BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT
YELAHANKA, BENGALURU - 560064

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Report on TCP/IP Sockets Programming

Name	Raghavendra K M
USN	1BY18IS093
Course Code	18CSL57
Course Name	Computer Network Laboratory
Faculty	Prof. Gireesh babu C N
Title	TCP/IP Sockets Programming
Date	22-09-2020

Signature of a Student **Signature of a Faculty**

Aim of the experiment:

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.

TCP/IP specifies how data is exchanged over the internet by provide end-to-end communications that identify how it should be broken into packets, addressed, transmitted, routed and received at the destination

The java.net package of the J2SE APIs contains a collection of classes and interfaces that provide the low-level communication details, allowing you to write programs that focus on solving the problem at hand.

The java.net package provides support for the two common network protocols −

- **TCP** − TCP stands for Transmission Control Protocol, which allows for reliable communication between two applications. TCP is typically used over the Internet Protocol, which is referred to as TCP/IP.
- **UDP** − UDP stands for User Datagram Protocol, a connection-less protocol that allows for packets of data to be transmitted between applications.

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.

The following steps occur when establishing a TCP connection between two computers using sockets −

- The server instantiates a ServerSocket object, denoting which port number communication is to occur on.
- The server invokes the accept() method of the ServerSocket class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a Socket object, specifying the server name and the port number to connect to.
- The constructor of the Socket class attempts to connect the client to the specified server and the port number. If communication is established, the client now has a Socket object capable of communicating with the server.

- On the server side, the accept() method returns a reference to a new socket on the server that is connected to the client's socket.

After the connections are established, communication can occur using I/O streams. Each socket has both an OutputStream and an InputStream. The client's OutputStream is connected to the server's InputStream, and the client's InputStream is connected to the server's OutputStream.

TCP is a two-way communication protocol, hence data can be sent across both streams at the same time. Following are the useful classes providing complete set of methods to implement sockets.

ServerSocket Class Methods

The **java.net.ServerSocket** class is used by server applications to obtain a port and listen for client requests.

The ServerSocket class has four constructors –

The ServerSocket constructor used in the program is:

Sr.No.	Method & Description
1	**public ServerSocket(int port) throws IOException** Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.

If the ServerSocket constructor does not throw an exception, it means that your application has successfully bound to the specified port and is ready for client requests.

Following method of the ServerSocket class is used in th program

Sr.No.	Method & Description
1	**public Socket accept() throws IOException** Waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out, assuming that the time-out value has been set using the setSoTimeout() method. Otherwise, this method blocks indefinitely.

When the ServerSocket invokes accept(), the method does not return until a client connects. After a client does connect, the ServerSocket creates a new Socket on an unspecified port and returns a reference to this new Socket. A TCP connection now exists between the client and the server, and communication can begin.

Socket Class Methods

The **java.net.Socket** class represents the socket that both the client and the server use to communicate with each other. The client obtains a Socket object by instantiating one, whereas the server obtains a Socket object from the return value of the accept() method.

The Socket class has five constructors that a client uses to connect to a server −

Sr.No.	Method & Description
1	**public Socket(String host, int port) throws UnknownHostException, IOException.** This method attempts to connect to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.

When the Socket constructor returns, it does not simply instantiate a Socket object but it actually attempts to connect to the specified server and port.

Some methods of interest in the Socket class are listed here. Notice that both the client and the server have a Socket object, so these methods can be invoked by both the client and the server.

Sr.No.	Method & Description
1	**public InputStream getInputStream() throws IOException** Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket.
2	**public OutputStream getOutputStream() throws IOException** Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket.
3	**public void close() throws IOException** Closes the socket, which makes this Socket object no longer capable of connecting again to any server.

Source code:

TCPServer.java

```
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
public class TCPServer
{
  public static void main(String args[]) throws Exception
  {
     ServerSocket sersock = new ServerSocket(4000);
     System.out.println("Server ready for connection");
     Socket sock = sersock.accept();
     System.out.println("Connection successful | wating for file
     name");
     InputStream istream = sock.getInputStream( );
     BufferedReader br =new BufferedReader(new
     InputStreamReader(istream));
     String fname = br.readLine( );
     BufferedReader contentRead = new BufferedReader(new
     FileReader(fname) );
     OutputStream ostream = sock.getOutputStream( );
     PrintWriter pwrite = new PrintWriter(ostream, true);
     String str;
     while((str = contentRead.readLine()) !=  null)
     {
          pwrite.println(str);
     }
     System.out.println("File Contents sent successfully");

     sock.close();  sersock.close();
     pwrite.close();  br.close(); contentRead.close();
  }
}
```

TCPClient.java

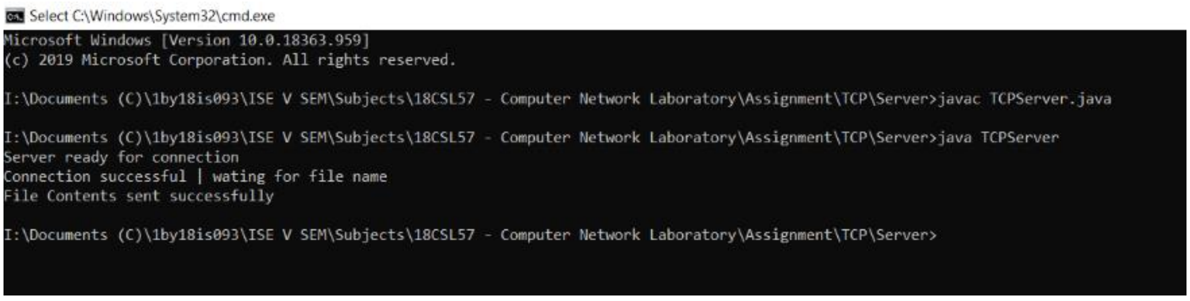
```
import java.net.*;
import java.io.*;
public class TCPClient
{
  public static void main( String args[ ] ) throws Exception
  {
     Socket sock = new Socket( "127.0.0.1", 4000);
     System.out.print("Enter the file name\n");
     BufferedReader br = new BufferedReader(new
     InputStreamReader(System.in));
     String fname = br.readLine();
     OutputStream  ostream = sock.getOutputStream( );
     PrintWriter pwrite = new PrintWriter(ostream, true);
     pwrite.println(fname);

     InputStream istream = sock.getInputStream();
     BufferedReader socketRead = new BufferedReader(new
     InputStreamReader(istream));

     String str;
     while((str = socketRead.readLine())  !=  null) // reading
                                                        line-by-line
     {
         System.out.println(str);
     }
     pwrite.close(); socketRead.close(); br.close();
     sock.close();
  }
}
```

Outputs:

TCP Server



Select C:\Windows\System32\cmd.exe

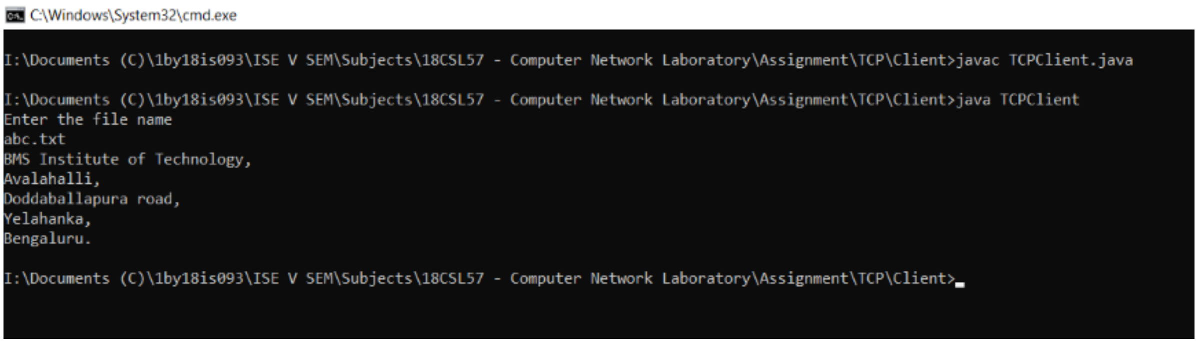
```
Microsoft Windows [Version 10.0.18363.959]
(c) 2019 Microsoft Corporation. All rights reserved.

I:\Documents (C)\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Assignment\TCP\Server>javac TCPServer.java

I:\Documents (C)\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Assignment\TCP\Server>java TCPServer
Server ready for connection
Connection successful | wating for file name
File Contents sent successfully

I:\Documents (C)\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Assignment\TCP\Server>
```

TCP Client



C:\Windows\System32\cmd.exe

```
I:\Documents (C)\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Assignment\TCP\Client>javac TCPClient.java

I:\Documents (C)\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Assignment\TCP\Client>java TCPClient
Enter the file name
abc.txt
BMS Institute of Technology,
Avalahalli,
Doddaballapura road,
Yelahanka,
Bengaluru.

I:\Documents (C)\1by18is093\ISE V SEM\Subjects\18CSL57 - Computer Network Laboratory\Assignment\TCP\Client>
```