**Experiment 06**

Name : Khan Mohd Aaquib Sameer

Roll No : 22CO21

Batch : 2

Aim : To perform socket programming

**Theory :**

**What is a Socket?**

A socket is one endpoint of a two-way communication link between two programs

running on the network. An endpoint is a combination of an IP address and a port

number. The socket is bound to a port number so that the TCP layer can identify the

application that data is destined to be sent.

java.net.Socket and java.net.ServerSocket are the java classes that implements

Socket and Socket server.

**Socket Programming**

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.

When the connection is made, the server creates a socket object on its end of the

communication. The client and the server can now communicate by writing to and

reading from the socket.

The java.net.Socket class represents a socket, and the java.net.ServerSocket class

provides a mechanism for the server program to listen for clients and establish

connections with them.

A socket is one of the most fundamental technologies of computer network

programming . It is a way of connecting two nodes on a network to communicate

with each other. Socket-based software usually runs on two separate computers on the

network, but sockets can also be used to communicate locally (interprocess) on a

single computer.

The Java Socket Programming has two sections.

**1. Java Server Socket Program**

**2. Java Client Socket Program**

**//Code :**

**Server.java:**

// A Java program for a Server

import java.net.\*;

import java.io.\*;

public class Server

{

//initialize socket and input stream

private Socket socket = null;

private ServerSocket server = null;

private DataInputStream in = null;

// constructor with port

public Server(int port)

{

// starts server and waits for a connection

try

{

server = new ServerSocket(port);

System.out.println("Server started");

System.out.println("Waiting for a client ...");

socket = server.accept();

System.out.println("Client accepted");

// takes input from the client socket

in = new DataInputStream(

new BufferedInputStream(socket.getInputStream()));

String line = "";

// reads message from client until "Over" is sent

while (!line.equals("Over"))

{

try

{

line = in.readUTF();

System.out.println(line);

}

catch(IOException i)

{

System.out.println(i);

}

}

System.out.println("Closing connection");

// close connection

socket.close();

in.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Server server = new Server(5000);

}

}

**Output:**

**aiktc@CO-LAB1-04:~/Desktop/CNL$ javac Server.java**

**aiktc@CO-LAB1-04:~/Desktop/CNL$ java Server**

**Server started**

**Waiting for a client …**

**After Running the Server.java File , We then run Client.java for client to get accepted .  
Client.java :**// A Java program for a Client

import java.net.\*;

import java.io.\*;

public class Client

{

// initialize socket and input output streams

private Socket socket = null;

private BufferedReader input = null;

private DataOutputStream out = null;

// constructor to put ip address and port

public Client(String address, int port)

{

// establish a connection

try

{

socket = new Socket(address, port);

System.out.println("Connected");

// takes input from terminal

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

// sends output to the socket

out = new DataOutputStream(socket.getOutputStream());

}

catch(UnknownHostException u)

{

System.out.println(u);

}

catch(IOException i)

{

System.out.println(i);

}

// string to read message from input

String line = "";

// keep reading until "Over" is input

while (!line.equals("Over"))

{

try

{

line = input.readLine();

out.writeUTF(line);

}

catch(IOException i)

{

System.out.println(i);

}

}

// close the connection

try

{

input.close();

out.close();

socket.close();

}

catch(IOException i)

{

System.out.println(i);

}

}

public static void main(String args[])

{

Client client = new Client("127.0.0.1", 5000);

}

}

**Output :   
aiktc@CO-LAB1-04:~/Desktop/CNL$ javac Client.java**

**aiktc@CO-LAB1-04:~/Desktop/CNL$ java Client**

**Connected**

**After Client.java shows Connected , Server.java give the Output saying Client Accepted.**

**Output :   
aiktc@CO-LAB1-04:~/Desktop/CNL$ javac Server.java**

**aiktc@CO-LAB1-04:~/Desktop/CNL$ java Server**

**Server started**

**Waiting for a client ...**

**Client accepted**