**NAME: VASANTH T ROLL NO: 18EUCS125**

# DATA COMMUNICATION AND COMPUTER NETWORKS

1. **The Server application accepts strings from clients and replies withreverse strings. For example, when client sends “HELLO”, Server replies with “OLLEH”. Use TCP Socket to establish connection. Both server and client(s) have to output both sending & receiving strings on the terminal.**

**AIM:**

To implement TCP socket programming to reverse string.

# DESCRIPTION:

A **client** is a computer hardware device or software that accesses a service made available by a server. The server is often (but not always) located on a separate physical computer.

A **server** is a physical computer dedicated to run services to serve the needs of other computers. Depending on the service that is running, it could be a file server, database server, home media server, print server, or web server.

# ALGORITHM:

**TCP Server:**

1. Create a socket
2. Bind it to the operating system 3.Listen over it
3. Accept Connections
4. Recieve data from Client and send it back to client 6.Close the socket

# TCP Client:

1. Create a Socket.
2. Connect to the server using connect()
3. Send data to server and receive data from server 4.Close the socket

# PROGRAM:

**CLIENT:**

**package** lab;

**import** java.net.\*;

**import** java.io.\*;

**import** java.util.Scanner;

**public class** firstclient{

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

InetAddress ia = InetAddress.*getLocalHost*(); Socket cSock = **new** Socket(ia,1234); DataInputStream in = **new** DataInputStream(cSock.getInputStream()); DataOutputStream out = **new** DataOutputStream(cSock.getOutputStream()); System.***out***.println("Please Enter String"); Scanner sc = **new** Scanner(System.***in***);

String inp = sc.nextLine(); out.writeUTF(inp); System.***out***.println("Presponse from server"); System.***out***.println(in.readUTF().toString()); cSock.close();

}

}

# SERVER:

**package** lab; **import** java.net.\*; **import** java.io.\*;

**public class** firstserver{

**public static void** main(String arg[]) **throws** Exception{ ServerSocket server = **new** ServerSocket(1234); System.***out***.println("Server is Waiting");

**while**(**true**){

Socket con = server.accept();

DataInputStream in = **new** DataInputStream(con.getInputStream()); DataOutputStream out = **new** DataOutputStream(con.getOutputStream()); StringBuilder inp = **new** StringBuilder(in.readUTF().toString()); StringBuilder op=inp.reverse();

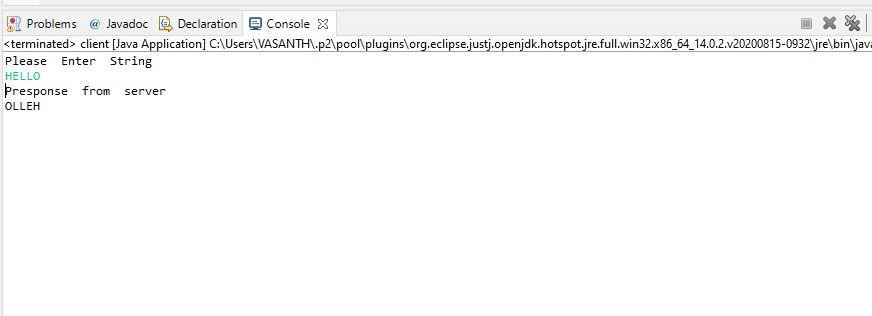
out.writeUTF(op.toString());

}

}

}

# OUTPUT:



**RESULT:**

Thus, the implementation of tcp socket programming is verified .

# Implement the Stop-n-Wait Protocol using TCP (Socket Programming)

**AIM:**

To implement Stop and Wait Protocol using socket programming.

# DESCRIPTION:

* + It is the simplest flow and error control mechanism. Atransmitter sends a frame then stops and waits for an acknowledgment.

Stop-and-Wait ARQ has the following features:

* + The sending device keeps a copy of the sent frame transmitted untilit receives an acknowledgement(ACK)
  + The sender starts a timer when it sends a frame. If an ACK is not received within an allocated time period, the sender resends it
  + Both frames and acknowledgment (ACK) are numbered alternately 0 and 1 (two sequence number only)
  + This numbering allows for identification of frames in case ofduplicate transmission
  + The acknowledgment number defines the number of next expected frame. (frame 0 received ACK 1 is sent)
  + A damage or lost frame treated by the same manner by the receiver
  + If the receiver detects an error in the received frame, or receivesa frame out of order it simply discards the frame
  + The receiver send only positive ACK for frames received safe; it is

silent about the frames damage or lost.

* + The sender has a control variable S that holds the number of most recently sent frame (0 or 1). The receiver has control variable R, that holds the number of the next frame expected (0,or 1)

# ALGORITHM:

* + Transmit all frames in the sender’s window (no more than from SLto SU−1).
  + Whenever the receiver gets a frame in its window:
* It generates an ACK for the highest frame correctly received (same as the frame for protocol 5).
* If the frame RL has been received it passes RL to the host and bumps RL and RU (advances the window).
  + Whenever the receiver gets a damaged frame or a frame not within its window it generates a NAK for one less than the frame expected (RL − 1) (only for protocol 6).
  + Whenever the sender receives an ACK for a frame within its window, it marks that frame as having been correctly sent and received. If SL is ACKed then increment SL and SU (advance the sender’s window) and transmit SU−1 (last previously unsent frame).
  + Whenever a timer goes off, retransmit the corresponding frame.

# Description on packages, classes, objects, constructors and methods: Client:

In Package java.net

* java.net.Socket
* Implements client sockets (also called just “sockets”).
* An endpoint for communication between two machines.
* Constructor and Methods
* Socket(String host, int port): Creates a stream socket and connectsit to the specified port number on the named host.
* ObjectOutputStream class- An ObjectOutputStream writes primitive data types and graphs of Java objects to an OutputStream.
* readLine()-Reads a line of text.
* writeObject()-Write the specified object to the ObjectOutputStream.

# Server:

* java.net.ServerSocket
* Implements server sockets.
* Waits for requests to come in over the network.
* Performs some operation based on the request.
* Constructor and Methods
* ServerSocket(int port)
* Socket Accept(): Listens for a connection to be made to this socket and accepts it. This method blocks until a connectionis made.
* ObjectInputStream-An ObjectInputStream deserializes primitive data and objects previouslywritten using anObjectOutputStream.
* ObjectOutputStream-An ObjectOutputStream writes primitive data types and graphs of Java objectsto an OutputStream.
* getInputStream()-Returns an input stream for this socket.
* getOutputStream()-Returns an output stream for this socket.
* ProcessBuilder-This class is used to create operatingsystem processes.
* command(String... command)- Sets this process builder'soperating system program andarguments.
* directory(File directory)-Sets this process builder's working directory.Subprocesses subsequently started by this object's start().
* readLine()-Reads a line of text.
* waitFor()-Causes the current thread to wait, if necessary, until theprocess represented by this Process object hasterminated.

# PROGRAM:

**CLIENT:**

package lab; import java.io.\*; import java.net.\*;

public class secclient {

public static void main(String[] args) { try {

System.out.println("SERVER"); String frame = null; String ack = null;

Socket con = new Socket("localhost", 333);

System.out.println("connected with server - IP:" + con.getInetAddress().getHostAddress()); ObjectOutputStream out = new ObjectOutputStream(con.getOutputStream());

ObjectInputStream in = new ObjectInputStream(con.getInputStream());

frame = "hello"; out.writeObject(Integer.toString(frame.length())); String subframe

= null;

int frameno = 0;

for (int i = 0; i < frame.length(); i++) { subframe = frame.substring(i, i + 1); out.writeObject("frame" + frameno + ":" + subframe);

System.out.println("frame" + frameno + "sent to server: " + subframe);

if (frameno == 0) frameno = 1; else

frameno = 0;

ack = (String) in.readObject(); System.out.println("Ack received from server:" + ack);

}

con.close();

} catch (Exception e) { System.out.println("socket error:" + e);

}

}

}

**SERVER:**

package lab; import java.io.\*; import java.net.\*;

public class secserver {

public static void main(String[] args) { try {

System.out.println("SERVER"); String frame = null; String ack = null;

ServerSocket ss = new ServerSocket(333); System.out.println("Waiting for connection");

Socket con = ss.accept(); System.out.println("Connected with client - IP:" + con.getInetAddress().getHostAddress()); ObjectInputStream in = new ObjectInputStream(con.getInputStream()); ObjectOutputStream out = new ObjectOutputStream(con.getOutputStream()); String framelength = (String) in.readObject();

int ackno = 0;

for (int i = 0; i < Integer.parseInt(framelength); i++) { frame = (String)

in.readObject();

System.out.println("Frame received from client" + frame);

if (ackno == 0) ackno = 1; else

ackno = 0;

ack = "ack" + ackno; out.writeObject(ack); System.out.println("Acknowledgement sent to client:"

+ ack);

}

in.close();

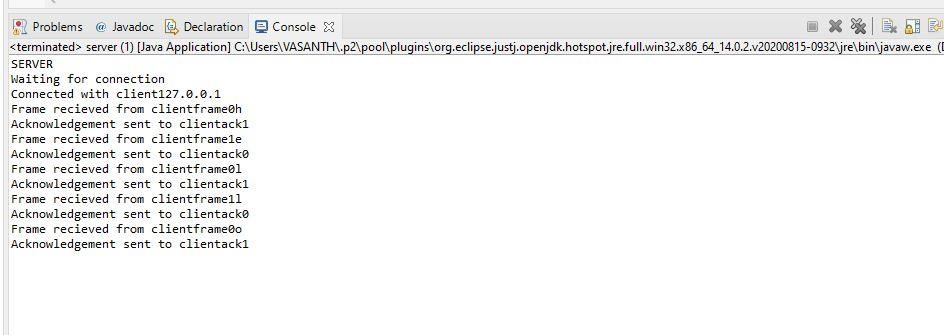
out.close();

ss.close();

} catch (Exception e) { System.out.println("error" + e);

}}}

# OUTPUT:



**RESULT:**

Thus, the implementation of Stop and Wait Protocol using socket programming.