Ch7 Network & socket programming

(7.1) Networking Basics (7.2) Introduction to socket
(7.3) Socket programming (7.4) Understanding ports
(7.5) Networking classes in java (7.6) creating own server & client in java.
(7.7) Creating multithread Java server (7.8) URL & URL connection class.

Network programming can be defined as the act of writing computer codes to make a program that communicates with the program or system over the internet.

It is the process of writing program that can execute across multiple devices in which all the devices are connected with each other through internet.

java.net package of J2SE API provide collection of classes & interface for network programming. java.net package provide two common network protocol:
(a) TCP (Transmission Control protocol)
(b) UDP (User Datagram protocol)

TCP which stands for Transmission Control protocol is a network protocol that allows for reliable communication between 2 apps, setting up a connection between application. Connection oriented programming uses TCP as communication protocol. TCP is usually used over IP which is reffered as TCP/IP

UDP which stands for User Datagram Protocol is a network protocol that allows for communication without setting up connection between application, causing high chances of potential data loss & breach. It is a connection less protocol where data are sends in form of package without maintaining any sequence or order, which makes receiver to manage the sequence at last.

Socket & socket programming

↳ Socket can be defined as the communication end point that allows two computer or processes to exchange data & information with each other over a network.

↳ Its a communication mechanism, between 2 computers using TCP or UDP. Socket and ServerSocket class is used for TCP based socket programming & similarly DatagramSocket & DatagramPacket are the classes used for connection less (UDP) socket programming.

⇒ Socket comes under pair, where one socket is used by computer / program initiating the communication (i.e. client) & another socket is used by receiver (server).

⇒ Socket are identified using IP Address, which helps to identifies the computer, and port number that helps in finding which specific program is running on that computer.

Hence, Socket programming involves writing codes, that create, configure & uses socket for communication between computers & devices, where the code specifies TCP or UDP used by socket, setting up connection & sending, receiving data.

Ports: In the world of computer science, network programming, ports can be defined as the numeric value which helps to identify the specific program, process, application running on any system. For eg:- MySql Local Database runs on a port number of 3306 in our Laptop.

Ports serves as a system's program location finder and enables multiple application to run simultaneously there are 3 types of ports ranging from 0 to 65535

(a) well known ports (b) Registered ports
(c) Dynamic or private port.

port number 0 - 1023 is known to be well known port or system port. They are assigned to specific apps or program by IANA (Internet Assigned number authority) and can't be used.

Registered port are the port from 1024 - 49151 for apps or program by IANA. eg: port 80 for HTPP & port 25 for SMTP.

Dynamic or private port is used by O.S for temporary connection whenever needed 49152 - 65535.

P no. 0

(2.9)

Networking classes in Java

Networking classes are the set of classes & interfaces provided by java.net package of Java 2 SE API, for creating networking apps & programs that runs and connects with other apps over the internet.

Some of the networking classes provided by java.net package are:

Classes

(a) Socket (b) ServerSocket (c) URL (d) URLConnection
(e) InetAddress (f) DatagramPacket (g) DatagramSocket.

Class

Interface

(1) CookiePolicy (2) CookieStore (3) Protocol family
(4) SocketOption.

Socket : This class represents the client side end point for network communication. It is the socket for client who initiates the rqst.

ServerSocket : This class represents the serverside end point for network communication. It listeners to the request made by client side socket.

URL: This class represents the link or URL. It points to the resources on the internet. It consists of multiple methods, which gives further information about URL.

URL connection class : This class represents the connection communication between URL & application. It comes under URL and the open connection method is responsible

for URLConnection Class.

(a) InetAddress: This class is responsible for the task related with IP Address: getting hostname, DNS lookup, etc is carried out by the methods provided InetAddress class.

URL & URL connection class

↳ URL which stands for uniform Resource Locator represents the resource located in the www such as webpage or FTP directory.

URL class is a class that is provided by java.lang package and responsible for representing URL. It's a pointer that points to the resource of www.

URL is divided into following parts:

file

eg: https://www.example.com:3000/search?q=123#section_id

protocol — hostname — portnumber — path — query — Ref

protocol://hostname:portnumber/path?query#reference

There are different methods of URL class

(a) public String getQuery(): Returns the query of the URL

(b) public String getHost():

(b) Public String getHost(): ⇒ Returns the hostname of the URL

(c) public string getProtocol(): Returns the protocol that the URL uses.

(d) public int getPort(): Returns the port number of the URL where the application is running

(e) public string getPath(): Returns the path of the URL

(f) public string getRef(): Returns the reference of the URL.

(g) public string getAuthority(): It returns the authority part of the URL or returns empty null if it is

(h) public int getDefaultPort(): Returns the default port for the protocol of the URL.

(i) public string getFile(): It returns the filename of the URL

(j) public URL toURI(): It returns equivalent URI of URL.

Q:

Eg.

```
import java.net.*;
public class URLDemo
{
  public static void main (String[] args)
  { try{
    URL url= new URL("https://www.youtube.com:3856/search?q=123#demo");

    System.out.println("Protocol : "+url.getProtocol());
    "    "    "  ("Hostname : "+url.getHost());
    "    "    "  ("portnumber : "+url.getPort());
System.out.println("Path : "+url.getPath());
    "    "    "  ("Query : "+url.getQuery());
    "    "    "  ("Reference : "+url.getRef());
  }
  catch (Exception e)
  {
    e.printStackTrace();
  }
}
}
```

URL connection class

↳ Java URLConnection class represents the communication link between a URL and applications. The openConnection() method of URL class returns back the URLConnection class's Objects.

URLConnection class can be used to read & write data to a specific resource specified by URL.

(13)

Note: If we connect to a URL whose protocol is HTTP, the ~~openConnection()~~ method openConnection() returns the HTTPURLConnection object.

If we connect to a URL whose protocol is FTP, openConnection() returns the ftpURLConnection object.

Eg:

Q Bank ch7 Networking

22 fall PC TCP vs UDP

TCP	UDP
@ TCP which stands for Transmission Control protocol is a network protocol that allows for reliable communication between apps, hence setting up a connection between them.	=> UDP which stands Users Datagram protocol is a network protocol that don't need a connection setup for communication, hence known to be less reliable.
=> Also known to be connection oriented programming protocol	=> Also known to be connectionless protocol.
=> It is comparitively slower since connection takes times.	=> Comparitively faster as, no connection need to be set up.
=> It is more reliable, hence making no chance of data loss,	=> It is more vulnerable cause occasional chance of data loss.

⇒ It is used in application, where data consistency is highly needed like in financial banking sector.

⇒ It is used by HTTP, HTTPS, FTP, SMTP

⇒ TCP doesn't need to rearrange data in packets since order will be maintained in sequence.

⇒ TCP does flow control

⇒ It is used in applications where occasional data loss is not big issue.

⇒ UDP is used by DNS, DHCP, VoIP etc

⇒ UDP needs to rearrange the data packets since the order will be disordered

⇒ UDP doesn't have option for flow control.

Q Create a TCP client/server program in which multiple users can send and receive message from each other.

⇒

Q Creating a simple client server program in Java where client send a range and server sends the fibonacci series till that range

↳

file EchoServer.java

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

  public class EchoServer
  {
    public static void main(String[] args)
    {
```

```
System.out.println("Server is started");
try{
ServerSocket ss = new ServerSocket(9806); // 9806 is portnumber
Socket soc= ss.accept();
System.out.println("connection is established");

}catch(Exception e)
{
e.printStackTrace();
}

}
}
```

File EchoClient.java

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

public class echoclient{
  public static void main(String[] args)
  {
    System.out.println("Client started");
try{ Socket soc= new Socket("localhost", 9806);
    // providing the IP address of server, but here localhost is
       server, & 9806 is portnumber of server */
} catch(Exception e)
{
e.printStackTrace();
}
```

5 0, 1, 1, 2, 3

mple network program to find the fibonnacci series of the range provided by client and server finds it & returns to client.

file ServerSide.java

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

public class ServerSide
{
    public static void main(String[] args)
    { try{
        System.out.println("Server is waiting.....");
        ServerSocket SS= new ServerSocket(9806);
        Socket soc= SS.accept();
        System.out.println("connection is established");

        BufferedReader br= new BufferedReader(soc.getInput
                Stream(), true);
    //Forgetting client input
    BufferedReader br= new BufferedReader(new InputStreamRe.
                ader(soc.getInputStream()));
    String range= br.readLine();
    Int IntRange= Integer.parseInt(range);
```

```
int fn = 0;
int sn = 1;   int tn;
int [] fiboSeries = new int [intRange];
fiboSeries [0] = fn;
fiboSeries [1] = sn;

for( int i = 2; i <= intRange ; i++)
 {
   tn = fn + sn;
   fiboSeries[i] = tn;
   fn = sn;
   sn = tn;
 }

// sending out the array of fibonacci
 PrintWriter out = new PrintWriter (soc.getOutputStream(),
                                    true);
  out.println(fiboSeries);
} catch (Exception e)
 {
  e.printStackTrace();
 }
}
}
```

```
// File ClientSide.java
import java.net.*;
import java.io.*;
import java.util.*;

public class ClientSide
{
  public static void main(String[] args)
  {
    System.out.println("Client started");
    Socket soc = new Socket("localhost", 9808);

    // Scanner sc = new Scanner(System.in);
    System.out.println("enter the range: ");
    BufferedReader br = new BufferedReader(new
        InputStreamReader(System.in));
    String range = br.readLine();

    // For providing the input to server
    PrintWriter out = new PrintWriter(soc.getOutputStream(),
        true);
    out.println(range);

    // Receiving the fibo series
    BufferedReader br = new BufferedReader(
      new InputStreamReader(soc.getInputStream()));

    String fiboArray = br.readLine();
    String[] fiboArrays = fiboArray.split(", ");
```

```
for (String elements: fiboArrays)
{
  System.out.println(Elements);
}
} catch (Exception e)
  {
    e.printStackTrace();
  }
}
}
```