Ex No: 3

Date:

IMPLEMENTATION OF TCP USING SOCKET PROGRAMMING

AIM:

To understand the concept of socket programming and implement it using Java Programming in command prompt.

THEORY:

The Transmission Control Protocol (TCP) is a connection-oriented protocol used to establish reliable communication between devices over a network. It ensures the ordered delivery of data through mechanisms like the 3-way handshake, error detection, and retransmissions.

In socket programming, TCP is implemented using sockets to establish a communication link between a client and a server. The server listens for incoming client requests on a specific port, while the client initiates the connection. TCP ensures:

- **Connection Establishment:** Using the 3-way handshake, where SYN (synchronize), SYN-ACK (synchronize-acknowledge), and ACK (acknowledge) packets are exchanged.
- **Reliable Data Transfer:** Data is divided into segments, numbered sequentially, and acknowledged by the receiver.
- **Error Handling and Retransmission:** Lost or corrupted packets are retransmitted to maintain data integrity.

Socket programming typically involves the following steps:

1. **Server-Side:**
    - Create a socket.
    - Bind the socket to a port.
    - Listen for incoming connections.
    - Accept a connection and exchange data.
2. **Client-Side:**
    - Create a socket.

- Connect to the server.
- Send and receive data.

By leveraging TCP sockets, applications like web browsers, file transfer tools, and email clients achieve reliable communication over the internet.

ALGORITHM:

1. Import the necessary Libraries which include util,io,net.
2. Inside main method use a try block for writing the code establishing the connection.
3. For Server side connection, we create object of the ServerSocket class and pass the localhost port number as argument for the constructor.
4. Then we use the reference of Socket class top accept method of ServerSocket class the client's requests.
5. Using buffered reader we get the input from the input stream and user's input.
6. Using while loop we receive all the messages and print them.
7. If the message given is "QUIT" then we end the while loop and close the connection.
8. In Client's side connection we create object of Socket class and give the name 'localhost' and port number as arguments for the constructor.
9. Using buffered reader we get the input from the input stream and user's input.
10. Using while loop we receive all the messages and print them.
11. If the message given is "QUIT" then we end the while loop and close the connection.

PROGRAM:

SERVER PROGRAM:

```
import java.io.*;
import java.net.*;

class tcpServer {
   public static void main(String[] args) {
      try {
         System.out.println("Server is Running!");
         ServerSocket s1 = new ServerSocket(8081);
         Socket s = s1.accept();
         System.out.println("Client connected!");

         BufferedReader br = new BufferedReader(new
InputStreamReader(s.getInputStream()));
         PrintWriter out = new PrintWriter(s.getOutputStream(), true);
```

```
            String msg = "";
            while (true) {
                msg = br.readLine();
                if (msg == null) break; // Handle client disconnect
                System.out.println("From client: " + msg);
                out.println("Message received: " + msg); // Responding to the client
                if (msg.equals("quit")) break;
            }

            br.close();
            out.close();
            s.close();
            s1.close();
            System.out.println("TERMINATED!");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

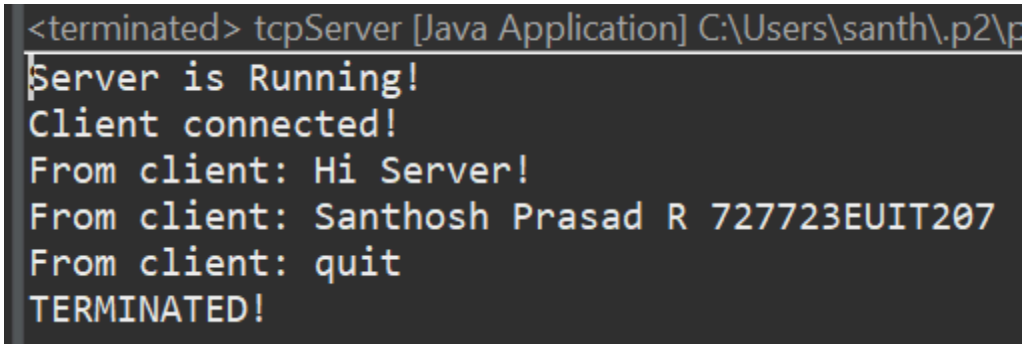
CLIENT PROGRAM:

```
import java.io.*;
import java.net.*;
class tcpClient {
    public static void main(String[] args) {
        try {
            System.out.println("Client is running!");
            Socket s = new Socket("localhost", 8081);

            BufferedReader in = new BufferedReader(new
InputStreamReader(System.in));
            BufferedReader br = new BufferedReader(new
InputStreamReader(s.getInputStream()));
            PrintWriter p = new PrintWriter(s.getOutputStream(), true);

            String msg = "";
            while (true) {
                System.out.print("Enter message: "); // Prompt user for input
                msg = in.readLine();
                p.println(msg);
                System.out.println("Server says: " + br.readLine());
                if (msg.equals("quit")) break;
```

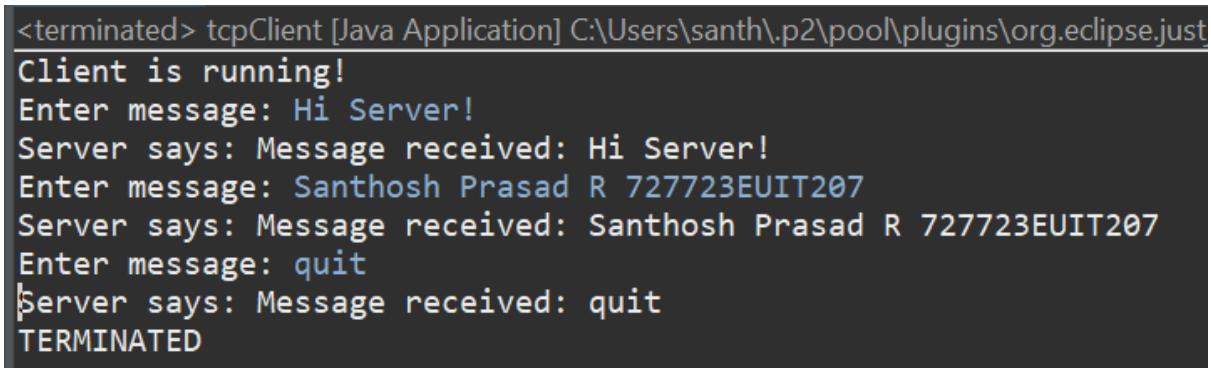
```
            }

            p.close();
            br.close();
            s.close();
            System.out.println("TERMINATED");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

SCREENSHOT OF OUTPUT:

Server:



Client:



RESULT:

Thus, the concept of Socket Programming is studied and implemented using Java Programming and is successfully executed in Eclipse IDE and verified.