UNIT: 5: Network programming

5.1: Networking Basics:
TCP, UDP, Ports, IP Address network classes in JDK

5.2: Working with URLs:
connecting to URLs, Reading directly from URL
InetAddress class

5.3: Sockets: TCP sockets, UDP sockets, Serving multiple clients, Halt close, Interruptible sockets, sending Email.

2070/12: what is socket? How can you communicate two programs in a network using TCP socket?

* A network socket is one endpoint in a communicati on flow between two programs running over a network. sockets can also be used for communi. cation between processes within the same computer

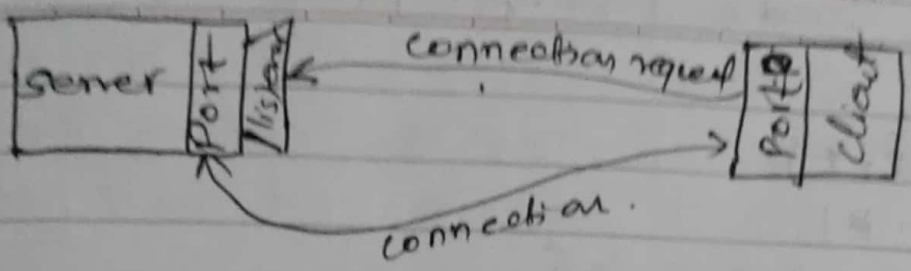
* sockets are created and used with a set of programming requests or "function calls" sometimes called the sockets application program interface (API)

* Normally a server runs on a specific computer and has a socket that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request.

* The client knows the hostname of the machine on which the server is running & the port number on which the server is listening. The client needs to identify itself to the server hence, it ~~not~~ binds to a local port number that it will use during connection.

* If connection is established, the server gets a new socket bound to the local port number and client also creates a socket if connection is established & both can use the saket to communicate with the server.

GURUKUL



TCP socket programming:

* To initiate a TCP session a server and a client are required. Firstly a server is set up to listen at a given port. When the client attempts to connect that port the connection is successful and both the server & client have an instance of the socket class.

* We import socket class that resides in java.net package i.e. import java.net.*; . We use 'localhost' that corresponds 127.0.0.1 . i.e. both the client & server are running on the same computer. The server.java & client.java communicate via. sockets

server

P.T.O

pdf (Duke Uni) socket

Date: Page:

sever code (server.java)

```
import java.net.*;
import java.io.*;
public class SimpleServer {
    public static void main(String args[]) throws IOExcep
        ServerSocket s = new ServerSocket(1234);
        Socket s1 = s.accept();

        OutputStream s1out = s1.getOutputStream([redacted]);
        DataOutputStream dos = new DataOutputStream(s1out);
        dos.writeUTF("Hello, Testing Socket");
        dos.close();
        s1out.close();
        s1.close();
    }
}
```

client code (Client.java)

```
import java.net.*;
import java.io.*;
public class SimpleClient {
    public static void main(String args[]) throws IOExcep
        Socket s1 = new Socket("localhost", 1234);
        InputStream s1IN = s1.getInputStream();
        DataInputStream dis = new DataInputStream(s1IN);
        String st = new String(dis.readUTF());
        System.out.println(st);
        dis.close();
    s1IN.close(); s1.close(); } }
```

TCP sockets [Previous ppt]

→ Register Service on port 1234
→ wait & accept a connection

→ Get a communication stream associated with socket.

→ send a string
→ close the connection but not socket server.

→ open connection to server at port 1234
→ read the input file from the socket

→ close the connection & exit.

Way2java

Date: Page:

UDP socket programming

* The UDP protocol does not use I/O streams in data transmissions unlike TCP/IP. The data is divided into number of packets and each packet is known as datagram.

* To handle datagrams, the java.net package includes classes like Datagram packet & Datagram socket. Datagram socket is capable of both sending & receiving the datagram packets with methods send() & receive().

* A datagram packet is represented as an array of bytes. and the UDP protocol doesnot have the concept of server socket.

* A simple client-server interaction in which the server sends system time to client & client displays it using UDP.

server side (server.java)

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

The address of the system should be obtained as an object of InetAddress.

```
DatagramSocket ds = new DatagramSocket();
InetAddress address = InetAddress.getLocalHost();
System.out.println("server ready");
while (true)
{   Thread.sleep(1000);
    Date currentDate = new Date();
    String s1 = currentDate.toString();
    byte arr[] = s1.getBytes();
    DatagramPacket dp = new DatagramPacket
                        (arr, arr.length, address, 2000);
    ds.send(dp);
}
}
```

The date obj is converted into byte array with getBytes() method of String class

The datagram packet constructor takes four parameters; name of byte array, length of byte array to be sent, system address, port number

send datagram packet to destination.

Client side (Client.java)

```
import java.net.*;
import java.io.*;
public class receiveTime{
  public static void main(String args[]) throws SocketException, IOException
  {
    DatagramSocket ds = new DatagramSocket(2000);
    DatagramPacket dp;
    while (true)
    {
      byte arr1[] = new byte[100]
      dp = new DatagramPacket(arr1, arr1.length);
      ds.receive(dp);
      byte arr2[] = dp.getData();
      String str = new String(arr2);
      S.o.println(str);
}}}
```

byte array obj of size 100, enough to store date sent by server.

byte array arr1 is passed to DatagramPacket constructor

receives the date sent by server into byte array object arr1

dp has the data sent by the server, getData() returns byte array that contains the message

To print the date, it is converted to string

UDP socket VS TCP socket:

Connection: UDP is connectionless protocol, TCP is a connection oriented protocol

Function: UDP is limited to message transport or transfer between multiple programs while TCP can transport message across the Internet.

Usage: UDP is suitable for applications that need fast, efficient transmission such as games and is useful for servers that answer small queries. TCP is suited for applications that require high reliability and transmission time is relatively less critical

Protocols: UDP is used by DNS, DHCP, VOIP, TFTP, SNMP, RIP
TCP is used by HTTP, HTTPs, FTP, SMTP, Telnet

order of data packets: UDP has no any inherent order & packets are independent of each other.
TCP rearranges data packets in the order specified

Speed of Transfer: The speed of UDP is faster since error recovery is not attempted.
TCP is slower than UDP.

Reliability: There is no any guarantee that the messages or packets would reach the destination. in UDP.
The TCP is reliable & data arrives in the order in which it was sent.

Header size	TCP header size is 20 bytes UDP header size is 8 bytes
streaming of data	Data is read as a byte stream, no any distinct indications are transmitted to signal message boundries in TCP. In UDP the packets are sent individually and checked for integrity only if they arrive.
weight	UDP is lightweight. There is no ordering of messages, no tracking connections etc. TCP is heaveweight. TCP requires three packets to set up a socket connection before data can be sent.
Data flow cntrl	UDP doesnot have option for flow control TCP does flow control and handles reliability and congestion control
Error checking	UDP does error checking but simply discards erroneous packets. Error recovery is not attempted. TCP does error checking & error recovery
Acknowledgement	UDP has no acknowledgement segments TCP has acknowledgement segments.

GURUKUL

UNIT 8 : RMI and CORBA

Date: Page:

RMI :

* The Remote Method Invocation is an API that provides a mechanism to create distributed application in Java. The RMI allows an object to invoke methods on an object running in another JVM.

* RMI can also be referred to as a client & server. The client requests the method running on server while server creates an object & makes it accessible remotely.
The object of server is called remote object and the server uses 'rmiregistry' to register the object available to the clients.

* The RMI provides the remote communication between the applications using two objects stub & skeleton.

USING RMI to develop a program

* The client application needs only two files, remote interface and client application. The client application invokes methods on the proxy object, RMI sends the request to the remote JVM. The return value is sent back to the proxy object & then to client application.

* The steps can be given as:

1) create the remote interface

for creating the remote interface, extend the remote interface and declare the RemoteException with all the methods of remote interface.

```
import java.rmi.*;
public interface Adder extends Remote{
public int add(int x, int y) throws RemoteException;
}
```

2) provide the implementation of the remote interface.

for providing the implementation of the Remote Interface, we need to,

- either Extend the UnicastRemoteObject class
- or use the exportObject() method.

```
import java.rmi.*;
import java.rmi.server.*;
```

USING RMI to develop a program

* The client application needs only two files, remote interface and client application. The client application invokes methods on the proxy object, RMI sends the request to the remote JVM. The return value is sent back to the proxy object & then to client application.

* The steps can be given as:

1) create the remote interface

for creating the remote interface, extend the remote interface and declare the RemoteException with all the methods of remote interface.

```
import java.rmi.*;
public interface Adder extends Remote{
public int add(int x, int y) throws RemoteException;
}
```

2) provide the implementation of the remote interface.

for providing the implementation of the Remote Interface, we need to,

- either Extend the UnicastRemoteObject class
- or use the exportObject() method

```
import java.rmi.*;
import java.rmi.server.*;
```

```
Public class AdderRemote extends UnicastRemoteObject
                implements Adder{

  AdderRemote() throws RemoteException{
     super();
  }

  Public int add(int x, int y)
   { return x+y;}
}
```

3) create the stub & skeleton objects using the rmic tool.

The rmic tool invokes the RMI compiler and creates the stub & skeleton objects.

```
rmic AdderRemote
```

4) start the registry service using the rmiregistry tool

we specify the port number, if it is not specified default will be used. Here we use port 5000.

```
rmiregistry 5000
```

5) create and run the server program
now the rmi services need to be hosted in a server process. The naming class provides methods to get & store the remote object.

```
import java.rmi.*;
import java.rmi.registry.*;
public class MyServer{
public static void main(String args[]){
    try{
        Adder stub = new AdderRemote();
        Naming.rebind("rmi://localhost:5000/service",stub
    }
    catch(Exception e){
    System.out.println(e);}
}
}
```

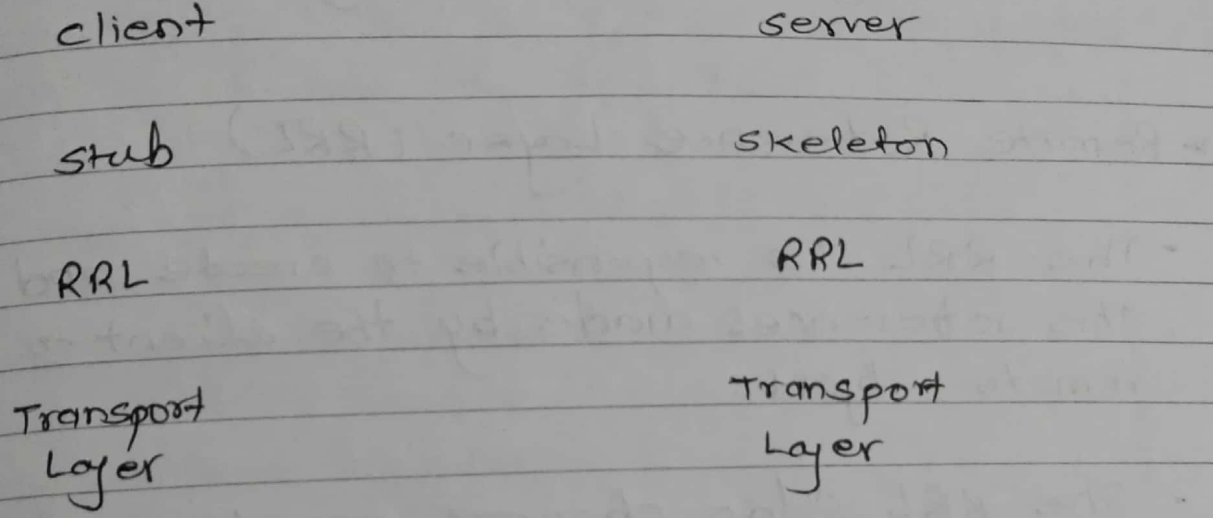
6) create and run the client program.
At the client we obtain the stub object by ~~looking~~ using the lookup() method of the Naming class and invoking the method on this object.

```
import java.rmi.*;
public class Myclient{
public static void main(String args[]){
```

```
try{
  Adder stub = (Adder) Naming.lookup("rmi://localhost
                                      :5000/service");
  System.out.println(stub.add(34,4));
}
catch(Exception e)
{ System.out.println(e); }
}
}
```

RMI Architecture

* The interface that the client & server objects use to interact with each other is provided through stubs/skeleton, remote reference and transport layers.



* stub/skeleton Layer.

- The RMI provides remote communication between applications using two objects stub & skeleton.

- stub acts as a gateway for client
- skeleton acts as gateway for server.
- stub communicates with skeleton to pass request to the remote object & skeleton communicates with stub to pass the result to client.
- Network related code is placed in stub and skeleton.

- Main job of stub/skeleton is to transmit the data to remote reference layer in form of marshal streams.

- Per stub and skeleton are responsible for marshalling and unmarshalling.

* Remote Reference Layer (RRL)

- The RRL is responsible to create and manage the references made by the client to the remote object.

- The RRL also chooses specific remote reference protocol independent of the stubs. Each remote object implementation chooses it's own reference protocol that operates on it's behalf.

- It also transmits data to the transport layer via the abstraction of stream oriented connection.

* Transport Layer

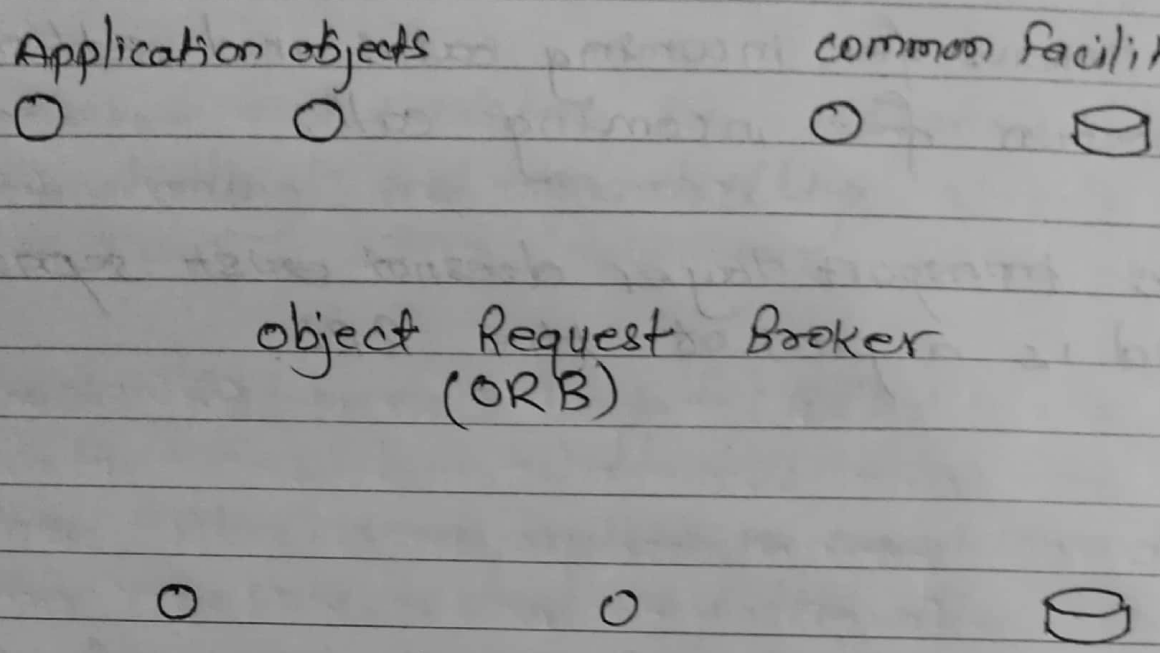
- The transport layer is responsible for actually setting up connections and handling the transport of data from one machine to another.

- It also monitors for the 'liveners' of the existing connection and setup new connections
- It listens for incoming calls and setting up connection for incoming call.
- The transport layer doesnot exist separately and is a part of the RRL.

Common object Request Broker Architecture (CORBA)

* The CORBA is a standrad developed by the object Management Group (OMG) to provide interoperabitity among distributed objects.
* The Object Management Group (OMG) is responsible for defining CORBA. The OMG comprises over 700 companies and organizations, including almost all the major vendors and developers of distributed object technology.
* CORBA is often described as a "software bus" because it is a software-based communications interface through which objects are located and accessed.

CORBA architecture



* Data communication from client to server is accomplished through a well defined object-oriented interface.

* The ORB determines the location of the target object, sends a request to that object and returns any response back to the caller.

* Using this object oriented technology, developers can take advantage of features such as inheritence, encapsulation, polymorphism and runtime dynamic binding

* The Interface Definition Language (IDL) is the OMG standard for defining the Interface. an Interface gives the services provided by the object.

* IDL defines the modules, interfaces, and operations for the applications and IDL is not considered as a programming language.

* The CORBA specification dictates there shall be an ORB through which an application would interact with other objects

RMI Vs CORBA :

Date: Page:

i) RMI is a Java specific technology. CORBA has implementations for many languages. CORBA can be used to share objects between programs written in different languages (eg. C++ & Java).

ii) CORBA uses IDL to separate interface from implementation. RMI just uses Java interfaces.

iii) Since CORBA is not tied to a particular language, the data types do not always map exactly to the types used by programming language. (long in IDL is an int in Java).

iv) RMI programs can download new classes from remote JVMs but CORBA doesnot have this code sharing mechanism.

v) CORBA can efficiently perform necessary synchronization for concurrent code but we have do the synchronization explicitly while using RMI.

vi) RMI can be configured to operate over Internet Inter-ORB protocol (IIOP), an protocol used by CORBA.

UNIT 7: Servlets & Java Server Pages (JSP)

Q1/3: What is servlet? Differentiate it with JSP. Discuss life cycle of servelet in detail.

Servlets are small programs that execute on the server side of a web connection. It is a java programming language class used to extend capabilities of servers that host applications accessed via a request-response programming model.

The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets.

The difference between JSP and servlet are as follows:

i) Servlet is a server-side program whereas ~~JSP~~ JSP is a webpage scripting language.

ii) It is easier to code in JSP but a bit difficult to write in servlet because it involves http request and response

iii) JSP runs slower compared to a servlet as it takes compilation time to convert into java servlets. Servlets run farster than JSP.

iv) Servlets are best for use when there is more procesing and manipulation involved whereas JSP is preferred when there is not much procesing.

v) The advantage of JSP over servlets is that we can build custom tags which can directly

UNIT 7: Servlets & Java Server Pages (JSP)

Q1/B: what is servlet? Differentiate it with JSP. Discuss life cycle of servelet in detail.

servlets are small programs that execute on the server side of a web connection. It is a java programming language class used to extend capabilities of servers that host applications accessed via a request-response programming model.

The javax.servlet and javax.servlet.http packages provide interfaces and classes for writing servlets.

The difference between JSP and servlet are as follows:

i) Servlet is a server-side program whereas ~~JPS~~ JSP is a webpage scripting language.

ii) It's easier to code in JSP but a bit difficult to write in servlet because it involves http request and response

iii) JSP runs slower compared to a servlet as it takes compilation time to convert into java servlets. Servlets run farster than JSP.

iv) Servlets are best for use when there is more procesing and manipulation involved whereas JSP is preferred when there is not much procesing.

v) The advantage of JSP over servlets is that we can build custom tags which can directly

call Java beans where there is no such custom tag facility in servlets.

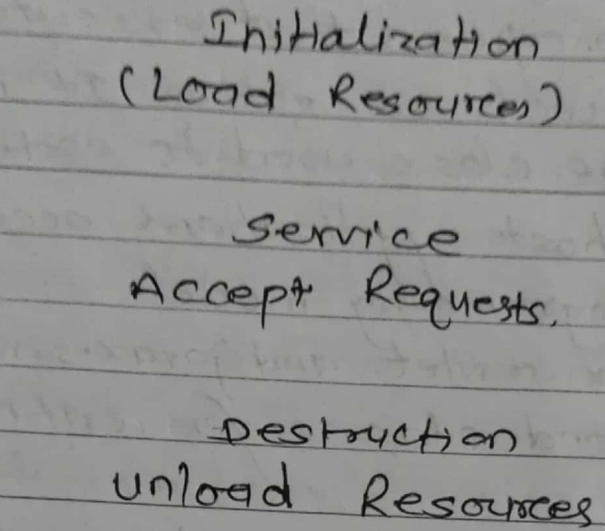


Fig: Diagram of the servlet life cycle.

Servlets follow a three-phase life: initialization, service and destruction.

i) Initialization: It is the first phase of the servlet life cycle and represents the creation and initialization of resources the servlet may need to service requests. All servlets must implement the javax.servlet.*; which defines the init() method of servlet interface.

ii) Service:

This phase represents all interactions with requests until the servlet is destroyed.

The service() method is invoked once per a request and is responsible for generating the response to that request.

iii) Destruction:

It is the third phase when a servlet is being removed from use by a web container the servlet interface defines the destroy() method to correspond the destruction life cycle phase.

What is JSP? write it's advantages over ASP/servlet. Discuss about JSP architecture.

Java server pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building web-based applications.

JSP have access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than servlet because we can separate designing & development It also provides some additional features such as Expression Language, custom Tag etc.

The advantages of JSP over servlet are as follows

1. Extension to servlet: JSP technology is the extension to servlet technology. we can use all the features of servlet in JSP.
   we can also use implicit objects, predefined tags, expression language and custom tags in JSP, that makes JSP development easy.

2. Easy to maintain: JSP can be easily managed because we can easily separate our business logic with presentation logic. In servlet technology, we mix our business logic with the presentation logic.

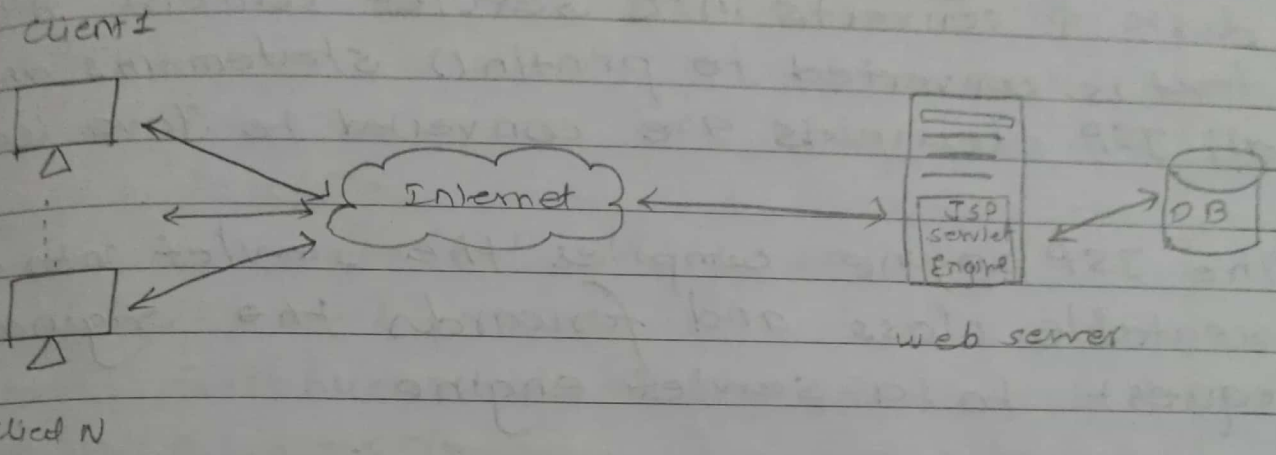
3. Fast Development (NO recompilation & redeployment): If JSP page is modified, we don't need to recompile & redeploy the project.
   The servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4. Less code than servlet: In JSP, we can use a lot of tags such as action tags, jstl, custom tags etc. that reduces code. Moreover we can use EL, implicit objects etc.

JSP Vs ASP:

1. JSP is developed by sun Microsystems while ASP is from Microsoft. JSP is free while ASP is paid.

2. JSP code is compiled at runtime, while ASP code is interpreted.



* The web server needs a JSP engine i.e a container to procerses JSP pages. The JSP container is responsible for intercepting requests for JSP pages.

* The JSP container works with the web server to provide the runtime environment and other service a JSP needs. It can detect what elements are the parts of JSPs.

JSP processing:

* As a normal page, browser sends HTTP request to the web server.

* The web server recognizes that the HTTP request is for a JSP page & forwards to JSP engine.

* The JSP engine loads the JSP page from disk & converts into servlet content. All text is converted to println() statements and all JSP elements are converted to Java code.

* The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.

* The servlet engine loads the servlet class and executes it, the output is produced in HTML format. The output is further passed on the web server by the servlet engine inside an HTTP response.

* The web server forwards the HTTP response to browser in terms of static HTML content.

* The web ~~pro~~ browser handles the dynamically generated HTML page inside the HTTP response exactly as it it were a static page.

Simple JSP program to display "Tribhuvan University" 10 times.

```
<HTML>
  <HEAD>
    <TITLE> Simple JSP program </TITLE>
  </HEAD>

  <BODY>
     <P>
      <!--<%
            for(int i=0; i<10; i++){
      %> -->
       Tribhuvan University <BR>
       <!--<%
          }
       %> -->
     </P>
  </BODY>
</HTML>
```

UNIT: 6 : Java Beans

Date: Page:

* A Java Bean is a software component that has been designed to be reusable in a variety of different environments. There is no restriction on the capability of Bean. It may perform a simple function, such as checking the spelling of a document, or a complex function, such as stock analysis

* Beans are important because they allow us to build complex systems from software components. Java Beans defines an architecture that specifies how these building blocks can operate together.

* Java Beans is an object oriented programming interface from sun Microsystems that lets you build re-usable applications or program building blocks called components that can be deployed in a network on any major OS platform.

Java bean vs Java class

Date: Page:

Java bean is also a class which normally doesnot have procesing logic. but the major differences can be given as:

- It should have a public default constructor i.e a constructor that takes no arguments.
- It should be serializable.
- A Java ben has getter and setter methods to get and set it's properties.
- core of the object is member elements and not operations which can be read or written.

Hence a Java bean is a simple Java class that conforms to some conventions. A Bean can also have associated support classes; a BeanInfo class providing information about bean, it's properties & it's events.

Advantages of Java Beans:

* Bean obtains all the benefits of Java's "write once, run-anywhere" paradigm.
* The properties, events, and methods of a Bean are exposed to an application builder tool and can be controlled.
* A Bean may be designed to operate correctly in different locales, which makes it useful in global markets.
* Auxiliary software can be provided to help a person configure a Bean. This software is only needed when the design-time parameters for that component are being set.
* The configuration settings of a Bean can be saved in persistent storage & restored at a later time.
* A Bean may register to receive events from other objects and can generate events that are sent to other objects.

Bean writing Process:

1. Create a directory for the New Bean

   Start NetBeans
   choose File >> New project
   specify the project Name, Location, Folder

2. Create a source file for the Bean.

   The source file is the code for the bean.
   we place all our program code in source file.
   A simple student bean with few properties:

```
Public class studentBean implements java.io.serializab
    private String firstname = null;
    private string lastname= null;
    private int age = 0;

    public studentBean(){
    }
    public string getFirstname(){
        return firstName; }
    public string getLastname(){
        return lastName; }
    public int getAge(){
        return age; }
```

```
public void setfirstname(string firstname){
  this.firstname = firstname;
}

public void setLastname(string lastname){
  this.lastname = lastname;
}

public void setAge(int age){
  this.age = age;
}
}
```

3. Compile the source code for the New Bean
   we compile the source code to create a class file. we use the command 'javac' with the filename of bean source file.

```
javac student.java.
```

4. create a Manifest file
   To create a manifest file, switch to the 'e:\bdk\demo' directory. This is the directory in which the manifest files for the BDK demos are located. put the source code for manifest file in the file 'student.mft' It is shown as;

Date: Page:

Name: sunw/demo/student/student.class
Java-Bean: True

This indicates that there is one .class file in the JAR file and that is a Java Bean.

5. Generate a JAR file

we finally generate a jar file that will be used in the Bean development kit for the testing/using purpose.

6: start the Bean development kit.

7: Test the JAR file,

Java Thread Life cycle:

* A thread goes through various stages in it's life cycle. The life cycle of the thread in java is controlled by JVM. A thread can be in one of the five states which are as follows:

a) New: A thread begins its life cycle in the new state. It remains in this state until the program starts the thread. It is also referred to as a born thread.

b) Runnable: The thread is in runnable state after invocation of start method. A thread in this state is considered to be executing its task

c) Waiting: sometimes, a thread transists to the waiting state while the thread waits for another thread to perform a task. A thread transists back to the runnable state only when ~~not~~ the another thread signals to continue executing.

d) Timed waiting: A runnable thread can enter the timed waiting state for a specified interval of time. A thread in this state transists back to the runnable state when that time interval expires.

e) Terminated (Dead): A runnable thread enters the terminated state when it completes its task or otherwise terminates.

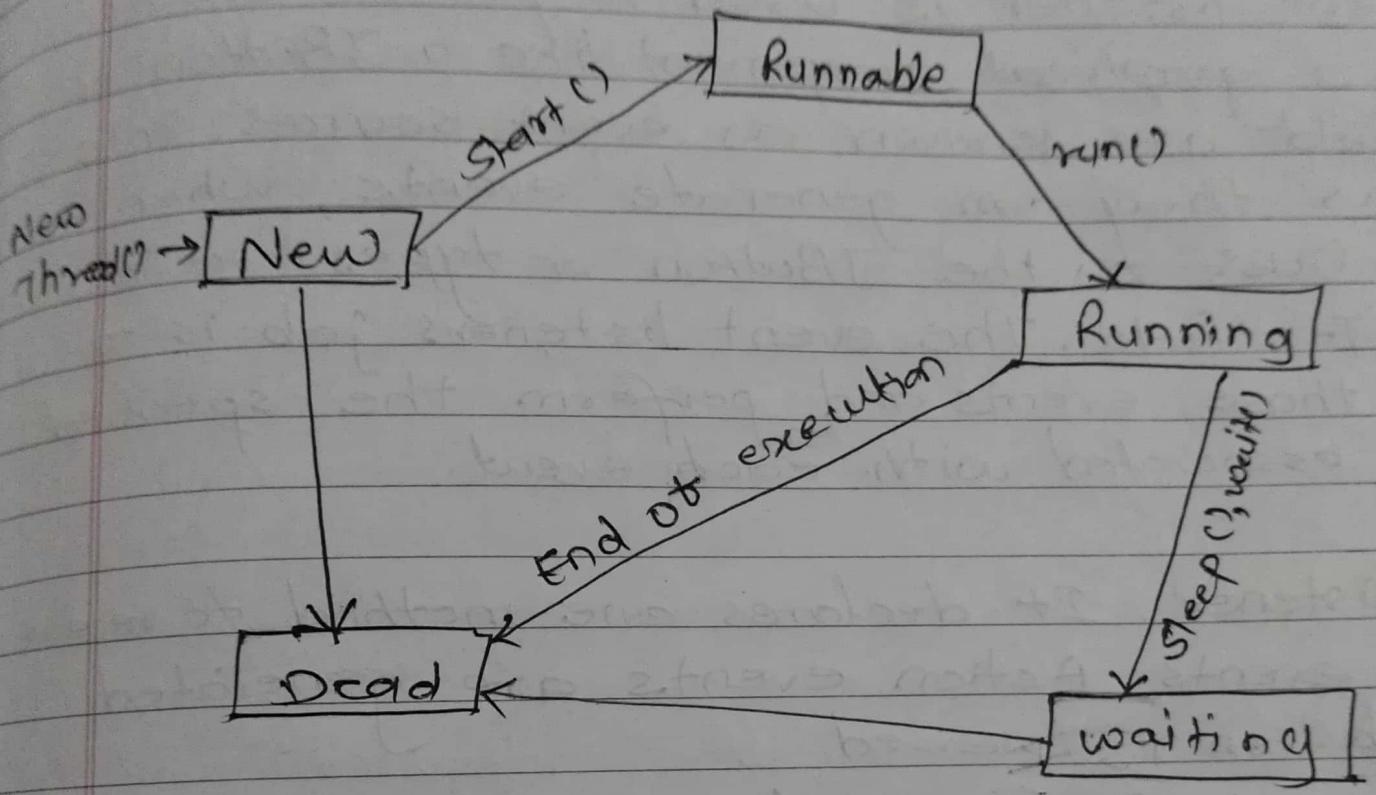


Fig: steges of Thread life cycle.

2070/11 : Discuss the role of event listeners to handle events with suitable example (5)

An event listener is used to process events. For eg: a graphical component like a JButton or JTextfield are known as event sources. This means they can generate events, when a user clicks on the JButton or types text into the JTextfield. The event listeners job is to catch those events and perform the specified task associated with each event.

1) ActionListener: It declares one method to receive action events. Action events are generated when button is pressed.
   void actionPerformed (actionEvent ae)

2) AdjustmentListener: It declares one method to receive adjustment events. Adjustment events are generated when a scrollbar is manipulated.

3) ItemListener: It defines one method to recognize when a checkbox or list item is clicked or when choice selection is made.
   void itemStateChanged (itemEvent ie)

4) ContainerListener: It declares two methods to recognize when a component is added to or removed from container.

void componentAdded (ContainerEvent ce)
void componentRemoved (ContainerEvent ce)

5) FocusListener: It declares two methods to recognize when a component gains or looses keyboard focus.
void focusGained (FocusEvent fe)
void focusLost (FocusEvent fe)

example:

```
import javax.swing.*;
import java.awt.event.*;

public class EventDemo extends JFrame
        implements ActionListener{

    JTextField tf = new JTextField(15);

    public static void main(String Args[])
    {
        EventDemo ed = new EventDemo();
        ed.edemo();
        ed.setSize(300,400);
        ed.setVisible(true)
        ed.setDefaultCloseOperation(EXIT_ON_CLOSE)
    }
```

Date: Page:

```
publia void edemo()f
JButton b1 = new JButton ("click me");
  add (tb);
  add (b1);
  b1. addActionListener (this);
  }

public void actionPerformed (ActionEvent ae)
 {
    tf. setText ("welcome");
 }
}
```

[Explain program if needed]