Assignment no. 05

Page No.:
Date: 19/12/22

Q.1) What is the role of networking in Java?

→ 1) Networking is the concept of connecting multiple remote or local devices together. Java networking classes and interface use java.net package. These classes and interface provide the functionality to develop system independent network communication.

2) The java.net package provides the functionality for two common protocol.

a) TCP/IP :- TCP/IP stands for transmission control protocol that provides reliable communication between two devices also betn two application so that they can easily communicate. It is connection based protocol.

b) UDP :- UDP stands for user datagram protocol. It sends packet of data called datagram from one computer to another with no guarantee of arrival. It is connectionless based protocol.

3) Networking Terminology :-

a) Request / Response :- When an input data is sent to an application via network, it is called Request. The output data coming out from application back to client program is called Response.

b) protocol :- A protocol is basically a set of rules and guidelines which provides instruction to send request and receive

response over network.
For ex. TCP, UDP, FTP, SMTP, etc.

c) IP address:- It stands for Internet Protocol address. It is an indentification numbers that is assigned to node of a computer in the network.
For ex:- 192.168.2.01.
Range:- 0.0.0.0 to 255.255.255.255

d) Port Number:- The port number is identification number of server software. The port number is unique for different application. It is 32bit positive integer number having between Ranges 0 to 35535.

e) Socket:- Socket is listener through which computer can receive and response. It is the end point of two way communication.

f) URL:- URL stands for Uniform Resource locator. It is the string of text that identifies all the resource on internet telling address of the resource how to communicates with it and retrieve something from it.

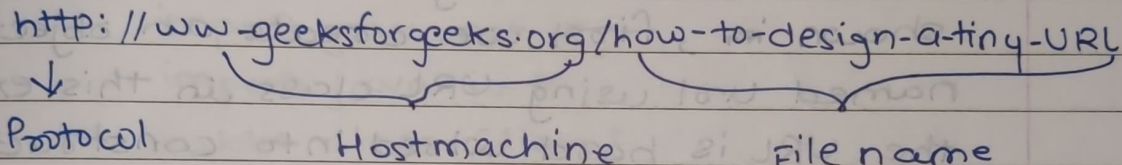
Q.2) How to manipulate URL in java. How to read file on web server.

→ The URL class is the gateway of the resource available on the internet.
A class URL represents a Uniform Resource Locator or it can ~~refer to a more~~ point to a simple file or directory or it can refer to more

complicated object such as a query to a database or to a search engine.

As many of you must be knowing uniform resource locator, URL is a string of text that identifies all resources on internet, telling us the address of resources, how to communicate with it and retrieve something from it.

A simple URL looks like:



- Components of URL:-

a) protocol :- HTTP is the protocol here.
b) Hostname:- Name of machine on which resource lives.
c) File Name:- The path name to the file on the machine.
d) Port number:- Port number to which to connect.

- Constructors of URL class:-

1) URL(String address) throws MalformedURLException:
It creates a URL object from specified string
2) URL(String protocol, String host, String file):-
Creates a URL object from specified protocol, host, port and filename.
3) URL(String protocol, String host, int port, String file):-
Creates a URL object from specified protocol, host port and filename.

- Reading a file on web server :-

1) Given the proper permission, you can easily read files directly from filesystem of an HTTP server. ~~In this~~

2) I have created method name getTODOlist() to perform applet's read function. This method is called once, when the applet is first started, and again each time the user clicks on the refresh Button.

3) The code for the goTodolist() method shown ~~in~~, first we create an object named url using URL class in this ex. the URL is hard wired into constructor method for the URL class.

4) Next we create URL connection object named urlConnection by invoking url.openConnection(). Then, this connection is defined to be on input connection by invoking setDoInput() method of URLConnection class. To make sure that we get a real copy of data file and not a cached file, we also invoke setUsesCaches(false).

Q.3) How to establish simple client using stream socket?

→ To connect to other machine we need a socket connection. A socket connection means two machines have information about each other's network location (IP Address) and TCP port. The java.net.Socket class represents a Socket. To open a socket:

Socket socket = new Socket("127.0.0.1", 5000)

- First argument :- IP address of server.
- Second argument :- TCP Port.
- Communication :- To communicate over a socket connection, streams are used to both input and output the data.
- Closing the connection :- The Socket Connection is closed explicitly once the message to server is sent.

```
import java.net.*;
import java.io.*;
public class Client {
private Socket socket = null;
private DataInputStream input = null;
private DataOutputStream out = null;
public client (String address, int port) {
   try {
   socket = new Socket (address, port);
System.out.println("Connected");
input = new DataOutputStream(socket.getOutput-
                                    Stream());
input = new DataInputStream (System.in);
out = new DataOutputStream(socket.getOutput-
                                    Stream());
   }
catch (UnknownHostException u) {
System.out.println(u);
}
catch (IOException i) {
```

```
System.out.println(i);
  }
String line = "";
while(!line.equals("Over")){
  try{
    line = input.readLine();
    out.writeUTF(line);
    }
    catch(IOException i){
     System.out.println(i);
      }
    }
    try{
     input.close();
     out.close();
     socket.close();
  }
  catch(IOException i){
  System.out.println(i);
     }
   }
public static void main(String args[]){
Client client=new Client("127.0.0.1",5000);
   }
}
```

Q.4) How to establish a simple server using stream socket?

→ To write a server application two sockets are needed.

- A ServerSocket which waits for the client requests.

- A plain old socket to use for communication with client.
- Communication :- getOutputStream() method is used to send the output through socket.
- Close the connection :- After finishing, it is important to close the connection by closing socket as well as input/output streams.

```
import java.net.*;
import java.io.*;
public class Server {
private Socket socket = null;
private ServerSocket server = null;
private DataInputStream in = null;
public Server (int port) {
try {
server = new ServerSocket(port);
System.out.println("Server Started");
System.out.println("Waiting for Client...");
socket = server.accept();
System.out.println("Client accepted");
in = new DataInputStream(
new BufferedInputStream(socket.getInputStream()));
String line = "";
while (!line.equals("Over")) {
try {
line = in.readUTF();
System.out.println(line);
}
catch (IOException i) {
System.out.println(i);
}
}
```

```
System.out.println("Closing Connection");
    socket.close();
    in.close();
    }
catch(IOException i){
    System.out.println(i);
    }
}
public static void main(String args[]){
Server server = new Server(5000);
    }
}
```

Q.5) ~~Explain Client Server interaction with example.~~

→ ~~e simple stream socket connection.~~

Q.7) How connectionless client server interaction handles with use of datagrams?

→ • Connection oriented is like the telephone system in which you dial and are given a connection to the telephone of the person with whom you wish to communicate. The connection is maintained ~~as~~ for the duration of your phone call, even when you are ~~not getting~~ not talking.

• Connectionless transmission with datagrams is more likely the way mail is carried via postal service. If a large message will not fit in one envelope, you break it into seperate message pieces that you place in seperate, sequentially numbered envelopes.

Each of the letter is then mailed at the same time.

- The letters could arrive in order, out of orders or not at all.
- The person at receiving end reassembles the message pieces into sequential order before attempting to make the sense of message.
- If your message is small enough to fit in one envelope, you need not worry about 'out-of-sequence' problem, but it is still possible that message might not arrive.
- We use datagrams to send packets of info. via the user Datagram Protocol (UDP) between client application and server application. In Client application, user types message and presses Enter. The program converts message into a byte array and places it in a datagram packet that is sent to server. the server receives packet and displays information in it, then echoes the packet back to client. Upon receiving packet, the client displays information it contains.

Q.8) Explain the following :-

a) Networking :-

Networking is the concept of connecting multiple remote or local devices together. Java programs communicates over the network at application layers.

All the java networking classes

and interface provides the functionality to develop system-independent network communication.

b) Stream Socket:-

To communicate over a socket connection, streams are used to both input and output the data. The socket connection is closed explicitly once the message to server is send.

c) Datagram:-

Datagram is independent self contained message sent over the network whose arrival time and content are not guaranteed.

DatagramPacket and Datagram Socket class in the system-independent datagram communication using UDP.

Q.) Explain Client-server interaction with simple stream socket connection.

→ a) Simple Server:-

To write a server application two socket are needed.

- A server socket which waits for client requests.
- A plain old socket to use for a communication with client.
- Communication:-

getOutputStream() method is used to

send output through socket.

- close connection:-
  After finishing, it is important to close the connection. By closing socket as well as input/output stream.

b) Simple Client:-
To connect to other machine we need ~~to~~ socket connection. A socket connection means two machines have information about each other network location (IP address) and TCP Port.

Socket socket = new Socket("127.0.0.1", 5000)

- First argument:- IP address.
- Second argument:- TCP port
- Communication:- To communicate over a socket connection, streams are used to both input/output data.
- Closing connection:- The socket connection is closed explicitly once the message is sent.

Q.10) Write down a Java program using UDP Based client - server architecture.

→ *Server side:-

```
import java.net.*;
import java.util.Date;
class server{
public static void main(String args[]) throws
                                    Exception{
```

```
DatagramSocket b= new DatagramSocket();
InetAddress address = InetAddress.getLocalHost
                                    ();

System.out.println("Your time check is ready");
System.out.println("you can check your time");
 while (true){
    Thread.sleep (2000);
Date currentDate = new Date();
String s1 = currentDate.toString();
byte arr[] = s1.getBytes();
DatagramPacket dpack = new DatagramPacket
        (arr, arr.length, address, 2000);
   b.send(dpack);
   }
  }
```

* Client side:-

```
   import java.io.IOException;
   import java.net.*;
  class client{
public static void main(String args[]) throws
                                    Exception{
 DatagramSocket b = new DatagramSocket(2000);
 Datagram a;
 for (int i=0; i<3; i++){
 byte arr1[] = new bytes[100];
 a = new DatagramPacket(arr1, arr1.length);
 b.receive(a);
 byte arr2[]= a.getdata();
 String str = new String(arr2);
 System.out.println(str);
  }
 }
```

• Output:-

```
Your timechecker is ready
check your time
```

```
Sun, 25 Dec
:
:
:
```