5. Basics of network programming

Networking
Networking is the interconnection of computers used for information sharing & Resource sharing

Internet protocol:
That breaks data into small pieces & packates and sends it to specific address over a network

TCP
Transmission Control protol:
Connection oriented

UDP
user Datagram protocol:
Connection less
Fast

Client
client is a machine that wants to gain access to particular resource of the server

Server
Server is any thing that has some resources to share

The client & Server communicate over a network through protocol

Internet address

Internet Address is a number that uniquely identify each computer or the internet

there are two internet addressing scheme available

1] IPV4 (32-bit)
2] IPV6 (128-bit)

There are 5 network classes

Class A
Class B
Class C
Class D
Class E

Class Inet Address

This class deals with the internet address an object of this class store internet address

Methods :

1] Static ⇒ InetAddress getLocalHost() throws unknownHostException

1] getLocalHost()

It returns the internet Address of local host

2] getByName()

It returns the internet Address of host name pass to it.

3] getAllByName():

It returns array of Internet address that represent all of the address that a particular host name.

Classes

1] class Java.net.Socket

It represent the Socket that both client and Server uses to communicate with each other.

⇒ constructor

1] Socket()

2] Socket(InetAddress address, int port)

⇒ methods:-

1] Void Connect(Socket Addressend point)

2] Void close()
3] Inet Address getInetAddress()

Constructor:

1] It creates an unconnected socket

2] It creates socket & connect its to the specified port number at specified Internet address.

methods:

1] It connect socket to the server
2] It closes the socket
3] It return the address to which socket is connected.

⇒ Steps involves creating the client application

1] Create a socket object
2] Create input output stream for communicating with the server
3] Sent request parameter to server
4] Wait for the servers Response and Receive the Response
5] Close the socket when done

2] Class java.net.ServerSocket
This class is used by Server application to obtion a port & listen for client request

Constructor :

1) ServerSocket()

It creates an unbound Server Socked
clients, to this socket using acceptmethod
~~client~~ binds

2) ServerSocket(Int port)
It creates Server Socket bound to the specific port.

Method:

accept() :
This method listens for connection Request to be made to this Socket and accept it

Steps to create Simple Server Program in Java.

1] It creates Server Socket object
2] accept client request
3] create iostreams
4] Accept request parameter from client
5] Responsed back to the client
6] Close the socket.

Ex:

myserver.java

```
import java.io.*;
import java.net.*;
public class myserver
{
public static void main (String args[])
{
try
{
ServerSocket ss = new ServerSocket (6666);
Socket s = ss.accept();
DataInputStream dis = new DataInput
                      Stream (s.getInputStream());
String str = (String) dis.readUTF();
System.out.println ("message =" + str);
ss.close();
}
Catch (Exception e)
{
```

```
System.out.println(e);
}
}
```

myclient.java

```
import java.io.*;
import java.net.*;
public class ~~myclient~~ socket
{
psvm
{
try
{
Socket s = new Socket("localhost", 6666);
DataOutputStream dout = new DataOutputStream
                        (s.getOutputStream());
dout.writeUTF("Hello UDP");
dout.close();
s.close();
}
}
Catch(Exception e)
{
System.out.println(e);
}
}
```

output:

Javac myserver.java
Javac myclient.java
Java myserver
Java myclient:

msg = Hello Server

- Class URL

It Represent uniform Resource locator

A resource can be file or directive

Constructor:

1) URL (String spec)
It created URL object from specified String

Method:

1) gethost()
It returns the host name of invoking URL

2) getport()
It returns the port no of invoking URL

iii] getprotocol()
It returns protocol name of invoking URL

Class URL Connection

onec client next connection to the Remote Server. the role of URL Connection class start.

To commonly used Methods are openConnection

1) Connect method:
It is used to make connection to the Remote object.

2) getContent()
It retrives the content of invoking object.

3) getContentLength()
It returns the length of content of invoking object

Datagram packet

UDP is appropriate for data transfer where does not matter even if a packet is lost during transmission

The datapacket send over the network by UDP it authorised in the form of datagram

a datagram consist of message which content header information & remaining port content message

Header information consist of source port number & destination port number

java.net package provide classes that are used to send packet data over network.

1) Java.net.DatagramPacket

2) Java.net.DatagramSocket

I] Java.net.Datagrampacket.

This is the class responsible for connection less datagram dilivery service

Constructor:

1] Datagrampacket(Byte[], int len).

It creates Datagram packet object that receive packet of specified length

In specified byte array. It acts as a container for data

Datagram packet class content the Data to be sent or received as well as senders or receivers address & port number.

II] Java.net.DatagramSocket

It will represent connection less socket for sending and receiving datagram packet

An object of this class represent

Socket for sending & receiving datagram packet

Constructor:

1] DatagramSocket()
It creates datagram socket object that can be connected to any available port

Method

1] Void connect(Socket Address)

2] Void disconnect()

Ex:

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
public class UDP sender
{
public static void main (string args[])
{
try
{
DatagramSocket Socket = new
                DatagramSocket();
String message = "Hello, UDP!";
```

```
byte[] buffer = message.getBytes();
InetAddress receiverAddress =
InetAddress.getByName("localhost");
DatagramPacket packet = new
        DatagramPacket(buffer, buffer
                .length,
                receiverAddress, 9876);
Socket.send(packet);
System.out.println("message sent:
                +message);
Socket.close();
}
catch (Exception e)
{
e.printStackTrace();
}
}
```

Q.

```
import java.net.DatagramPacket;
import java.net.DatagramSocket;
public class UDPReceiver
{
Public Static void main (String args[])
{
try
{
DatagramSocket Socket = new DatagramS
                        (9876);
byte[] buffer = new byte [1024];
DatagramPacket packet = new
        DatagramPacket (buffer, buffer leng
System.out.println ("waiting for message");

String receivedmessage = new String
        (packet.getData(); getLength());
System.out.println ("message received"
                    received messag

 Socket.Close();
} catch (Exception e)
{
 e.printStackTrace();
}
}
)
```