Advanced Java Technology

Mr. Prashant Sahatiya, Assistant Professor
Information Technology Engineering

CHAPTER-3

Java Network Programming

Outline

- Network Programming with Java.net package
- Client programs
- Server programs
- Content and protocol handlers
- Chat application example

Network Basics

- Represent interconnection of computing devices either by using cable or wireless devices for resources sharing.
- In network, there may be several computers, some of them receiving the services and some providing services to other.
- The computer which receives service is called a client.
- The computer which provides the service is called server.

Networking Terminology

- **IP Address :** A unique identification number allotted to every device on a network.
- **DNS (Domain Name Service) :** A service on internet that maps the IP addresses with corresponding website names.
- **Port Number :** 2 byte unique identification number for socket.
- **URL (Uniform Resource Locator):** Global address of document and other resources on the world wide web.
- **TCP/IP:** Connection oriented reliable protocol, highly suitable for transporting data reliably on a network.
- **UDP:** Transfers data in a connection less and unreliable manner

Networking Terminology

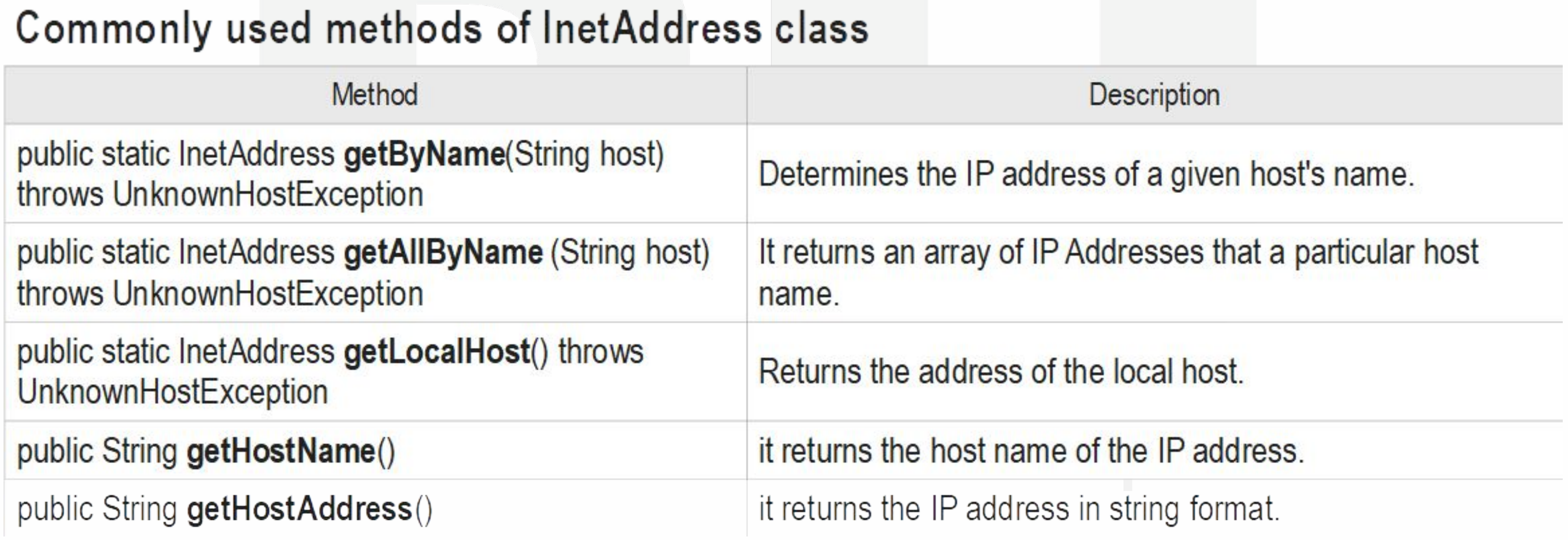
- The term network programming refers to writing programs that execute across multiple devices (computers), in which the devices are all connected to each other using a network.
  - java.net package provides many classes to deal with networking applications in Java

- Here there are few classes related to the connection and then identifying a connection
  - InetAddress
  - URL
  - URLConnection

- For making a actually communication (sending and receiving data) deal with few more classes like ,
  - Socket
  - ServerSocket
  - DatagramPacket
  - DatagramSocket

InetAddress

- InetAddress class belong to the java.net package.
- Using InetAddress class, it is possible to know the IP Address of website / host name
- InetAddress class is used to encapsulate both the numerical IP address and host name for the address.



Commonly used methods of InetAddress class

Method	Description
public static InetAddress **getByName**(String host) throws UnknownHostException	Determines the IP address of a given host's name.
public static InetAddress **getAllByName** (String host) throws UnknownHostException	It returns an array of IP Addresses that a particular host name.
public static InetAddress **getLocalHost**() throws UnknownHostException	Returns the address of the local host.
public String **getHostName**()	it returns the host name of the IP address.
public String **getHostAddress**()	it returns the IP address in string format.

InetAddress.getByName()

- The getByName() method takes host name (server name) and return InetAddress
- Which is nothing but the IP address of the server.

```
import java.net.*; //required for InetAddress Class
public class Address{
        public static void main(String[] args){
        try {
                InetAddress ip = InetAddress.getByName(“www.paruluniversity.ac.in”);
                System.out.println("IP: "+ip);
        }catch(UnknownHostException e) {
                System.out.println(e);
        }
}
}
```

Output:

```
IP:
www.paruluniversity.ac.in/89.238.188.50
```

InetAddress.getAllByName()

- The getAllByName() method returns an array of InetAddresses that represent all of the address that a particular host name.

```
import java.net.*; //required for InetAddress Class
public class Address{
	public static void main(String[] args){
	try {
		InetAddress addressList[ ] = InetAddress.getAllByName(“wixnets.com”);
		for(int i=0;i<addressList.length;i++){
			System.out.println(addressList[i]);
			}
	}catch(UnknownHostException e) {
		System.out.println(e);
	}
  }
}
```

Output:

```
wixnets.com/104.18.48.113
wixnets.com/172.67.198.137
wixnets.com/104.18.49.113
```

InetAddress.getLocalHost()

- The getLocalHost() method takes local host name and return InetAddress
- Which is IP address and name of your current system.

```
import java.net.*; //required for InetAddress Class
public class Address{
        public static void main(String[] args){
        try {
        –    InetAddress localhost=InetAddress.getLocalHost();
        –    System.out.println("LocalHost: "+ localhost);
        }catch(UnknownHostException e) {
                System.out.println(e);
        }
    }
}
```

Output:

```
LocalHost:
LAPTOP-NB4I63VB/10.254.3.79
```

InetAddress.getHostName()

- The getHostName() method takes IP address and return host/ server name in string format.

```
import java.net.*; //required for InetAddress Class
public class Address{
        public static void main(String[] args){
        try {
                InetAddress ip = InetAddress.getByName("10.254.3.79");
                System.out.println("Hostname:"+ip.getHostName());
        }catch(UnknownHostException e) {
                System.out.println(e);
        }
    }
}
```

Output:

```
Hostname: LAPTOP-NB4I63VB
```

InetAddress.getHostAddress()

- The getHostAddress() method takes host name (server name) and return IP address in string format.

```
import java.net.*; //required for InetAddress Class
public class Address{
      public static void main(String[] args){
      try {
            InetAddress ip = InetAddress.getByName(“www.paruluniversity.ac.in”);
            System.out.println(“HostAddress: ”+ip.getHostAddress());
      }catch(UnknownHostException e) {
            System.out.println(e);
      }
   }
}
```

Output:

```
HostAddress: 89.238.188.50
```

Program

- **Write a program to accept a website name and return its IPAddress, after checking it on Internet**

```
import java.net.*; //required for InetAddress Class
public class Address{
        public static void main(String[] args){
        try {
                InetAddress ip =
InetAddress.getByName(“www.paruluniversity.ac.in”);
                System.out.println(“Host Name: ”+ip.getHostName());
                System.out.println(“IP Address:”+ ip.getHostAddress());
        }catch(UnknownHostException e) {
                System.out.println(e);
        }
    }
}
```

Output:

```
Host Name: www.paruluniversity.ac.in
IP Address: 89.238.188.50
```

URL

- Uniform Resource Locator
- URL provides an intelligible form to uniquely identify resources on the internet.
- URLs are universal, every browser uses them to identify resources on the web.
- URL Contains 4 components.
    1. Protocol (http://) //File Name or directory name
    2. Server name or IP address (www.paruluniversity.ac.in)
    3. Port number which is optional (:8090)
    4. Directory resource (index.html)

URL

- URL is represent by class URL in java.net package.
- Use following formats for creating a object of URL class

```
URL obj=new URL(String urlSpecifier) throws MalformedURLException
```

OR

```
URL obj=new URL(String protocol, String host, int port, String path) throws
MalformedURLException
```

OR

```
URL obj=new URL(String protocol, String host, String path) throws
MalformedURLException
```

URL Class Methods

Method	Description
public String **getProtocol()**	it returns the protocol of the URL.
public String **getHost()**	it returns the host name of the URL.
public String **getPort()**	it returns the port number of the URL.
public String **getFile()**	it returns the file name of the URL.
public String **getAuthority()**	it returns the authority part of the URL.
public String **toString()**	it returns the string representation of the URL.
public String **getQuery()**	it returns the query string of the URL.
public String **getDefaultPort()**	it returns the default port of the URL.
public URLConnection **openConnection()**	it returns the instance of URLConnection i.e. associated with this URL.
public URI **toURI()**	it returns a URI of the URL.

Program

- **Write a program to get the Protocol, Host Name, Port Number, and Default File Name from given URL.**

```
import java.net.*; //required for InetAddress Class
public class URLDemo{
        public static void main(String[] args){
        try {
                URL url=
                  new URL(“http://www.paruluniversity.ac.in/PIET”);
                System.out.println(“Protocol: “+url.getProtocol());
                System.out.println(“Host : “+url.getHost());
                System.out.println(“Port : “+url.getPort());
                System.out.println(“File : “+url.getFile());
        }catch(MalformedURLException e) {
                System.out.println(e);
        }
    }
}
```

Output

```
Protocol: http
Host:
www.paruluniversity.ac.in
Port: -1
File: /PIET
```

URLConnection

- URLConnection class is useful to actually connect to a website or resource on a network and get all the details of the website.

- For example, to know the details of www.paruluniversity.ac.in, we should pass its URL to the object of URL class.

- Then using openConnection() method, we should establish a connection with the site on internet.

- openConnection() method returns URLConnection object.

```
URL obj=new URL(String urlSpecifier) throws MalformedURLException
URLConnection conn=obj.openConnection();
```

Parul® University

DIGITAL LEARNING CONTENT

URLConnection Class Methods

Method	Description
public int **getContentLength()**	it returns the size in bytes of the content as a int.
public long **getContentLengthLong()**	it returns the size in bytes of the content as a long.(Added by JDK 7)
public String **getContentType()**	it returns the content-type of the resource.
public long **getDate()**	it returns the time and date of the response in milliseconds.
public long **getExpiration()**	it returns the expiration time and date of the resource.
public String **getHeaderField(int index)**	it returns the value of specific index position.
public String **getHeaderField(String fieldName)**	it returns the value of the header field whose name is specified by field name.
public InputStream **getInputStream()** throws IOException	Returns an input stream that reads from open connection.
public OutputStream **getOutputStream()** throws IOException	Returns an output stream that writes into open connection.

Program

- **Write a program to display the details and page contents of your website.**

```
import java.net.*; //required for InetAddress Class
import java.io.*;
import java.util.*;
public class URLConnectionDemo{
    public static void main(String[] args){
        try {
            URL url=new URL(“https://www.w3schools.com/html/default.asp”)
            URLConnection con = url.openConnection();
              System.out.println(“Date: “ + new Date(con.getDate()));
              System.out.println(“Content-type: “ + con.getContentType());
              System.out.println(“Expiry: “ + con.getExpiration());
              System.out.println(“Length of content: “ + con.getContentLengt
              if(con.getContentLength()>0){
              int ch;
              InputStream in=con.getInputStream();
              while ((ch=in.read())!=-1) {
                  System.out.print((char)ch);
                    }
            }
        }catch(MalformedURLException e) {
            System.out.println(e);
        }
    }
```

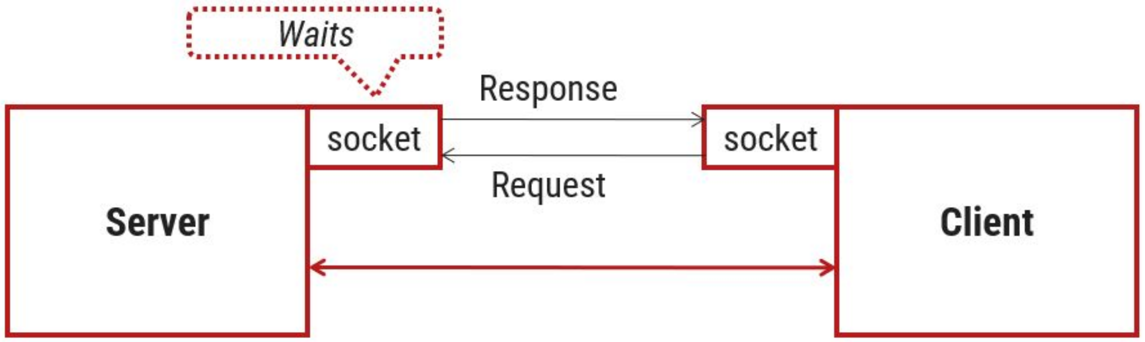
Output

```
Date: Wed Jan 27 10:22:47 IST 2021
Content-type: text/html
Expiry: 1611737567000
Length of content: 62188

<!DOCTYPE html>
<html lang="en-US">
<head>
<title>HTML Tutorial</title>
<meta charset="utf-8">
<meta name="viewport"
content="width=device-width,
initial-scale=1">
...
...
...
<![endif]-->
</body>
</html>
```

Client – Server Architecture

- A Client-Server Architecture consists of two types of components: clients and servers.
- A server component is waiting for requests from client components.
- When a request is received, the server processes the request, and then send a response back to the client.

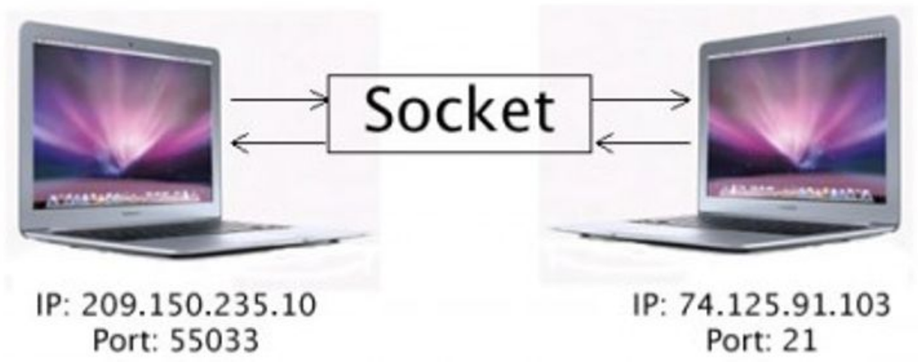


Parul® University

DIGITAL LEARNING CONTENT

Socket Overview

- *"A **socket** is one endpoint of a two-way communication link between two programs running on the network."*
- A **Socket** is combination of an IP address and a port number.
- A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.



- There are two kinds of TCP sockets in java.
- One is for server, and other is for client.
- The Socket class is for clients, and ServerSocket class is for server.

Parul® University

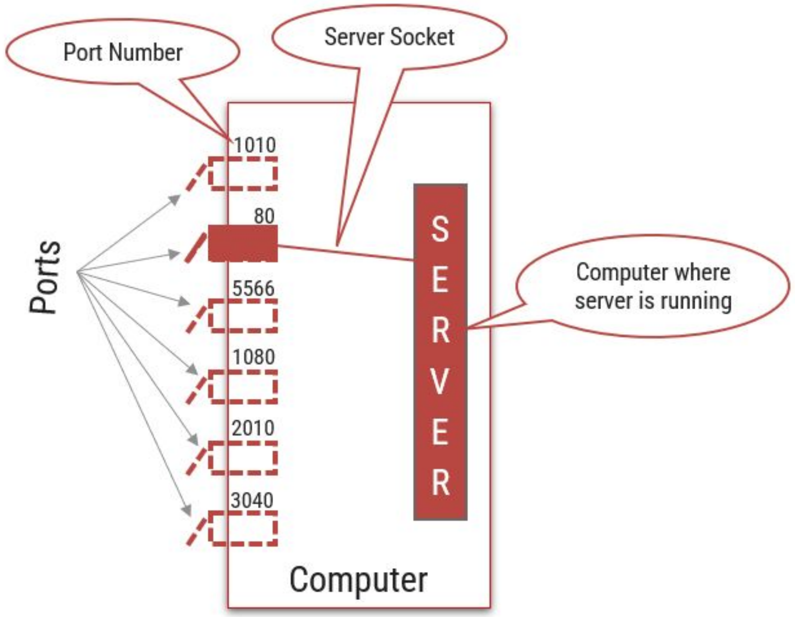
DIGITAL LEARNING CONTENT

Socket Overview

- The server is just like any ordinary program running in a computer.
- Each computer is equipped with some ports.
- The server connects with port.
- This process is called **binding** to a port.
- The connection is called a server socket.

The Java code for creating server in Network Programming:

```
ServerSocket ss = new ServerSocket(80)
```

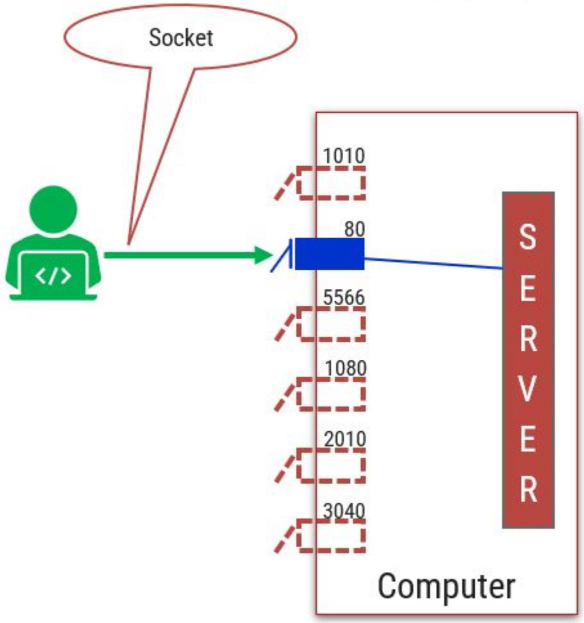


Parul® University

DIGITAL LEARNING CONTENT

Socket Overview

- Server is waiting for client machine to connect.
- Now, client come for communication with server.
- In the next step the client connects to this port of the server's computer.
- The connection is called a (client) socket.
- Now, connection is established between client and server.
- Every time a client is found, its Socket is extracted, and the loop again waits for the next client.



The Java code for creating socket at client side.

```
Socket sock = new Socket("www.darshan.ac.in",80);
```

TCP/IP socket setup at Client & Server side

- At server side, create server socket with some port number using ServerSocket class of java.net package.

```
ServerSocket ss=new  ServerSocket(int port);
```

- Now, we should make the server wait till a client accepts connection, this is done using accept() method.
- This object is used to establish communication with the clients

```
Socket s=ss.accept();
```

- At client side, create socket with host name and port number using Socket class.
- Use following formats for creating a object of Socket class.

```
Socket s=new Socket(String hostName, int port);
```

OR

```
Socket s=new Socket(InetAddress ipAddress, int port);
```

Parul® University

DIGITAL LEARNING CONTENT

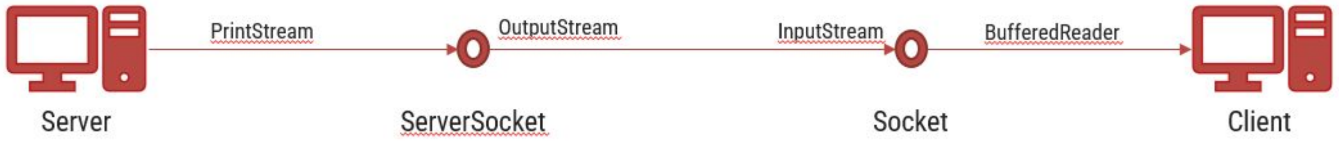
Sockets class method

- Socket defines several instance method.

Method	Description
public InetAddress **getInetAddress()**	Returns the address of the Socket object.
public int **getPort()**	Returns the remote port to which the invoking Socket object is connected
public int **getLocalPort()**	Returns the local port number.
public InputStream **getInputStream()** throws IOException	Returns an input stream that reads(receive) data from this open connection.
public OutputStream **getOutputStream()** throws IOException	Returns an output stream that writes(send) data to open connection.
public void **connect(SocketAddress endpoint, int timeout)**	Connects this socket to the server with a specified timeout value.

(I/O) package in Java

- In Java, streams are the sequence of data that are read from the source and written to the destination.
- The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java.
- There are two type of Streams
  - InPutStream - The InputStream is used to read data from a source.
  - OutPutStream - The OutputStream is used for writing data to a destination.
  - PrintStream – it formats the primitive values as text



Parul® University

DIGITAL LEARNING CONTENT

Creating a Server That Sends Data

- Create a server socket with port number

```
ServerSocket ss=new ServerSocket (8070);
```

- Waiting for establish a connection with client

```
Socket s=ss.accept();
```

- For sending a data attach output stream to the server socket using getOutputStream() method.

```
OutputStream obj=s.getOutputStream();
```

- Create PrintStream object to send data into the socket

```
PrintStream ps=new PrintStream(obj);
```

- Call print() or println() method to send data.

```
ps.println(str);
```

- Close the connection.

```
ss.close(); //close ServerSocket
s.close(); //close Socket
ps.close(); // //close PrintStream
```

Parul® University

DIGITAL LEARNING CONTENT

Creating a Client That Receiving Data

- Create a Socket object with server address and port number

```
Socket s=new  Socket("localhost",8070);
```

- For receiving data attach input stream to the socket, using getInputStream() method

```
InputStream inStr=s.getInputStream();
```

- To read the data from socket, we can take the help of BufferedReader

```
BufferedReader br=new BufferedReader(new InputStreamReader(inStr));
```

- Reade data from BufferedReader object using read() or readLine() method.

```
String receivedMessage = br.readLine();
```

- Close the connection.

```
br.close(); //close BufferReader
s.close(); //close Socket
```

Program

Write a program to create server for the purpose of sending message to the client and also write client side program, which accept all the strings sent by the server.

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

public class MyServer{
public static void main(String[] args){
  try {
            ServerSocket ss = new ServerSocket(888);
            Socket s = ss.accept();
            OutputStream obj = s.getOutputStream();
            PrintStream ps = new PrintStream(obj);
            ps.println("Hello client");
            ss.close(); //close ServerSocket
            s.close(); //close Socket
            ps.close(); //close Printstream

        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

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

public class MyClient {
public static void main(String[] args){
  try {
            Socket s=new Socket("localhost",888);
            InputStream inStr=s.getInputStream();
            BufferedReader br=new BufferedReader(new
InputStreamReader(inStr));
            String receivedMessage = br.readLine();
            System.out.println("Message: "+receivedMessage);
            br.close(); //close BufferReader
            s.close(); //close Socket

        } catch (IOException ex) {
            ex.printStackTrace();
        }
}}
```

Output

```
Message: Hello client
```

Datagrams

- Datagrams are bundles of information passed between machines.
- A datagram is an independent, self-contained message sent over the network whose arrival, arrival time, and content are not guaranteed.
- Once the datagram has been released to its intended target, there is no assurance that it will arrive or even that someone will be there to catch it.
- When the datagram is received, there is no assurance that it hasn't been damaged in transit or that whoever sent it is still there to receive a response
- Java implements datagrams on top of the UDP (User Datagram Protocol) protocol by using two classes:
  - DatagramPacket object is the data container.
  - DatagramSocket is the mechanism used to send or receive the DatagramPackets.

DatagramSocket class method

- DatagramSocket defines several instance method.

Method	Description
public InetAddress **getInetAddress()**	If the socket is connected, then the address is returned.
public int **getPort()**	Returns the number of the port to which the socket is connected.
public int **getLocalPort()**	Returns the local port number.
public boolean **isBound()**	Returns true if the socket is bound to an address.
public boolean **isConnected()**	Returns true if the socket is connected to a server.

DatagramSocket

- DatagramSocket class represents a connection-less socket for sending and receiving datagram packets.
- Use following formats for creating a object of DatagramSocket class

```
DatagramSocket ds=new  DatagramSocket() throws SocketException;
```

- it creates a datagram socket and binds it with the available Port Number on the localhost machine.

```
DatagramSocket ds=new  DatagramSocket(int port) throws SocketException
```

- it creates a datagram socket and binds it with the given Port Number.

```
DatagramSocket ds=new  DatagramSocket(int port,InetAddress ipAddress) throws SocketException
```

- it creates a datagram socket and binds it with the specified port number and host address.

Parul® University

DIGITAL LEARNING CONTENT

DatagramPacket

- Java DatagramPacket is a message that can be sent or received.
- If you send multiple packet, it may arrive in any order.
- Additionally, packet delivery is not guaranteed.
- Use following formats for creating a object of DatagramPacket class

```
DatagramPacket  dp=new  DatagramPacket(byte data[ ],int size)
```

- it specifies a buffer that will receive data and the size of a packet.
- It is used for receiving data over a DatagramSocket

```
DatagramPacket dp=new  DatagramPacket(byte data[ ], int offset,int size)
```

- Allows you to specify an offset into the buffer at which data will be stored.

```
DatagramPacket  dp=new  DatagramPacket(byte data[ ], int size, InetAddress ipAddress, int port)
```

- It's transmits packets beginning at the specifies a target address and port, which are used by a DatagramSocket to determine where the data in the packet will be sent.

Parul® University

DIGITAL LEARNING CONTENT

Sending DatagramPacket by DatagramSocket

- Create a DatagramSocket object.

```
DatagramSocket ds=new DatagramSocket ();
```

- Create a InetAddress object with reciver's ip address

```
InetAddress ip = InetAddress.getByName("Reciver Address");
```

- For sending a data create DatagramPacket object and pass the data within constructure,
- Also specify the size of data, address of receiver with port number

```
DatagramPacket dp=new  DatagramPacket(byte data[ ], int size, InetAddress ipAddress, int port)
```

- Call send() method of DatagramSocket and pass DatagramPacket into method.

```
ds.send(dp);
```

- Close the connection.

```
ds.close(); //close DatagramSocket
```

Receiving DatagramPacket by DatagramSocket

- Create a Datagram Socket object with specific port number.

```
DatagramSocket ds=new DatagramSocket (int port);
```

- Create a byte array for store a receive data, working like a buffer

```
byte[] buffer = new byte[1024];
```

- For receiving a data create Datagram Packet object and pass buffer and buffer size in constructor

```
DatagramPacket dp=new  DatagramPacket(buffer,1024)
```

- Call receive() method of DatagramSocket and pass DatagramPacket into method.

```
ds.receive(dp);
```

- Call getData() method of DatagramPacket for reading data.

```
String str =new String(dp.getData(),0,dp.getLength());
```

- Close the connection.

```
ds.close(); //close DatagramSocket
```

Program

Write a program to create Sender and Receiver for connectionless communication.

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

public class UDPSender {
    public static void main(String[] args) {
        try {
            DatagramSocket ds=new DatagramSocket();
            String str="Message from Sender";
            InetAddress ip=InetAddress.getByName("localhost");
            DatagramPacket dp=new DatagramPacket(str.getBytes(),
str.length(), ip, 6666);
            ds.send(dp);
            ds.close();

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

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

public class UDPReceiver {
    public static void main(String[] args) {
        try {
            DatagramSocket ds = new DatagramSocket(6666);
            byte buffer[] = new byte[1024];
            DatagramPacket dp = new DatagramPacket(buffer, 1024);
            ds.receive(dp);
            String str =new String(dp.getData(),0,dp.getLength());
            System.out.println("Receive: "+str);
            ds.close();

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

Output

```
Message: Message from Sender
```

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in