

Network Programming with Java

Unit - 5

Topics to be covered

- ❏ Network Programming

- ❏ java.net package

 - ❏ InetAddress Class

 - ❏ URL Class

 - ❏ URLConnection Class

- ❏ Establishing 2-way communication (Client/Server)

 - ❏ TCP/IP Client Sockets

 - ❏ TCP/IP Server Sockets

Network Programming

- ❏ To develop an application that can communicate/pass data over the network using various network protocols is known as Network Programming
- ❏ Various protocols include TCP, UDP, FTP etc.
- ❏ Advantages of Network Programming:
 - ❏ Sharing resources
 - ❏ Centralize Software Management
- ❏ `java.net` is the package to perform network programming

Java Networking Terminologies

📄 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.

Java Networking Terminologies

📌 Port Number

- 📌 The port number is used to uniquely identify different applications. It acts as a communication endpoint between applications.
- 📌 The port number is associated with the IP address for communication between two applications.

📌 MAC Address

- 📌 MAC (Media Access Control) address is a unique identifier of NIC (Network Interface Controller). A network node can have multiple NIC but each with unique MAC address.
- 📌 For example, an ethernet card may have a MAC address of 00:0d:83::b1:c0:8e.

Java Networking Terminologies

☞ 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.

☞ But, 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.

☞ Socket

☞ A socket is an endpoint between two way communications.

☞ Visit next page for Java socket programming.

java.net package

- ❏ Java Networking is a concept of connecting two or more computing devices together so that we can share resources.
- ❏ Java socket programming provides facility to share data between different computing devices.
- ❏ The java.net package supports two protocols,
 - ❏ **TCP:** The Transmission Control Protocol ensures that communication between sender and receiver is reliable. TCP is used in conjunction with the Internet Protocol, which is referred to as TCP/IP.
 - ❏ **UDP:** By allowing data packets to be transferred between two or more nodes, User Datagram Protocol provides a connection-less protocol service.

java.net Package

☞ The `java.net` package can be divided into two sections:

☞ **A Low-Level API:** It deals with the abstractions of addresses i.e. networking identifiers, Sockets i.e. bidirectional data communication mechanism and Interfaces i.e. network interfaces.

☞ **A High Level API:** It deals with the abstraction of URIs i.e. Universal Resource Identifier, URLs i.e. Universal Resource Locator, and Connections i.e. connections to the resource pointed by URLs.

☞ This package provides many classes to perform network operations

Classes from java.net package

Authenticator	Inet4Address	Socket
CacheRequest	Inet6Address	SocketAddress
CacheResponse	IDN	SocketImpl
ContentHandler	URLConnection	SocketPermission
CookieHandler	HttpCookie	StandardSocketOptions
CookieManager	NetPermission	URI
DatagramPacket	NetworkInterface	URL
DatagramSocket	PasswordAuthentication	URLClassLoader
DatagramSocketImpl	Proxy	URLConnection
InterfaceAddress	ProxySelector	URLDecoder
JarURLConnection	ResponseCache	URLEncoder
MulticastSocket	SecureCacheResponse	URLStreamHandler
InetSocketAddress	ServerSocket	
InetAddress		

InetAddress Class

- ❏ Java InetAddress class represents an IP address. The `java.net.InetAddress` class provides methods to get the IP of any host name for example `www.google.com`, `www.facebook.com`, etc.
- ❏ An IP address helps to identify a specific resource on the network using a numerical representation.
- ❏ Most networks combine IP with TCP (Transmission Control Protocol). It builds a virtual bridge among the destination and the source.
- ❏ 2 types of IP: IPv4 & IPv6

InetAddress Class

Methods :

Method	Description
<code>public static InetAddress getByName(String host) throws UnknownHostException</code>	It returns the instance of InetAddress containing LocalHost IP and name.
<code>public static InetAddress getLocalHost() throws UnknownHostException</code>	It returns the instance of InetAddress containing local host name and address.
<code>public String getHostName()</code>	It returns the host name of the IP address.
<code>public String getHostAddress()</code>	It returns the IP address in string format.

InetAddress Example:

```
import java.io.*;
import java.net.*;
public class InetDemo{
    public static void main(String[] args){
        try{
            InetAddress ip =
InetAddress.getByName("www.google.com");
            System.out.println("Host Name: " + ip.getHostName());
            System.out.println("IP Address: " +
ip.getHostAddress());
        }catch(Exception e){System.out.println(e);}
    }
}
```

Output:

Host Name: www.google.com

IP Address: 74.125.126.106

URL Class

❏ Used for Universal Resource Locator

❏ Pointer to locate a resources like, a Text file, an Image, a Directory, a Binary file, an HTML Document etc. in www (World Wide Web).

❏ A typical URL is in the form of:

`http://www.example.com:80/home.html`

❏ **Protocol:** In this case the protocol is HTTP, It can be HTTPS in some cases

❏ **Hostname:** Hostname represent the address of the machine on which resource is located, in this case, `www.example.com`

❏ **Port Number:** It is an optional attribute. If not specified then it returns -1. In the above case, the port number is 80.

❏ **Resource name:** It is the name of a resource located on the

URL Class

📄 `java.net` package has a `URL` class, a subclass of `java.lang.Object`.

📄 Constructors:

Constructor	Explanation
<code>public URL(String url)</code>	This constructor creates an object of <code>URL</code> class from given string representation
<code>public URL(String protocol, String host, int port, String file)</code>	This constructor creates an object of <code>URL</code> from the specified protocol, host, port number, and file.
<code>public URL(String protocol, String host, String file)</code>	This constructor creates an object of <code>URL</code> from the specified protocol, port number, and file. The default port number is used in this case.
<code>public URL(URL context, String src)</code>	This constructor creates an instance of a <code>URL</code> by parsing the given <code>src</code> with the specified

URL Class

Method	Description
<code>public String getProtocol()</code>	it returns the protocol of the URL.
<code>public String getHost()</code>	it returns the host name of the URL.
<code>public String getPort()</code>	it returns the Port Number of the URL.
<code>public String getFile()</code>	it returns the file name of the URL.
<code>public String getAuthority()</code>	it returns the authority of the URL.
<code>public String toString()</code>	it returns the string representation of the URL.
<code>public String getQuery()</code>	it returns the query string of the URL.

URL Class

Method	Description
<code>public String getDefaultPort()</code>	it returns the default port of the URL.
<code>public URLConnection openConnection()</code>	it returns the instance of URLConnection i.e. associated with this URL.
<code>public boolean equals(Object obj)</code>	it compares the URL with the given object.
<code>public Object getContent()</code>	it returns the content of the URL.
<code>public String getRef()</code>	it returns the anchor or reference of the URL.
<code>public URI toURI()</code>	it returns a URI of the URL.

URL Example:

```
import java.net.*;
public class URLEDemo{
    public static void main(String[] args){
        try{
            URL url=new URL("https://www.programiz.com/java-
programming/online-compiler/");

            System.out.println("Protocol: "+url.getProtocol());
            System.out.println("Host Name: "+url.getHost());
            System.out.println("Port Number: "+url.getPort());
            System.out.println("File Name: "+url.getFile());
        }catch(Exception e){System.out.println(e);}
    }
}
```

Output:

Protocol: https

Host Name: www.programiz.com

Port Number: -1

File Name: /java-

programming/online-compiler/

URL v/s URI

URL	URI
URL is used to describe the identity of an item.	URI provides a technique for defining the identity of an item.
URL links a web page, a component of a web page or a program on a web page with the help of accessing methods like protocols.	URI is used to distinguish one resource from other regardless of the method used.
URL provides the details about what type of protocol is to be used.	URI doesn't contains the protocol specification.
URL is a type of URI.	URI is the superset of URL.
It comprises of protocol, domain, path, hash, and so on.	It comprises of scheme, authority, path, query and many more.
Ex- https://www.google.com/	Ex- <code>urn:isbn:0-294-56559-3</code>

URLConnection Class

- ❏ The Java `URLConnection` class acts as a bridge between the URL and the application. It can be used to read and write data to the URL-referenced resource.
- ❏ `URLConnection` is an **abstract** class. The **two subclasses** **`HttpURLConnection`** and **`JarURLConnection`** makes the connection between the client Java program and URL resource on the internet.
- ❏ With the help of `URLConnection` class, a user can **read and write** to and from any resource referenced by an URL object.
- ❏ Once a connection is established and the Java program has an `URLConnection` object, we can use it to read or write or get further information like content length, etc.
- ❏ Constructor:

```
protected URLConnection(URL url)
```

URLConnection Methods

Method	Description
<code>void addRequestProperty(String key, String value)</code>	It adds a general request property specified by a key-value pair
<code>void connect()</code>	It opens a communications link to the resource referenced by this URL, if such a connection has not already been established.
<code>boolean getAllowUserInteraction()</code>	It returns the value of the allowUserInteraction field for the object.
<code>int getConnectionTimeout()</code>	It returns setting for connect timeout.
<code>Object getContent()</code>	It retrieves the contents of the URL connection.
<code>Object getContent(Class[] classes)</code>	It retrieves the contents of the URL connection.
<code>String getContentEncoding()</code>	It returns the value of the content-encoding header field.
<code>int getContentLength()</code>	It returns the value of the content-length header field.

URLConnection Methods

Method	Description
<code>long getDate()</code>	It returns the value of the date header field.
<code>static boolean getDefaultAllowUserInteraction()</code>	It returns the default value of the allowUserInteraction field.
<code>boolean getDefaultUseCaches()</code>	It returns the default value of an URLConnetion's useCaches flag.
<code>boolean getDoInput()</code>	It returns the value of the URLConnection's doInput flag.
<code>boolean getDoOutput()</code>	It returns the value of the URLConnection's doOutput flag.
<code>long getExpiration()</code>	It returns the value of the expires header files.
<code>static FileNameMap getFilenameMap()</code>	It loads the filename map from a data file.
<code>String getHeaderField(int n)</code>	It returns the value of n th header field
<code>String getHeaderField(String name)</code>	It returns the value of the named header field.

URLConnection Methods

Method	Description
<code>int getHeaderFieldInt(String name, int Default)</code>	It returns the value of the named field parsed as a number.
<code>String getHeaderFieldKey(int n)</code>	It returns the key for the n th header field.
<code>long getHeaderFieldLong(String name, long Default)</code>	It returns the value of the named field parsed as a number.
<code>Map<String, List<String>> getHeaderFields()</code>	It returns the unmodifiable Map of the header field.
<code>long getIfModifiedSince()</code>	It returns the value of the object's ifModifiedSince field.
<code>InputStream getInputStream()</code>	It returns an input stream that reads from the open condition.
<code>long getLastModified()</code>	It returns the value of the last-modified header field.
<code>OutputStream getOutputStream()</code>	It returns an output stream that writes to the connection.
<code>Permission getPermission()</code>	It returns a permission object representing

URLConnection Methods

Method	Description
<code>Map<String, List<String>> getRequestProperties()</code>	It returns the value of the named general request property for the connection.
<code>URL getURL()</code>	It returns the value of the URLConnection's URL field.
<code>boolean getUseCaches()</code>	It returns the value of the URLConnection's useCaches field.
<code>Static String guessContentTypeFromName(String fname)</code>	It tries to determine the content type of an object, based on the specified file component of a URL.
<code>static String guessContentTypeFromStream(InputStre am is)</code>	It tries to determine the type of an input stream based on the characters at the beginning of the input stream.
<code>void setAllowUserInteraction(boolean allowuserinteraction)</code>	It sets the value of the allowUserInteraction field of this URLConnection.
<code>static void setContentHandlerFactory(ContentHand lerFactory fac)</code>	It sets the ContentHandlerFactory of an application.

URLConnection Methods

Method	Description
<code>static void setDefaultAllowUserInteraction(boolean defaultallowuserinteraction)</code>	It sets the default value of the allowUserInteraction field for all future URLConnection objects to the specified value.
<code>void setDefaultUseCaches(boolean defaultusecaches)</code>	It sets the default value of the useCaches field to the specified value.
<code>void setDoInput(boolean doinput)</code>	It sets the value of the doInput field for this URLConnection to the specified value.
<code>void setDoOutput(boolean dooutput)</code>	It sets the value of the doOutput field for the URLConnection to the specified value.

💡 How to get URLConnection object???

```
public URLConnection openConnection() throws IOException {}
```


URLConnection Example

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

public class Test {
    public static void main(String[] args) {
        try {

            String url = "https://google.com";
            String filePath = "Google.html";

            URL urlObj = new URL(url);
            URLConnection urlCon = urlObj.openConnection();
```

URLConnection Example

```
InputStream inputStream = urlCon.getInputStream();
BufferedInputStream reader = new BufferedInputStream(inputStream);
BufferedOutputStream writer = new BufferedOutputStream(new
FileOutputStream(filePath)); byte[] buffer = new byte[4096];
int bytesRead = -1;

while ((bytesRead = reader.read(buffer)) != -1) {
    writer.write(buffer, 0, bytesRead);
}
writer.close(); reader.close();
} catch(IOException ioe) {System.out.println(ioe.getMessage());}
}
}
```

Socket Programming

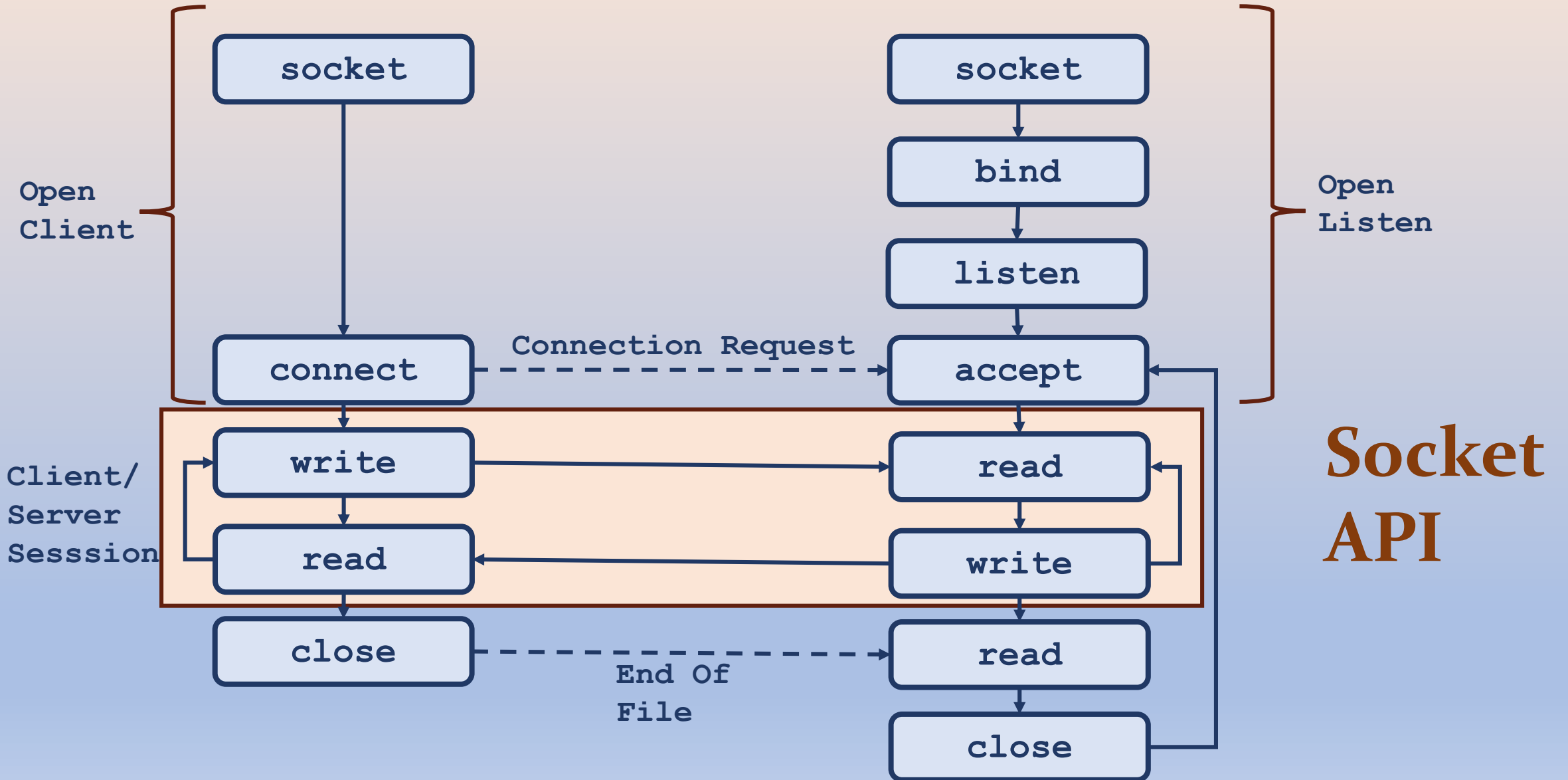
- ❏ Java Socket programming allows **apps running on different JREs** to communicate with one another.
- ❏ **Connection-oriented** or **connection-less** Java Socket programming is possible.
- ❏ Connection-oriented socket programming is done using the **ServerSocket** classes, whereas connection-less socket programming is done with the **DatagramSocket** and **DatagramPacket** classes.
- ❏ The client in socket programming must know two information:
 - ❏ **IP Address of Server** and
 - ❏ **Port number.**

Socket Programming

- ❏ First one-way client-server communication.
- ❏ In this, the client delivers a message to the server, which reads it and prints it.
- ❏ Two classes are utilized in this case: **Socket** and **ServerSocket**.
 - ❏ Socket class:
 - ❏ Used to communicate between the client and the server.
 - ❏ We can read and write messages in this class.
 - ❏ ServerSocket class:
 - ❏ Used on the server.
 - ❏ The `accept()` method of the `ServerSocket` class prevents the console from being used until the client is connected.
 - ❏ After a successful client connection, it returns a `Socket` instance to the server.

Client

Server



Socket Class

❏ A socket is simply an endpoint for communications between the machines.

❏ The Socket class can be used to create a connection through socket.

❏ Method	Description
public InputStream getInputStream()	returns the InputStream attached with this socket.
public OutputStream getOutputStream()	returns the OutputStream attached with this socket.
public synchronized void close()	closes this socket

ServerSocket Class

- ❏ The ServerSocket class can be used to create a server socket.
- ❏ This object is used to establish communication with the clients.
- ❏ Methods:

Method	Description
public Socket accept()	returns the socket and establish a connection between server and client.
public synchronized void close()	closes the server socket.

Communicating through Java Sockets (Client / Server)

First 2 different java files are needed (1 - for Client [Client.java], 1 - for Server [Server.java])

Creating a server socket (in MyServer.java):

```
ServerSocket ss = new ServerSocket(6666); // Port: 6666  
Socket s = ss.accept(); //establish connection and wait for client
```

Creating a client socket (in MyClient.java):

```
Socket s = new Socket("localhost",6666);
```

Then using this connection, use various methods to communicate data in between client-server and process that data as per the application.

Here a one way communication is demonstrated first like **echo server**.

MyClient.java (1 – way Communication)

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

public class MyClient {
    public static void main(String[] args) {
        try{
            Socket s = new Socket("localhost", 6666);
            DataOutputStream out =new DataOutputStream(s.getOutputStream());
            out.writeUTF("Hello Server");
            out.flush();
            out.close();
            s.close();
        }catch(Exception e){System.out.println(e);}
    }
}
```

MyServer.java (1 – way Communication)

```
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(); //establishes connection
            DataInputStream in = new DataInputStream(s.getInputStream());
            String data = (String) in.readUTF();
            System.out.println("Message= " + data);
            ss.close();
        }catch(Exception e){System.out.println(e);}
    }
}
```

2 - way communication through Java Socket (Server.java)

```
import java.io.*;
import java.net.*;
public class server {
    public static void main(String[] args){
        try{
            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();  }catch(Exception
e){System.out.println(e);}  }
```

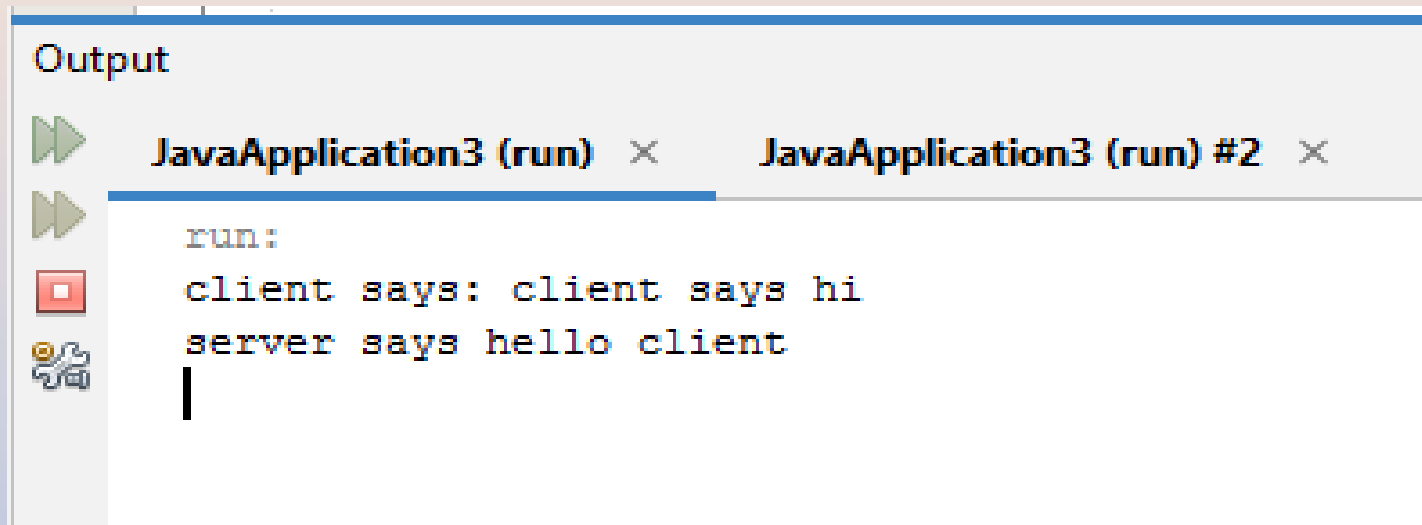
2 - way communication through Java Socket (Client.java)

```
import java.io.*;
import java.net.*;
public class client {
    public static void main(String[] args) {
        try{
            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));
```

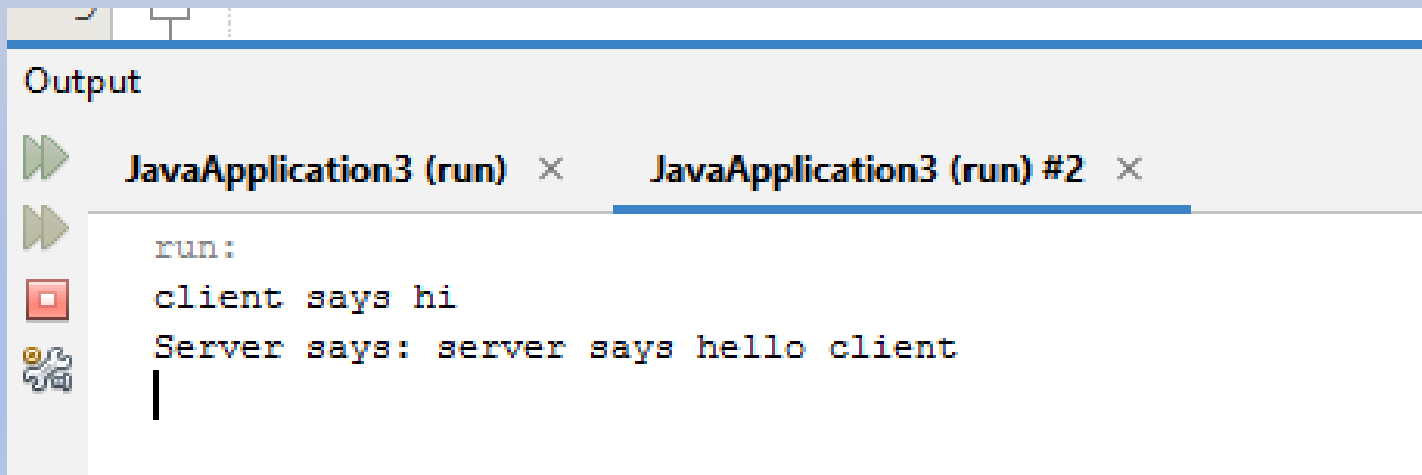
```
String str="",str2="";
while(!str.equals("stop")){
    str=br.readLine();
    dout.writeUTF(str);
    dout.flush();
    str2=din.readUTF();
    System.out.println("Server says: "+str2);
}

    dout.close();
    s.close();
}catch(Exception e){System.out.println(e);}
}
```

Output:



```
Output
JavaApplication3 (run) × JavaApplication3 (run) #2 ×
run:
client says: client says hi
server says hello client
|
```



```
Output
JavaApplication3 (run) × JavaApplication3 (run) #2 ×
run:
client says hi
Server says: server says hello client
|
```