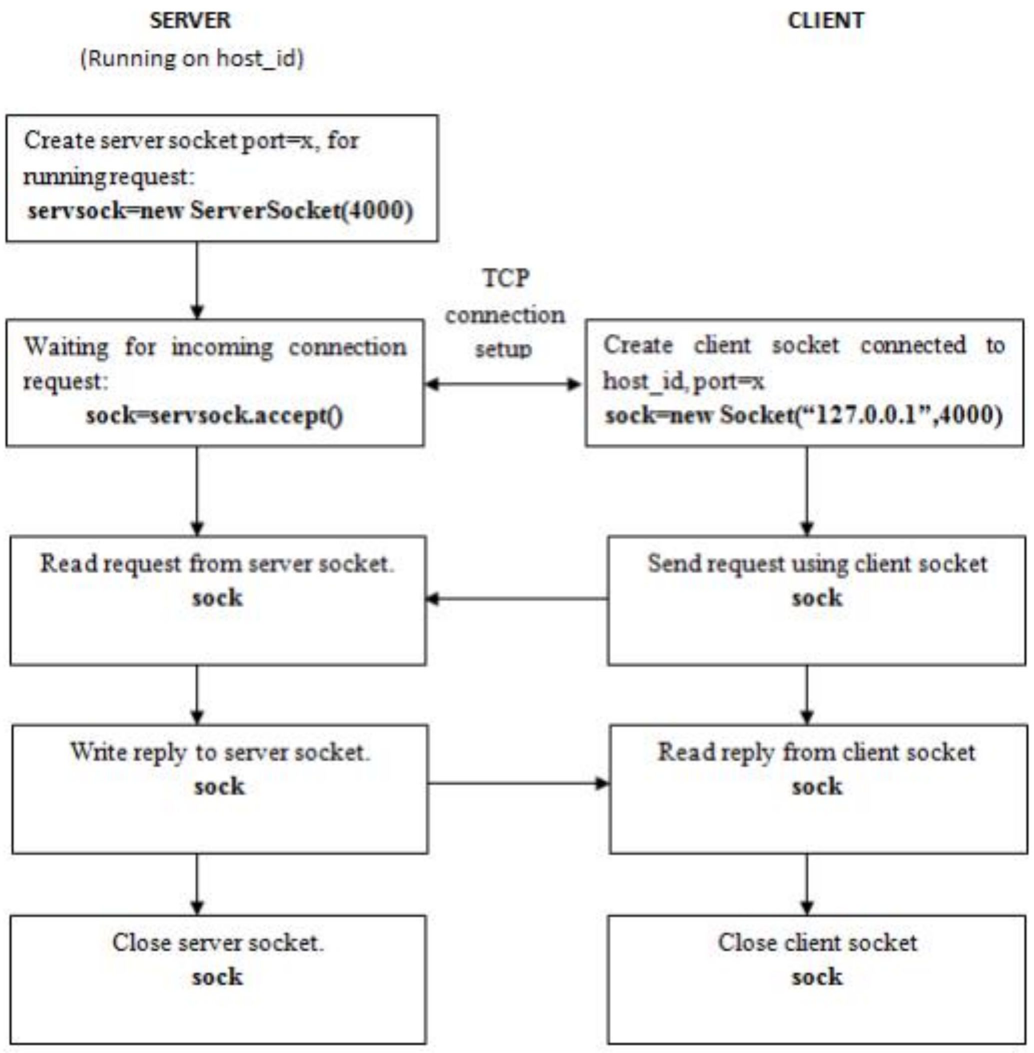
Program- 7

Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.

Socket is an interface which enables the client and the server to communicate and pass on information from one another. 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.


SERVER
(Running on host_id)
CLIENT
Create server socket port=x, for running request:
servsock=new ServerSocket(4000)
TCP connection setup
Waiting for incoming connection request:
sock=servsock.accept()
Create client socket connected to host_id, port=x
sock=new Socket("127.0.0.1",4000)
Read request from server socket.
sock
Send request using client socket
sock
Write reply to server socket.
sock
Read reply from client socket
sock
Close server socket.
sock
Close client socket
sock

Client program

```
import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.EOFException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.net.Socket;
import java.util.Scanner;
class Client
{
public static void main(String args[])throws Exception
{
String address = "";
Scanner sc=new Scanner(System.in);
System.out.println("Enter Server Address: ");
address=sc.nextLine();
//create the socket on port 5000
Socket s=new Socket(address,5000);
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Send Get to start...");
String str="",filename="";
try
{
while(!str.equals("start"))
str=br.readLine();
dout.writeUTF(str);
dout.flush();
filename=din.readUTF();
System.out.println("Receving file: "+filename);
filename="client"+filename;
System.out.println("Saving as file: "+filename);
long sz=Long.parseLong(din.readUTF());
System.out.println ("File Size: "+(sz/(1024*1024))+" MB");
byte b[]=new byte [1024];
System.out.println("Receving file..");
FileOutputStream fos=new FileOutputStream(new File(filename),true);
long bytesRead;
do
{
bytesRead = din.read(b, 0, b.length);
fos.write(b,0,b.length);
}while(!(bytesRead<1024));
System.out.println("Completed");
fos.close();
dout.close();
s.close();
}
catch(EOFException e)
{
}
}
}
```

SERVER PROGRAM

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
public class Server
    public static void main(String[] args) {
        String filename;
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter File Name: ");
        filename = sc.nextLine();
        sc.close();
        try (ServerSocket serverSocket = new ServerSocket(5000)) {
            System.out.println("Server started and waiting for a request...");
            while (true) {
                try (Socket socket = serverSocket.accept();
                     DataInputStream din = new DataInputStream(socket.getInputStream());
                     DataOutputStream dout = new DataOutputStream(socket.getOutputStream()))
{
                  System.out.println("Connected with " + socket.getInetAddress().toString())
                    String request = din.readUTF();
                    if (request.equals("start")) {
                        System.out.println("Received 'start' request.");
                        File file = new File(filename);
                        if (!file.exists()) {
                            System.out.println("File not found: " + filename);
                            dout.writeUTF("File not found");
                            dout.flush();
                            continue;  }
            // Send the filename and file size
                        dout.writeUTF(filename);
                        dout.flush();
                        long fileSize = file.length();
                        dout.writeUTF(Long.toString(fileSize));
                        dout.flush();
                        System.out.println("Sending file: " + filename);
                        System.out.println("File size: " + (fileSize / (1024 * 1024)) + " MB");
                        try (FileInputStream fin = new FileInputStream(file)) {
                            byte[] buffer = new byte[1024];
                            int bytesRead;
                            while ((bytesRead = fin.read(buffer)) != -1) {
                               dout.write(buffer, 0, bytesRead);
                               dout.flush();
                            }
                        }
                        System.out.println("File sent successfully.");
                    } else {
                        dout.writeUTF("Invalid request");
                        dout.flush();
                    }
                 } catch (IOException e) {
                    e.printStackTrace();
                    System.out.println("An error occurred while handling the client request.");
                 }
            }
              } catch (IOException e) {
            e.printStackTrace();
            System.out.println("Failed to start the server.");
        } }}
```

Note: Create two different files Client.java and Server.java. Follow the steps given:

1. Open a terminal run the server program and provide the filename to send

2. Open one more terminal run the client program and provide the IP address of the server. We can give localhost address "127.0.0.1" as it is running on same machine or give the IP address of the machine.

3. Send any start bit to start sending file.

4. Refer https://www.tutorialspoint.com/java/java_networking.htm for all the parameters, methods description in socket communication.

OUTPUT:

AT SERVER SIDE

```
user@user-OptiPlex-3050:~/Desktop$ javac Server.java
user@user-OptiPlex-3050:~/Desktop$ java Server
Enter File Name:
flower.jpg
Server started and waiting for a request...
Connected with /127.0.0.1
Received &apos;start&apos; request.
Sending file: flower.jpg
File size: 0 MB
File sent successfully
```

AT CLIENT SIDE

```
user@user-OptiPlex-3050:~/Desktop$ javac Client.java
user@user-OptiPlex-3050:~/Desktop$ java Client
Enter Server Address:
127.0.0.1
Send Get to start...
start
Receving file: flower.jpg
Saving as file: clientflower.jpg
File Size: 0 MB
Receving file..
Completed
```