# Java Networking

Ajay Khatri

# Java Networking

- Connecting two or more computing devices together so that we can share resources.

# Advantage of Java Networking

- Sharing resources
- Centralize software management

# Java Networking Terminology

- IP Address
- Protocol
- Port Number
- MAC Address
- Connection-oriented and connection-less protocol
- Socket

# IP Address

- IP address is a unique number assigned to a node of a network e.g. 192.168.0.1 . It is composed of octets that range from 0 to 255.
- It is a logical address that can be changed.

# Protocol

- A protocol is a set of rules basically that is followed for communication. For example:
  - TCP
  - FTP
  - Telnet
  - SMTP
  - POP etc.

# Port Number

- Uniquely identify different applications.
- Acts as a communication endpoint between applications.
- Associated with the IP address for communication between two applications.

# MAC (Media Access Control)

- MAC Address is a unique identifier of NIC (Network Interface Controller).
- A network node can have multiple NIC but each with unique MAC.

# Connection-oriented and connection-less protocol

- In connection-oriented protocol, acknowledgement is sent by the receiver. So it is reliable but slow. The example of connection-oriented protocol is TCP.
- In connection-less protocol, acknowledgement is not sent by the receiver. So it is not reliable but fast. The example of connection-less protocol is UDP.

# IP Address

- A socket is an endpoint between two way communication.

# Java Socket Programming

Ajay Khatri

# What is Socket Programming ?

- Java Socket programming is used for communication between the applications running on different JRE.
- Java Socket programming can be connection-oriented or connection-less.
- **Socket and ServerSocket** classes are used for connection-oriented socket programming and **DatagramSocket and DatagramPacket** classes are used for connection-less socket programming.

# Client must know two information

- IP Address of Server, and
- Port number.

# *MyServer.java*

```
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection
DataInputStream dis=new DataInputStream(s.getInputStream());
String  str=(String)dis.readUTF();
System.out.println("message= "+str);
ss.close();
```

# *MyClient.java*

- Socket s=**new** Socket("localhost",6666);
- DataOutputStream dout=**new** DataOutputStream(s.getOutputStrea m());
- dout.writeUTF("Hello Server");
- dout.flush();
- dout.close();
- s.close();

# Read-Write both side(Server)

```
ServerSocket ss=new ServerSocket(3333);
Socket s=ss.accept();
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String str="",str2="";
while(!str.equals("stop")){
str=din.readUTF();
System.out.println("client says: "+str);
str2=br.readLine();
dout.writeUTF(str2);
dout.flush();
}
din.close();
s.close();
ss.close();
```

Offline Java Tutorial App :- https://play.google.com/store/apps/details?id=in.ajaykhatri.javatutorial

# Read-Write both side (Client.java)

```
Socket s=new Socket("localhost",3333);
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(
new InputStreamReader(System.in));

String str="",str2="";
```

```
while(!str.equals("stop")){
str=br.readLine();
dout.writeUTF(str);
dout.flush();
str2=din.readUTF();
System.out.println("Server says: "+str2);
}
dout.close();
s.close();
```

# References

https://www.javatpoint.com

https://play.google.com/store/apps/details?id=in.ajaykhatri.javatutorial

Offline Java Tutorial App :- https://play.google.com/store/apps/details?id=in.ajaykhatri.javatutorial