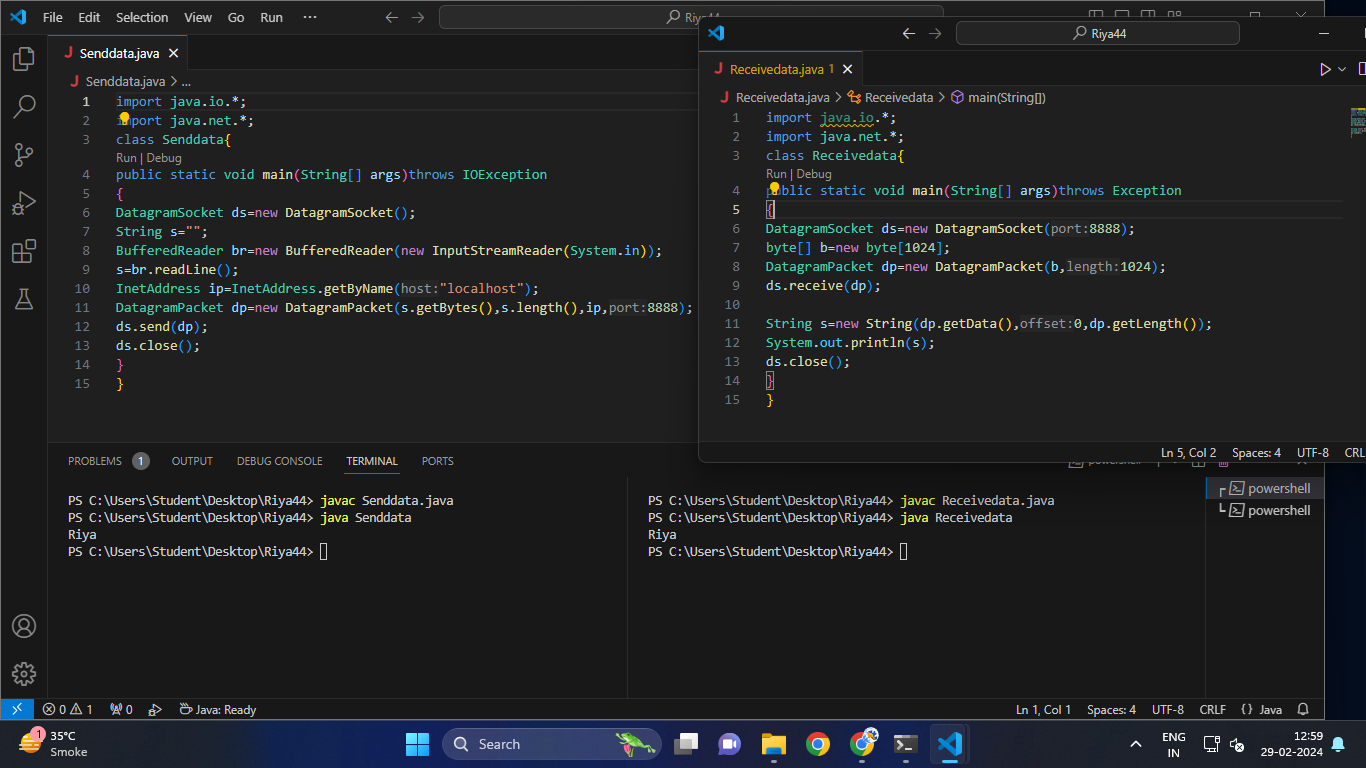
1. Senddata.java and Receivedata.java programs (provided), Program outputs.
2. Client and server tcp read-write socket programs and its outputs.

**Post-Experimental Exercise:** *(To be written on journal sheet)*

1. Make a table of all important methods with their description that comes under DatagramPacket and DatagramSocket (for UDP) and Socket and ServerSocket (for TCP).

**Conclusion:** *(To be written on journal sheet)*

***RIYA INDAP,44***

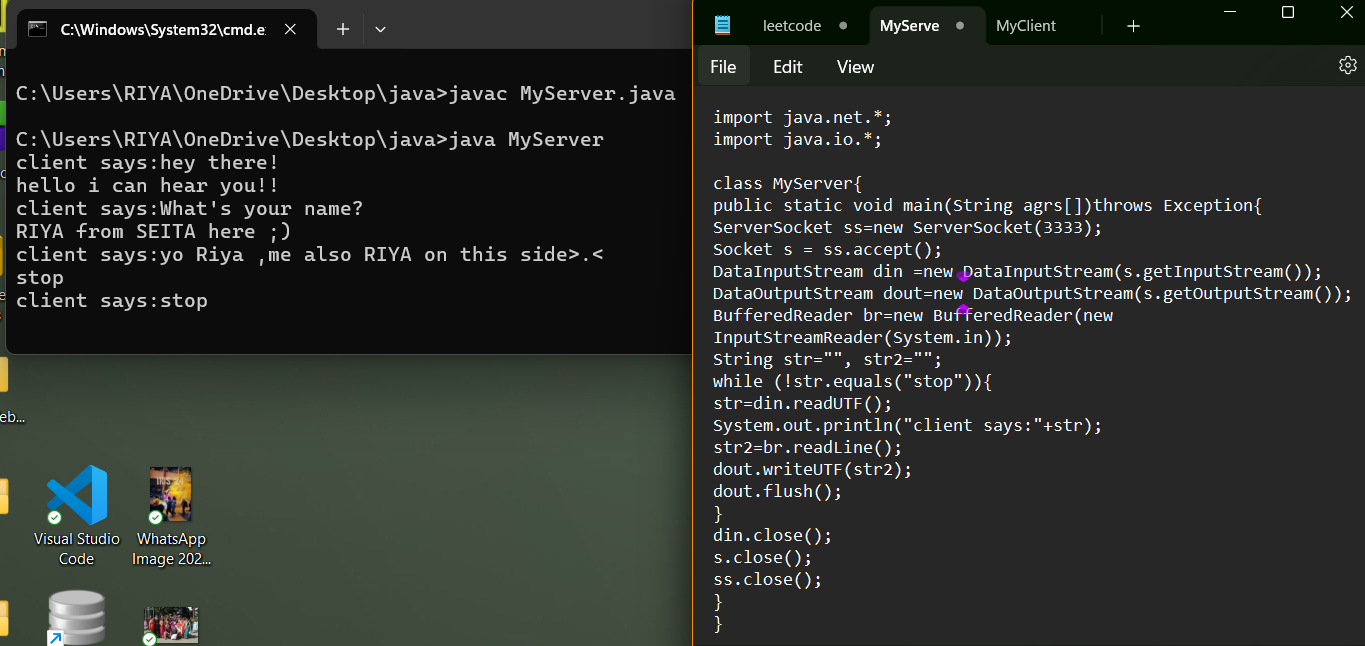


*Senddata:*

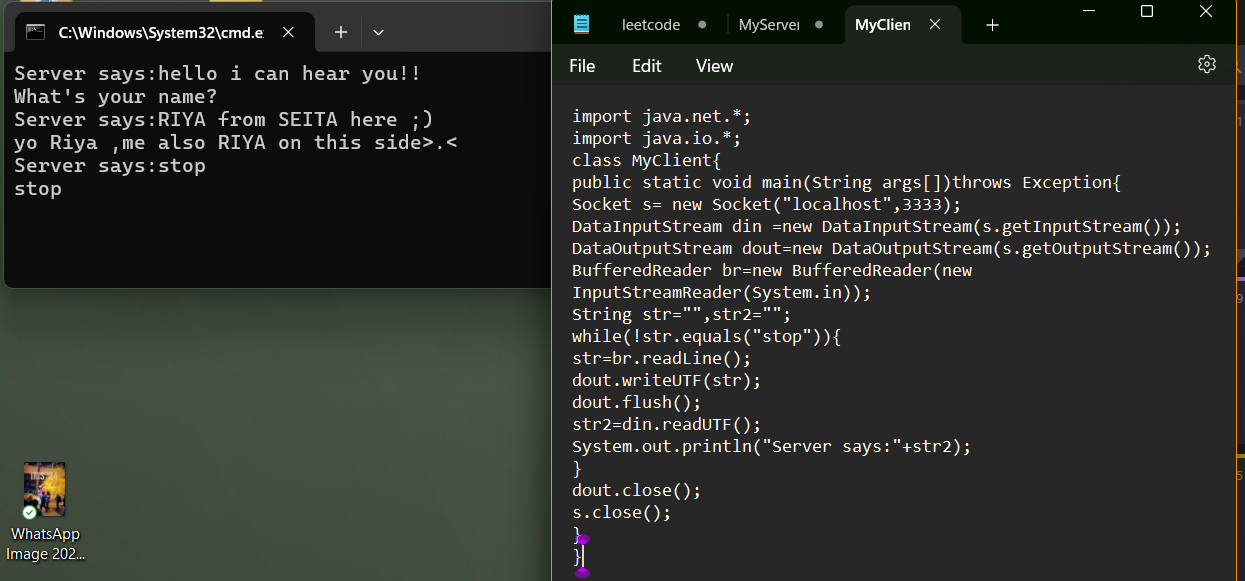
*This Java program establishes a DatagramSocket and allows a user to input a string from the console. It then creates a DatagramPacket containing the input string and sends it to the localhost on port 8888 using UDP. Finally, it closes the DatagramSocket. The code essentially demonstrates a basic implementation of sending data over a network using UDP sockets in Java.*

*Receivedata:*

*This Java program creates a DatagramSocket bound to port 8888 and waits to receive a DatagramPacket. Upon receiving a packet, it extracts the data from the packet, converts it to a string, and then prints it to the console. Finally, it closes the DatagramSocket. Essentially, this code demonstrates a simple implementation of receiving data over a network using UDP sockets in Java.*

Q.Implement socket programming in java to read and write on both sides (client and server) using TCP socket.**

*Server Side:This Java code represents a basic server application that listens on port 3333 for incoming connections from clients. Once a client connects, the server starts a conversation where the client sends a message, and the server responds to it. This conversation continues until the client sends the message "stop".*



*Client Side:This Java code represents a basic client application that connects to a server running on localhost (the same machine) at port 3333. It engages in a conversation with the server, where the client sends a message, and the server responds to it. This conversation continues until the client sends the message "stop".*