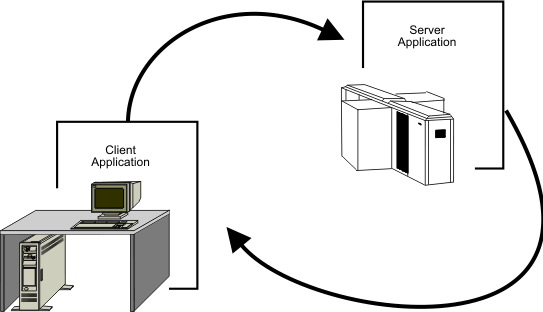
**Experiment no – 01**

**Aim: Write a Program to send and receive message from a client and server.**

**Theory: -**

The Client/Server Model

The term "client/server" describes a type of distributed processing in which an application is divided into two parts, each possibly residing on separate operating systems, but working together to provide a service to the end user. As shown in Figure 1, one part of the application, the client, typically resides on a workstation and requests a service for the end-user. The other part of the application, the server, usually resides on a larger machine, such as a mainframe computer. The server program uses the resources of the mainframe computer to perform services requested by each client.



***Figure 1. The Client/Server Computing Environment***

Depending on how it is designed, a server can process requests from multiple clients concurrently. Generally, there is one server for many clients.

The client is usually the part of the application that is "seen" by the end-user. Therefore, the client half of a client/server application most often resides on a workstation, where the end-user can interact with the application through the workstation operating system's graphical user interface.

Servers, on the other hand, are usually transparent to the end-user. That is, the person who sits at the workstation only perceives the client half of the application, the part that displays the information (though it was actually retrieved by a remote server). Because there is only one server for a given set of clients, servers provide an ideal way of managing shared access to system resources, such as data sets. For this reason, servers are likely to reside on larger machines such as z/OS mainframe computers.

Usually, the same person writes both the client and server parts of a client/server application.

**Program: -**

**Server.java**

import java.io.\*;

import java.net.\*;

class Server {

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

        ServerSocket ss = new ServerSocket(2222);

        Socket s = ss.accept();

        InputStreamReader isr = new InputStreamReader(System.in);

        BufferedReader keyboard = new BufferedReader(isr);

        BufferedReader br1 = new BufferedReader(new

InputStreamReader(s.getInputStream()));

        PrintWriter pw = new PrintWriter((s.getOutputStream()));

        while (true) {

            System.out.println("Enter Send/Quit/Receive");

            String k = keyboard.readLine();

            switch (k.charAt(0)) {

                case 'S':

                case 's':

                    String sendmsg = keyboard.readLine();

                    pw.println(sendmsg);

                    pw.flush();

                    break;

                case 'R':

                case 'r':

                    String recvmsg = br1.readLine();

                    System.out.println(recvmsg);

                    break;

                case 'Q':

                case 'q':

                    System.exit(0);

            }

        }

    }

}

**Client.java**

import java.io.\*;

import java.net.\*;

class Client {

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

        InputStreamReader isr = new InputStreamReader(System.in);

        BufferedReader keyboard = new BufferedReader(isr);

        Socket s = new Socket("localhost", 2222);

        BufferedReader br1 = new BufferedReader(new InputStreamReader(s.getInputStream()));

        PrintWriter pw = new PrintWriter((s.getOutputStream()));

        while (true) {

            System.out.println("Enter Send/Quit/Receive");

            String k = keyboard.readLine();

            switch (k.charAt(0)) {

                case 'S':

                case 's':

                    String sendmsg = keyboard.readLine();

                    pw.println(sendmsg);

                    pw.flush();

                    break;

                case 'R':

                case 'r':

                    String recvmsg = br1.readLine();

                    System.out.println(recvmsg);

                    break;

                case 'Q':

                case 'q':

                    System.exit(0);

            }

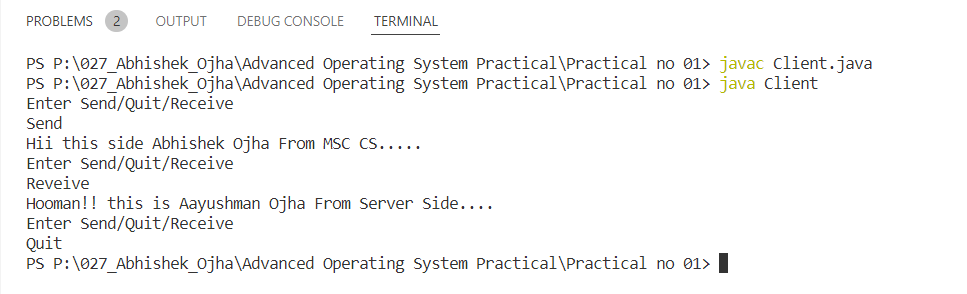
        }

    }

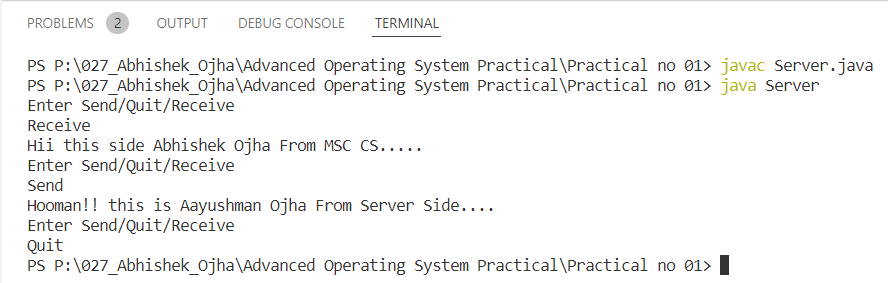
}

**Output:**

**Client**

****

**Server**



**Conclusion:**

Successfully demonstrated a client server program to send and receive a message.