Sockets

Objectives of This Session

- Describe the concept of Socket
- Describe and demonstrate the connection-oriented communication using Socket & ServerSocket
- Describe the one-way, two-way and multiple client chat server example
- Describe and demonstrate the connectionless communication using DatagramSocket & Datagram Packet

Socket

- Socket is a logical number comprising of the IP address(System identification) and the port number.
- A port number is a logical number assigned to a process. It helps to identify a process on a given system.
- Port numbers upto 1024 are reserved ports or also called as well-known ports.

Port

for example

- 21 – FTP
- 23 – Telnet
- 25 – SMTP
- 80 – HTTP
- 109 - POP

Sockets

Are used to implement

- bi-directional
- reliable
- persistent
- point-to-point
- stream-based connections.

Sockets

- Java makes network programming easier by encapsulating connection functionality in socket classes.
- The related classes are grouped together in java.net package.

Socket class

- A basic class for creating client sockets
- Supports TCP protocol.
- Designed to connect to server sockets & initiate protocol exchanges
- Constructors:
  - Socket(String hostname, int port)
  - Socket(InetAddress ia , int port)

ServerSocket class

- Designed to be a listener which waits on a fixed port for clients to connect
- Once created, it will register itself with the system as having an interest in the client's connection.
- Constructor of this class reflects the port number that accepts connections.

ServerSocket class

- ServerSocket(int port)
- ServerSocket(int port, int maxQueue)
- ServerSocket(int port, int maxQueue, InetAddress localAddress)

Socket accept()

Socket Programming(Writing Server)

- Create Server Socket
- Wait for connection
- Get I/O streams
- Read/Write data
- Close the connection

Socket Programming (Writing Client)

- Create a Socket
- Get I/O streams
- Read/Write data
- Close the connection

java.net classes

- InetAddress
- DatagramPacket
- DatagramSocket
- DatagramSocketImpl
- MulticastSocket

- ServerSocket
- Socket
- SocketImpl
- URL
- URLConnection

URL example

```
public class URLReader   {
    public static void main(String[] args)
   {  URL yahoo = new
              URL("http://www.yahoo.com/");
      Buffered Reader in = new BufferedReader(
                        new InputStreamReader(
                        yahoo.openStream()));
      String s;
      while ((s = in.readLine()) != null)
          System.out.println(s);
      in.close();
    } }
```

InetAddress class

- represents an Internet Protocol (IP) address
- Used to convert between host names & internet addresses.
- Class has no visible constructor.
- To create an InetAddress object use one of the available static methods.

InetAddress class

- byte[ ] getAddress()
- InetAddress getByName(<url>)
- InetAddress[ ] getAllByName()
- InetAddress getLocalHost()

Example Chat Server

```
public void runServer( )
{  ServerSocket server;
   Socket connection;
    try{
        server = new
          ServerSocket(2000,1,InetAddress.getLocalHost());
        connection = server.accept();
         DataInputStream sin = new
DataInputStream(connection.getInputStream());
         DataOutputStream sout = new
DataOutputStream(connection.getOutputStream());
          DataInputStream myin = new
                              DataInputStream(System.in);
}
```

Socket Programming

```
while(true)
{
    System.out.println(sin.readUTF());
    sout.writeUTF(myin.readLine());

}
```

Socket Programming

```
public void runClient( )
{
    Socket client;
    client = new Socket(InetAddress.getLocalHost(),2000 );
    DataInputStream sin = new
        DataInputStream(client.getInputStream());
    DataOutputStream sout = new
DataOutputStream(client.getOutputStream());
    DataInputStream myin = new
DataInputStream(System.in);

}
```

DatagramSocket

- This class represents a socket for sending and receiving datagram packets
- A datagram socket is the sending or receiving point for a packet delivery service
- Each packet sent or received on a datagram socket is individually addressed and routed.

Methods of DatagramSocket class

- **void send(DatagramPacket dp)**
  - Sends the packet to the remote address specified in the dp packet
- **void receive(Datagrampacket dp)**
  - Blocking call- waits until it reads a packet from this DatagramSocket.
- **void close()**
  - Closes this socket & releases any resource allocated to the socket object

DatagramPacket

- This class represents a datagram packet
- Datagram packets are used to implement a connectionless packet delivery service
- The packet class contains connection information as well as the data.

Socket Programming

- **DatagramPacket(byte data[ ], int size)**
  - Specifies a buffer that will receive data & the size of a packet
- **DatagramPacket(byte data[ ], int size,InetAddress ipAddress, int port)**
  - Specifies a target address & port which are used by datagram Socket to determine where the data in the packet will be sent.

Methods of DatagramPacket

InetAddress getAddress()

Returns destination Inetaddress,typically used for sending.

int getPort()

Returns the port number

byte[] getData()

Returns data contained in datagram(after receiving)

int getLength()

Returns length of valid data contained in the byte array that's returned from getData() method.

Socket Programming

```
DatagramSocket ds = new DatagramSocket (2000);
byte data[] = new byte[100];
receivePacket = new DatagramPacket(data, data.length);

ds.receive(receivePacket);
ds.send(receivePacket);
receivePacket.getAddress();
receivePacket.getPort();
receivePacket.getLength();
receivePacket.getData();
```