T.Y. B.Sc. (IT) : Sem. V

Advanced Java

Time : 2½ Hrs.] Prelim Question Paper Solution [Marks : 75

Q.1 Attempt the following (any TWO) [10]

Q.1(a) What are adapter classes in Java? Why are they used? [5]

(A)
- An adapter class provides the default(empty) implementation of all methods in an event listener interface.
- One can define a new class by extending one of the adapter classes and implement only those events relevant to the program.
- Example

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class AdapterDemo extends MouseAdapter
{
    public void init()
    {
        addMouseListener(new MyMouseAdapter(this));
        addMouseMotionListener(new MyMouseMotionAdapter(this));
    }
}
class MyMouseAdapter extends MouseAdapter
{
    AdapterDemo adDemo1;
    public MyMouseAdapter(AdapterDemo adDemo2)
    {
        adDemo1 = adDemo2;
    }

    //handles mouse clicked and ignores all other events.
   public void mouseClicked(MouseEvent me)
    {
        adDemo1.showStatus("Mouse clicked");
    }
}
```

Use :
- Adapter classes are very useful when one wants to process only few of the events that are handled by a particular event listener interface.

Q.1(b) Explain the java event delegation model. [5]

(A) Event handling mechanism is based on the delegation event model which is the standard method in event driven based java program to generate and handle events.
- General events are sending to one or more listeners.
- The listener waits for an event to occur and it process the same and then returns.
- In this model the program that creates the user interface server is separated out from methods the processes the events.
- Thus, user interface is capable to "delegate" the event processing.
- An event source provides events and dispatches them to all registered event listener.
- A single class can implement multiple listener interface.

(i) Events

- Event occurs while processing a button, entering a character via. the keyboard, selecting an item is a list or clicking the mouse.

(ii) Event Sources

- Multiple events can be generated by a single event source.
- Multicasting event: UI sources allow more than one listener to register.
- Unicasting the event: UI sources allow only and listener to register.

(iii)Event listeners:

- To process event handling implement the methods and override with the necessary logic.
- java·awt·event package defines the event listener and its associated methods.

Q.1(c) Write AWT based Java program that will read a number from user (TextBox) and display its factorial (Label). **[5]**

(A)

```
import java. awt. *; import java.awt.event.*;
public Class ex1 extends Applet implements ActionListener
{ TextField t1, t2;
int fact =1, m;   Button b1, b2, b3; String msg;
Label ℓ1 , ℓ2  ; FactEvent e;
public void init ()
{    e = this;
     t1 = new TextField (3);
     t2 = new TextField (10);
     b1 = new Button ("find fact");
     ℓ1  = new Label ("Enter any number");
     ℓ2  = new Label ("Result"; add ( ℓ1 );
     add(t1);  add(ℓ2);  add(t2);  add(b1);
b1. add ActionListener (this); }
public void actionPerformed(ActionEvent e)
     { string str = t1.get Text();
     if(str; = " ") { int num = integer.ParseInt/(str);
for (int : = num; i >  0; i ––)
     { fact = fact * i; }
msg = "  " + fact;
t2. setText (msg);
fact = 1; }}}
/*<applet code = FactEvent width = 400
height = 400 > < / applet>*/
```

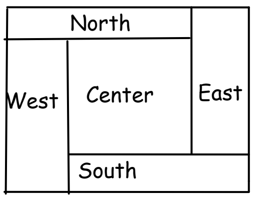
Q.1(d) What are the different types of layout in java? Explain BorderLayout. **[5]**

(A) Several AWT and Swing classes provide Layout manager for general use:

– BorderLayout
– CardLayout
– FlowLayout
– GridLayout

BorderLayout is the default Layout of the frame.

- It splits the container into five distinct sections where each section can hold one component.
- The five section of BorderLayout are known as NORTH, SOUTH, EAST, WEST and CENTER as shown in the diagram on the next page.
- To add components to a BL, one needs to specify which section one wants it placed.



example : JFrame f1 = new JFrame();
JPanel p1 = new JPanel();
f1.add(P1, BorderLayout.NORTH);

Q.2 Attempt the following (any TWO) [10]

Q.2(a) How can the user be made aware about the software loading process? Which component is facilitating the same? Explain with code specification. [5]

(A)
- A user can be made aware about the software loading process by viewing how much loading process is complete and how much is to be completed.
- This type of software loading progress can be shown using the Java component ProgressBar.
- It is a simple component, just a rectangle that is partially filled with color to indicate the progress of a software loading operation.
- By default, progress is indicated by a string "n%".
- A ProgressBar

  Example :

```
import java.awt.BorderLayout;
import java.awt.Container;
import javax.swing.BorderFactory;
import javax.swing.JFrame;
import javax.swing.JProgressBar;
import javax.swing.border.Border;
public class ProgressSample
{
    public static void main(String args[])
    {
        JFrame f = new JFrame("JProgressBar Sample");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container content = f.getContentPane();
        JProgressBar progressBar = new JProgressBar();
        progressBar.setValue(25);
        progressBar.setStringPainted(true);
                Border border = BorderFactory.createTitledBorder("Reading...");
        progressBar.setBorder(border);
        content.add(progressBar, BorderLayout.NORTH);
        f.setSize(300, 100);
        f.setVisible(true);
    }
}
```

Q.2(b) What are trees in java swings? Explain how nodes are created in a tree. [5]

(A) The JTree class is used to display the tree structured date or hierarchical date. It has a 'root node' at the top most which is a parent for all nodes in the tree. It inherits JComponent class.

The node is represented in swing as TreeNode which is an interface. The interface Mutable TreeNode extends this interface which represents a mutable node. Swing API provides implementation of this interface in the form of DefaultMutableTreeNode class; which is used to represent the node. This class has a handy add() method which takes in an instance of Mutable Tree Node. So, first root node is created and then recursively nodes are added to that root.

```
p class JTreeEx extends JFrame
{
    p JTreeEx()
    {
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        DefaultMutableTreeNode everything =
        new DefaultMutableTreeNode("Everything");
        JTree arm = new JTree (everything);
        DefaultMutableTreeNode animal=
        new DefaultMutableTreeNode("Animal");
        everything.add(animal);
        animal.add(new DMTN("Cat"));
        animal.add(new DMTN("Dog"));
        DMTN.veg = new DMTN("Vegetables");
        everthing.add(veg);
        veg.add(new DMTN("Onion"));
        veg.add(new DMTN("Carrot"));
        Container c = getContentPane();
        c.add (arm, BorderLayout.CENTER);
        setSize(200,200);
        setVisible(true);
    }
    print static void main(String args[])
    {
        new JTreeEx();
    }
}
```

Q.2(c) Write a Java program using swing components to illustrate the use of JSplitPane. [5]

(A)

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

public class JSplitEx extends JFrame
implements ListSelection Listener
{   JList ℓ11;
    JScrollPane jsp1;
    JLabel ℓ;
    JSpiltPane P1;
ImageIcon img1, img2, img3;
    container ()

JSplit Ex()
{
    Super("planet details");
    c = getContentPane();
String [] st = {"mercury", "Venus" ; "Earth"};

ℓ11 = new JList(St);
ℓ11.setSelectionMode (ListSelectionModel. SINGLE _ SELECTION);
ℓ11. addListSelectionListener(this);
jspl = new JScrollPane (ℓ1);
img1 = new ImageIcon ("Mercury.jpg");
img2 = new ImageIcon ("venus.jpg");
img3 = new ImageIcon("Earth.jpg");
ℓ1 = new JLabel (" ");
p1 = new JSPlitpane (JSPlitPane. HORIZONTAL _SPLIT, jspl, ℓ1);
    p1. SetBounds (40, 60, 250, 150);
    c. add (p1);
    SetSize (400, 400);
    SetDefaultClose operation(3);
    SetVisible (true);
}
public void valuechanged(ListSelectionEvents)
    {
    int ch = ℓ11. getSelectedIndex()+1;
switch (ch)
    {
case 1: ℓ1.SetIcon(img1); break;
case 2: ℓ1.SetIcon(img2); break;
case 3: ℓ1.SetIcon(img3); break;
default: ℓ1.SetText("imagenotfound");
}
}
public static void main(string args [])
{
    new JSplit Ex1 ();
}
}
```

Q.2(d) Differentiate between AWT and Swings. **[5]**

(A)

	AWT	Swing
(1)	AWT stands for abstract windows toolkit	Swing is also known as JFC's (Java found actionclasses)
(2)	AWT components are called Heavy weight component.	Swings are called light weight component because swing components are developed on top of the corresponding AWT components.
(3)	AWT components require java.awt package	Swing components requires java x. swing package
(4)	AWT components are platform dependent	Swing components are made in purely java and they are platform independent.
(5)	AWT is a thin layer of code on top of the operating system	Swing is larger than AWT with advance functionality
(6)	AWT components are thread– safe	Swing components are not thread–safe.

Q.3 Attempt the following (any TWO) **[10]**

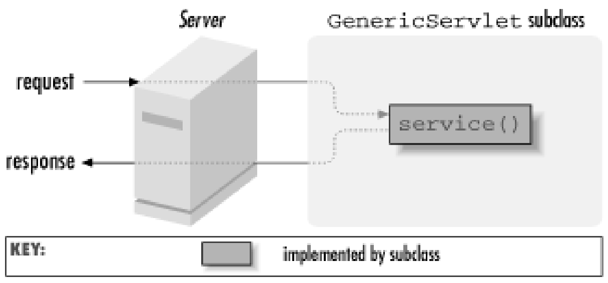
Q.3(a) Explain Generic servlet and HTTP servlet. **[5]**

(A) Generic Servlet

- GenericServlet class is direct subclass of Servlet interface.
- Generic Servlet is protocol independent.It handles all types of protocol like http, smtp, ftp etc.
- Generic Servlet only supports service() method. It handles only simple request.
- public void service(ServletRequest req,ServletResponse res).
- A generic servlet should override its service() method to handle requests as appropriate for the servlet.

- The service( ) method accepts two parameters: a request object and a response object. The request object tells the servlet about the request, while the response object is used to return a response.

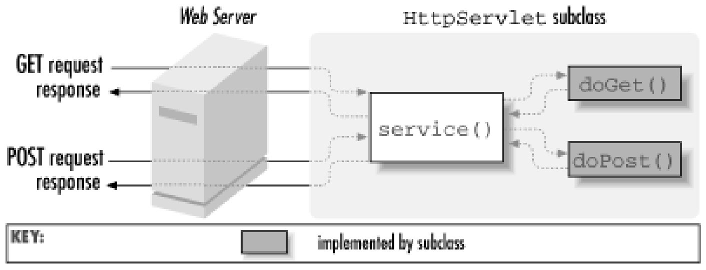
Generic Servlet only supports service( ) method.



Generic Servlet Request Handling

HttpServlet

- HttpServlet class is the direct subclass of Generic Servlet.
- HttpServlet is protocol dependent. It handles only http protocol.
- HttpServlet supports public void service(ServletRequest req, ServletResponse res) and protected void service(HttpServletRequest req,HttpServletResponse res).
- HttpServlet supports also doGet(),doPost(),doPut(),doDelete(),doHead(),doTrace(),doOptions() etc.

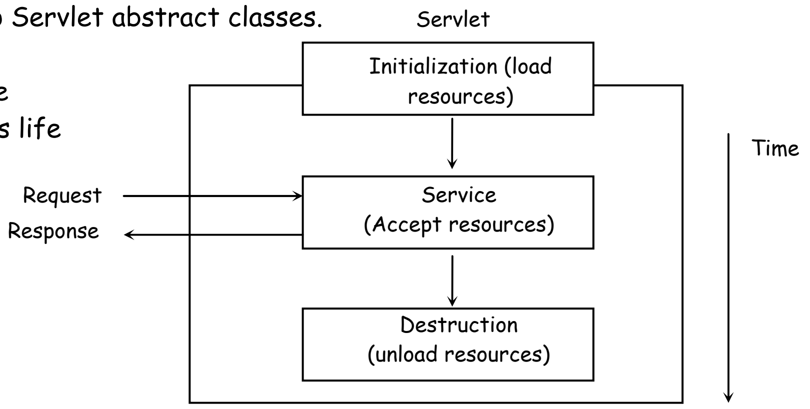


HTTP servlet request handling

- An HTTP servlet usually does not override the service() method. Instead, it overrides doGet() to handle GET requests and doPost() to handle POST requests.
- An HTTP servlet can override either or both of these methods, depending on the type of requests it needs to handle.
- The service( ) method of HttpServlet handles the setup and dispatching to all the doXXX( ) methods.

Q.3(b) Explain the life cycle of a servlet. **[5]**

(A) 1. A servlet is managed through a well defined life cycle that defines how its loaded and instantiated, is initialized handles request from clients and is taken out of service.
2. It is expressed in the API by the init, service, and destroy methods of the jawax.servlet
3. Servlet interface that all servlets must implement directly or indirectly through the Generic Servlet or Http Servlet abstract classes.


Servlet
Initialization (load resources)
Service (Accept resources)
Destruction (unload resources)
Request
Response
Time

Stages of Servlet Life cycle
It has following stages in its life
1. Initialize
2. Service
3. Destroy

1. **Initialize**
   - The init() method is designed to the called only one.
   - It is called when the servlet is first created, and not called again for each user requests.
   - It simply creates or loads some data that will be used throughout the life of the servlet.
2. **Service**
   - It is the main method to perform the actual taks.
   - The servlet container calls the method to handle the requests coming from the clients (browsers) and to write the formatted response back to the client.
   - It checks the HTTP request type (GET, POST, PUT, DELETE, etc…) and calls doGet(), doPost(), doPut(), doDelete() etc. methods as appropriate.
   - On that basis response to generated by the server to the client.
   - doGet() is used for normal request for a URL and doPost() is used for more security and private complex message.
3. **destroy()**
   - It is called only once at the end of the life cycle of a servlet.
   - After this method is called, the servlet object is marked for garbage collection.

Q.3(c) List various classes of Servlet API. Write the purpose of each. [5]

(A) Servlet API

Two packages contain the classes and interfaces that are required to build servlets. These are javax.servlet and javax.servlet.http. They constitute the servlet API.

Packages javax.servlet

Classes

- Class Generic Servlet: Implements the Servlet and Servlet config interface
- Class Servlet InputStream : Defines an input stream for reading request from a client
- Class Servlet Output Stream :Defines an outputStream for writing response to a client
- Class Servlet Exception : Indicates a servlet error occurred
- Class unavailable Exception : Indicate a servlet is either temporarily or permanently unavailable.
- Class Cookie: Represent a cookie as define by the original cookie specification.
- Class HttpServlet : Simplifies the process of writing HttpServlets
- Class HttpSessionBinding Event : Communicated to a HttpSessionBindingListener whenever the listener is bound or unbound from a HttpSession.

Q.3(d) Write a servlet to find the sum of n natural numbers. [5]

(A) Html File

```
<html>
<head>
</head>
<body>
<form action =  "additionServlet" method = "post">
    Enter number : <input type = "text" name="t1">
    <input type="submit">
</form>
</body>
</html>
```

AdditionServlet

```
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
public class Addition extends HttpServlet {
    protected void processRequest(HttpServletRequest request,
    HttpServletResponse  response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String n1=request.getParameter("t1");
        int n2 = Integer.parseInt(n1);
        try {
            int sum=0;
            int no;
            for(no=1;no<=n2;no++)
            {
                sum=sum+no;
            }
            out.println(sum);
        }
        finally {
            out.close();
        }
    }
```

Q.4 Attempt the following (any TWO) [10]

Q.4(a) What is directive in Java Server Pages? Explain page directives. [5]

(A) The JSP directives are messages that tell the web container how to translate a JSP page into the corresponding servlet.

There are three types of directives :

– page directive
– include directive
– taglib directive

Syntax of JSP directive :

```
<%@ direction attribute = "Value" %>
```

JSP page directive:

The page directive defines attributes that apply to an entire JSP page.

Syntax of JSP page directive :

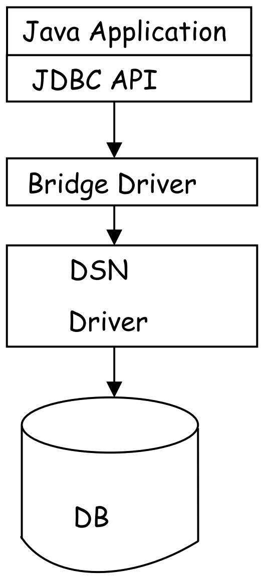
```
<%@ page attribute = "value" %>
```

Attributes of JSP page directive :

import : To import class, members of apply import = "java.util"
contentType : To indicate file type. e.g. contentType = "text/html"
extends : To extend parent class e.g. extends = "className"
info : To set information of JSP e.g. info = "Registration form"
buffer : To set buffer size in kbytes. e.g. buffer = "16kb"
language : To specify scripting lang. e.g. language = "Java"
isELIgnored : To ignore EI e.g. ELIgnore = "true"
isThreadSafe : To serialize e.g. isThreadSafe = "true"
autoFlash : bydefault it is true.
session : specifies whether JSP page participated in HttpSession.
errorPage : to show the path if exception occurs.
isErrorPage : To show whether the current JSP page is an error page or not.

Q.4(b) Describe the drivers used in Java Database Connectivity. **[5]**

(A) **Type 1** : JDBC ODBC bride driver



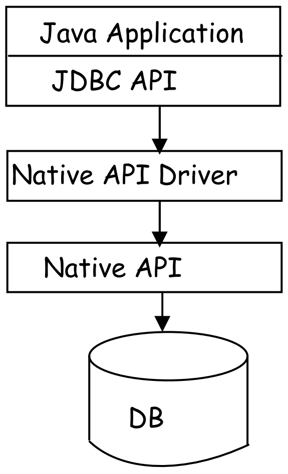
Advantages : 1) Free of Cost
2) Easy to implement

Disadvantages : 1) Slow
2) Only used with Microsoft database

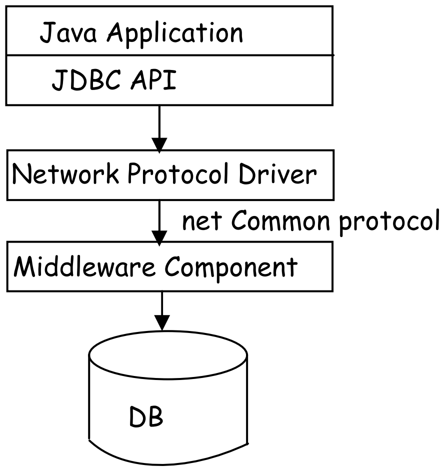
Type 2 : Native API

Advantage : Faster than Type 1

Disadvantages :Native API burdens the work.

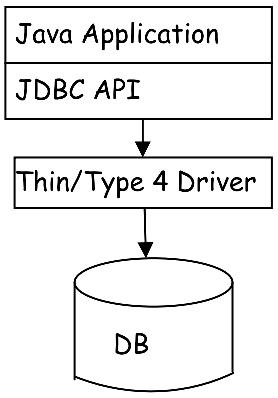


Type 3 : (Network Protocol)



Advantage : Most secured
Disadvantage : It is slower than Type 2 and Type 4

Type 4 : Database Protocol



Advantages : Fastest amonst all
Disadvantages: Costliest

Q.4(c) Explain the scrollable and updatable result sets in Java Data Base Connectivity. **[5]**

(A) Creating scrollable or updateable Result sets
While creating a statement, prepared statement or callable statements object the result set type and the concurrency type can be specified.

The method of connection class provides the format of mentioning the result set type and a concurrency type as:

i) Statement create statement int resultSetTypes int resultSet Concurrency
  - Static constant values for result set type are:
    a) Result Set·TYPE_FORWARD_ONLY
    b) Result Set·TYPE_SCROLL_SENSITIVE
  - Static constant values for concurrency type are:
    a) Result Set·CONCUR_READ ONLY
    b) Result Set·CONCUR_UPDATABLE

ii) Once statement, prepared statement or callable statement object has been created its properties can be retrieved as:
  a) int getResultSetType ( ) throws SQLException return the result SetType.

iii) Result Set Downgrade Rules
  - If result set type is TYPE_SCROLL_SENSITIVE and the JDBC driver cannot permit, then it changes to TYPE_SCROLL_INSENSITIVE

iv) Positioning and Processing Scrollable ResultSets
  - ResultSet Methods for Scrollable resultSet
    (a) Void beforefirst ( ) throws SQLException cursor position before the first row.
    (b) Boolean Last ( ) throws SQLException :
    cursor move to Lastrow.
  - ResulSet Methods for checking the current position in a scrollable result set.
    (a) Boolean is beforeFirst( ) throws SQLException. It return true if cursor is before first row.
    (b) int getRow( ) throws SQLException. It return the currents row number.

Q.4(d) Expalin the include and forward action element of JSP. **[5]**

(A) JSP Action
- JSP actions are processed during the request processing phase
- JSP also provides a powerful framework for developing custom actions, which are included in a JSP page using the taglib directive.

1. **The jsp : include Action**
   - The specification defines the include action for including static and dynamic resources in a JSP page
   - With the include directive the contents of the included resource is substituted into the JSP page at translation phase but with the include action the response of the included resource is added to the current response output stream during the request processing phase.

   Code
   ```
   <jsp: include page = "included page jsp"
   flush = "true"/>
   ```
   i) page = The resource to include
   ii) Flush = this is optional with the default value of false

2. **includeAction.jsp**
   ```
   <html>
   <head>
   <title> Include Action Test Page </title>
   </head>
   <body>
   <% @ include file = "include2.html" %>
   < jsp: include page = "include2. html"
   flush    = " true" /}
   </body >  </html>
   ```

3. **include2.html**
   ```
   <htme>
   <head>
   <title> Included HTML </title>
   </head>
   <body> This is some static text </body>
   </html>
   ```

4. **jsp: forwad Action**
   - The JSP specification defines the forward action to be used for forwarding the response to other web application resources.
   - The forward action is same as forwarding the resources using RequestDispatcherInterface code:
     `<jsp: forward page = "forwarded. html" />`
   - If forwards the request to forwarded.html
   - If content is already committed, an IllegalState Exception will be thrown

Example:

forward.html
```
<html>
<head>
<title> Forward Action Test </title>
</head>
<body>
<form method = "post" action = "forward.jsp">

<input type = "text" name = "username">
<input type = "text" name = "password">
<input type = "password" name = "password">
<Input type = "Submit" value = "Log In">
```

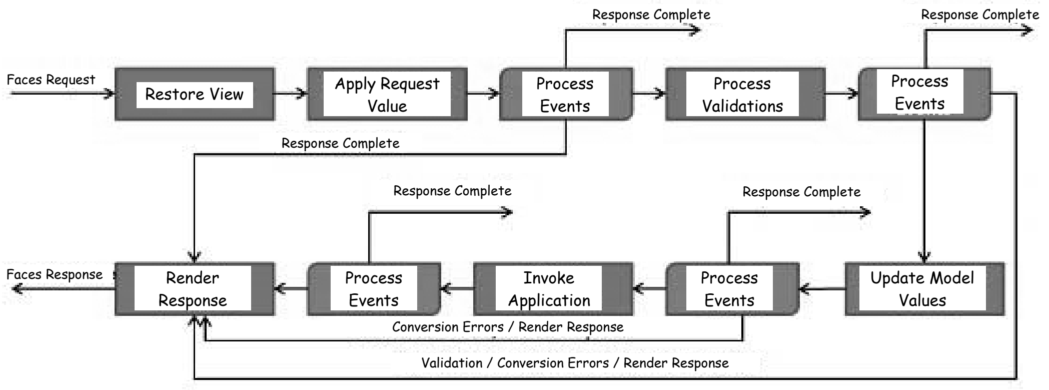
```
</form>
</body>
</html>
Forward.jsp
<html>
<head>
<title> Forwarded JSP </title>
</head>
<body>
<%
if ((request.getParameter("username")
equals ("Admin")) && (request.getParameter ("password"). equals("Admin"))){
% >
<%
}
else
{
% >
<% @ include file = "forward.html" %>
<%
}
%>
</body> </html>
```

Q.5 Attempt the following (any TWO) [10]

Q.5(a) Explain the life cycle of JSF in detail. [5]

(A) JSF application lifecycle consist of six phases which are as follows :

- Restore view phase
- Apply request values phase; process events
- Process validations phase; process events
- Update model values phase; process events
- Invoke application phase; process events
- Render response phase



The six phases show the order in which JSF processes a form. The list shows the phases in their likely order of execution with event processing at each phase.

Phase 1 : Restore view

JSF begins the restore view phase as soon as a link or a button is clicked and JSF receives a request.

During this phase, the JSF builds the view, wires event handlers and validators to UI components and saves the view in the FacesContext instance. The FacesContext instance will now contains all the information required to process a request.

Phase 2 : Apply request values
After the component tree is created/restored, each component in component tree uses decode method to extract its new value from the request parameters. Component stores this value. If the conversion fails, an error message is generated and queued on FacesContext. This message will be displayed during the render response phase, along with any validation errors.

If any decode methods / event listeners called renderResponse on the current FacesContext instance, the JSF moves to the render response phase.

Phase 3 : Process validation
During this phase, the JSF processes all validators registered on component tree. It examines the component attribute rules for the validation and compares these rules to the local value stored for the component.
If the local value is invalid, the JSF adds an error message to the FacesContext instance, and the life cycle advances to the render response phase and display the same page again with the error message.

Phase 4 : Update model values
After the JSF checks that the data is valid, it walks over the component tree and set the corresponding server-side object properties to the components' local values. The JSF will update the bean properties corresponding to input component's value attribute.
If any updateModels methods called renderResponse on the current FacesContext instance, the JSF moves to the render response phase.

Phase 5 : Invoke application
During this phase, the JSF handles any application-level events, such as submitting a form / linking to another page.

Phase 6 : Render response
During this phase, the JSF asks container/application server to render the page if the application is using JSP pages. For initial request, the components represented on the page will be added to the component tree as the JSP container executes the page. If this is not an initial request, the component tree is already built so components need not to be added again. In either case, the components will render themselves as the JSP container/Application server traverses the tags in the page.

Q.5(b) Write the benefits of EJB. **[5]**
(A) Benefits of Enterprise Javabeans:

i) Distributed applications : EJB allow building distributed application by combining components developed from different vendors.
ii) Easier application development and improved developer productivity
   - Programmers do not have to deal with low level details of transaction and state management, multithreading and resource pooling.
   - It improves the productivity of developer who can create complex applicants by focusing on business logic rather than environmental and transactional issues.
iii) Architecture independence and multiple deployments:
   - EJB architecture is independent of any specific platform, proprietary protocol or middleware infrastructure
iv) compatible :
   The EJB specification is compatible with other Java APIS and CORBA
v) Interoperable : It provides interoperability between enterprise bean and non Java applications.
vi) Customization : Enterprise bean applications can be customized without a access to the source code.
vii) Versatility and scalability:
   The EJB architecture can be used for small scale or large scale business transactions.

Q.5(c) Explain with suitable example <navigation-rule> element of faces-config.xml file of JSF. [5]

(A) Faces – config. xml

```
<? xml version = '1.0' encoding  = 'UTF – 8' ? >
< faces – config version = "2.0"
  xmlns = "http: //java. sun. com/run/ns/ javaaee"
  <mins : xsi = "http: // www. w3. org/2001/XMLSchema – instance"
  xsi :: SchemaLocation = "http. //java. sun.wn /xml/ns/javaee
  http: //java. sun. com/xml/ns/javaee/web –facesconfig _ 2-0-xsd">

<navigation – rule>
<from – view –id> //index.xhtml/ </from–view–id>
<navigation –case>
<from –outcome > welcome </from –outcome>
<to–view–id>/ welcome. xhtml</to – view –id>
</navigation –case>

<from –outcome >error </from –outcome>
<to–view–id>/error. xhtml</to–view–id>
</navigationcase>
</navigation –rule>
<navigation –rule>
<from –view–id> /error. xhtml </from–view–id>
<navigation–case>
<from –outcome> index </from –outcome>
<to – view –id> /index.xhtml </to–view–id>
</navigation case>
</navigation – rule>
<managed–bean>
<managed–bean–name> b1 </managed bean –name>
<managed–bean–class> bean.bean1 </managed–bean–class>
<managed–bean–Scope> request
     </managed –bean – Scope>
     </managed bean>
     </falls – config>
```

Navigation rule explain about the navigation of file from on location to another.

It contain a managed bean tag which speaks about the bean which is built if all business logic to be executed to access it may have a scope of Request, session, Application etc.

Q.5(d) Write the code that defines stateless bean which converts an amount from one currency to another. [5]

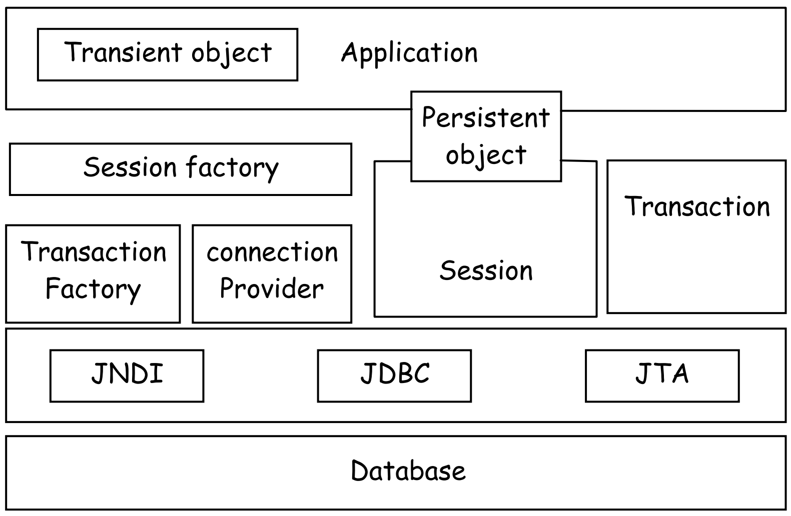
(A)

```
//index.jsp
<html>
<body>
<form method = "post" action = "test">
<b> Enter amount  in Rs </b>
<input type ="text" name = "n" value  = "      " size = "10">
<br>
<input type = "submit" value = "convert">
</form>
</body>
</html>

//test.java

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class test extends Httpservlet
{
    public void doPost (HttpServletRequest req, HttpServletRespose res)
    throws ServletException, IOException
    {
        res.setContentType ("text/html");
        PrintWriter PW = res.getWriter( );
        int rs = Integer.parseInt (req.getParameter ("n"));
        dollar d = new dollar ( );
        int a  = d.convert (rs);
        pw.println ("Amount in dollar =" + a);
        pw.close ( );
    }
}

//dollar.java

import javax.ejb.*;
@stateless
public class dollar
{
    public int convert (int rs)
    {
        int z = rs/60;
        return z;
    }
}
```

Q.6 Attempt the following (any TWO) **[10]**

Q.6(a) With suitable diagram explain the architecture of Hibernate. **[5]**

(A) Architecture of Hibernate.



Elements of Hibernate architecture

i) Configuration objects

- It represent a configuration or properties file for Hibernate
- It contains the properties to establish a database connection.

ii) SessionFactory (org.hibernate&SessionFactory)
- It consist of an optional cache of data that is reusable between tractions.
- One SessionFactory object is required per database using a separate configuration file.

iii) Session (org. hibernate. session)
- A session is used to get a physical connection with a data base.
- Persistent objects are secured and retrieved through a session object
- Session holds a cache of persistent data that are used while navigating the object.

iv) Transaction (org. hibernate. Transaction)
- A Transaction represents a unit of work with the database.
- It is an optional object. If hibernate application does not use transaction interfaces it needs to be done by applicationcode.

v) Persistent objects and collection
- It consist of persistent state and business functionalities, JavaBem/POJOs are used as persistent objects.
- It is associated with exactly one session.

vi) Transient and detached objects and collection
- It is the instances of persistent classes that are not associated with a session.

vii) TransactionFactory (org. hibernate.TranscationFactory)
- It is a factory for transcation instances
- It is not exposed to the application, but it can be extended or implemented as required.

viii)ConnectionProvider (org. hibernate.connection.connectionProvider)
- It is a pool of JDBC connections
  It abstracts the application from underlying DataSource or DriverManager.

ix) Extension Interfaces
- Hibernate includes a set of optional extension interfaces that can customize the behavior of the persistence layer.

Q.6(b) What is the use of Action component of Struts 2? Write and explain various roles of Action component. [5]

(A) Actions:
- Actions are the core for of the Struts 2 framework where each URL is mapped to a specific action. It provides the processing logic necessary to to receive the request from the user.
- Action helps the framework to determine which result needs to render the view that can be retuned.

- ***Action Mapping***
  - Input Values from the request are transferred to the properties of the action class
  - Framework calls the default method of the Action to invoke the business logic
  - If execute() method of Action class returns "Success.jsp" view is retuned to the browser

  example
  <action – mappings>
  <action name ="bean1" path =login">
  <forward name = "Success" path = "/success.jsp">

- ***Action Classes***
  - Struts 2 Action classes usually extend the ActionSupport class
  - Action Support provides default implementations for execute and input methods and also implements useful struts 2 interfaces.

- **Role of Action**

  a) Perform as a model and determines single/multiple results
  - Action act as a model by encapsulating the business logic within the execute method.

    Example : Execute() method of email validation application Public ActionForward execute(ActionMapping Mapping, ActionForm Form, HttpservletRequest req, HttpservletResponse res) throws Exception {

```
bean1 b1 = (bean1) form;
string m = b1.getEmail();
if (m == null // m.equals / " ") // m.indexof
(" @") == –1) {
        return mapping. findForward (FAILURE);
         }

    else {
    return mapping.findForward/SUCCESS);
   }
}
```

  b) Serve as Data Carrier

  Action server as data carrier from request to the view, where data get stored as JavaBean Properties

```
Private String username;
private String email;
public Sting getmail() {
    return email; }
public void SetEmail (String email)
    { this. email = email; }
public string get Username () }
    return username;
    }
public void set Username/ String username {
    this.username = username;
```

Q.6(c) What is the importance of hibernate mapping file? Explain with suitable example. [5]

(A)
- URM tool like hibernate user mapping, as a technique of associating the object properties with the corresponding columns of a database table.
- Mapping of an ORM tool can be either in the form of an XML or in the form of annotation
- A hibernate application can have one or more mapping files.
- Generally an object contains 3 properties like
  - i) Identity (objectName)
  - ii) State (object Values)
  - iii) Behavior (object Method)

Elements of hibernate mapping file is;

```
<hibernate – mapping>
<class name = "POJO classname" table = "table name in database">
<id name = "variable name" type =   "java/hibernate type">
<column name = "column name in database" / > /id >
```

Database commands in MySQL
1. Create database db;
2. use db;
3. create table guestbook (no int primary key auto_increment, name varchar(20);

POJO : Guestbook.java

```
Package hibernate;
public class Guestbook implements
java.io.Serializable; {
private Integer no;
private String name; }
```

Mapping file (Current book.hbm.xml)

```
<hibernate – mapping>
<class name = "hibernate. Guestbook"   table = "guestbook" catalog = "db">
<id name = "no" type =" java.lang.Integer">
<column name = "no" />
<generator class = "identity" />
</id>
<properly name = "name" type = "string">
<column name = "name" length = "20"/>
</property>
```

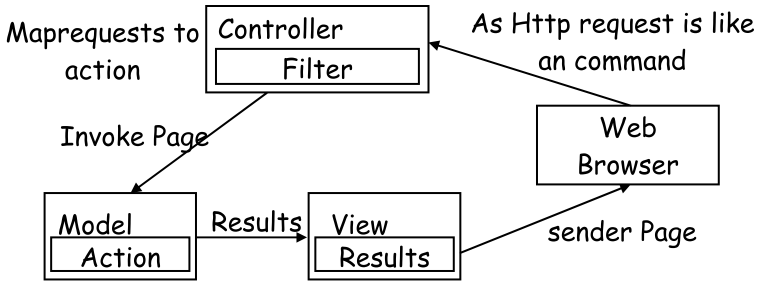
Q.6(d) Explain the application flow of Model-View-Controller architecture in struts framework. **[5]**

(A) Struts MVC design pattern :

(i) Action – Model
Model is implemented in struts using actions. Actions include the business logic and interact with the persistence storage to store, retrieve and manipulate data.

(ii) Result View
View can be a combination of result types and results . In struts the view is implemented using (JSP).

(iii) Filter Dispatcher–controller
It accepts the request and determines the appropriate Action.
- It stores all the request values into a JavaBean class
- Decides which action class to invoke for request processing.
- Model component consist of the data storage and business logic. Model returns a result to the controller that decides which output page to be sent as the response.



1. Datamodel
2. BusinessLogic

Q.7 Attempt the following (any THREE) [15]

Q.7(a) What is event? List various event classes in Java. Explain any two event classes. [5]

(A) Event :

i) An event is an object that describes a state change in a source.

ii) It can be generated as a consequence of a person interacting with the elements in a graphical user interface.

iii) Some of the activities that cause events are pressing a button etc.

Various event classes in Java

At the core of the Java's event handling mechanism are the classes that represents events. At the root of the Java event class hierarchy is Event object, which is in java. unit. It is the superclass for all events.

Following are the important classes.

1) Action Event
2) Adjustment Event
3) Component Event
4) Container Event
5) MouseEvent
6) FocusEvent
7) InputEvent
8) ItemEvent
9) MouseEvent
10) TextEvent

- **Action Event :**

  This class is defined in java.awt.event package. The ActionEvent is generated when button is clicked on the item of a list in double clicked. constructors :

  1) ActionEvent (java.lang.object source, int id, java.lang.string command): constructs an object.
  2) ActionEvent(java.lang.object source, int id, java.lang.string command, int modifiers): It is a modifier keys.

  Methods:

  1) int get Modifiers(); Return the modifier keys held down during this action Event.
  2) getActionCommand() : Return the command ing associated with the action.

- **ItemEvent :**

  The class which process the ItemEvent should implement this interface register using addItemhistener()method.

  Methods:

  1) void itemstatechanged (ItemEvent e) : Invoked when an item has been selected or deselected by the user.

Q.7(b) Create a simple table in swings to display the student details of TYBSc(IT) class. [5]

(A)

```
import java·awt·*;
import javax·swing·*;
class student extends JFrame
implements ActionListener
{   JLabel jt;
    Default Table Model dtm;
    JScrollPane jsp;
    JLabel l1, l2, l3;
    JTextField t1, t2, t3;
    JButton b1;
Student ( )
```

```
{    Super ("student");
     Object [ ] [ ] data – null;
String [ ] head = {"Student", "Roll No", "Student Name", "Student Age"};
               dtm = new Default Table Model (0,3);
               dtm = Set DataVector (data, head);
               jt = new JTable (dtm);
               jt · set Grid Color (color·RED);
               jt · set Row Selection Allowed (true);
               jsp = new JScrollPane (jt);
               JPanel p1 = new JPanel ( );
                    p1 · add (jsp);
     l1 = new JLabel ("Student RollNo");
      l2 = new JLabel ("Student Name");
       l3 = new JLabel ("Student Age");
          t1 = new JTextField (5);
          t2 = new JTextField (15);
          t3 = new JTextField (3);
JButton b1 = new JButton ("Add")
     b1 · addAction Listener (this);
JPanel P2 = new JPanel ( );
     P2·add(l1);
     P2·add(t1);
     P2·add(l2);
     P2 · add (t2);
     P2 · add (t3);
     P2 · add (b1);
Container c = getContentPane( );
     C · add (P1, BorderLayout · NORTH);
     C · add (P2, BorderLayout · (ENTER);
     setSize (600, 600);
     SetVisible (true);
SetDafault CloseOperation (EXIT_ON_CLOSE);
}
Public void actionPerformed (ActionEvent e)
     {
     int Student Roll No = Integer · parseInt (t1 · getText ( ));
     String name = t2 · getText ( );
int Student name = t2 · getText ( );
int age = Integer · parseInt (t3 · getText ( ));
Object [ ] data = {Student RollNo, Student Name, Student Age};
          dtm · addRow (data);
     t1 · setText ("  ");
     t2 · setText ("  ");
     t3 · setText ("  ");
}
Public state void main (string args [ ])
          {new Student ( ); }}
```

Q.7(c) Write a servlet program to print the factorial of a given number. **[5]**

(A)

```
fact·html
< html >
< head >
< little > factorial < /tittle >
     < /head >
```

```
        < body >
    < form method = post action = fact >
    Enter the number < input type = text   name=[ ])
    < br > < br >
    < input type = submit   value = submit >
    < input type = reset    value = reset >
        < /form > < /body > < /html >
```

```
fact.java
import javax·servlet·*;
public class fact extends HttpServlet
    {
public void doPost (HttpServletRequest request, HttpServletResponse response)
throws ServletException, IO Exception

    { response·setContentType ("text/html");
Print Writer out = response·getWriter( );
    try {
out·print ("< html > < body >");
int n = Integer·ParseInt (request.getParameter ("t1"));
out·print ("n =" + n);
int f = fact (n);
Out·print ("< br > n! ="+f);
    Out.print ("</body > < /html >");
    }
    catch (Exception e)
    {
    out·println ("Error" +e · getMessage ( ));
        } }
    private int fact (int n)
     {  if (n = 0)
        return 1;
    else
        return n* fact (n – 1);
        } }
```

Q.7(d) What is the difference between a traditional JSP page and a JSP document? Quote an example. **[5]**

(A) The JSP specification supports two basic styles of delimiting its scripting elements :
1. JSP pages
2. JSP documents

JSP pages use the traditional or shorthand syntax, whereas JSP documents are completely XML-compliant. JSP documents are also referred to as JSP pages using XML syntax.

Example : [in a regular JSP page]

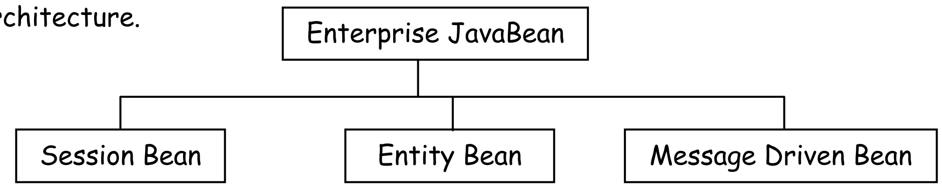
```
<%@ page language = "java" session = "true" %>
<%!
public java.util.Date.PrintDate()
{
    return (newjava.util.Date());
}
Int Counter;
%>
<html>
```

```
<head>
    <title> Displays Current Date </title>
    <head>
<body>
    The current date is : <%=PrintDate()%><br>
    This page is visited
    <%
        Counter++;
        Out.print(Counter);
    %>
    Times.
</body>
</html>
```

```
Example : [in an XML style JSP syntax]
<jsp:root  xmlns:jsp=http://java.sun.com/JSP/Page Version="2.0">
<jsp:directive.page language="java" session="true" />
<jsp:declaration>
  Public java.util.Date.PrintDate()
    {
        Return (new java.util.Date());
    }
int Counter;
</jsp:declarationte()</jsp:expression><br>
This page is visited
<jsp:scriptlet>
    Counter++;
    Out.print(Counter);
</jsp:scriptlet>
Times.
</body>
</html>
</jsp:root>
```

Q.7(e) State and explain the different types of Enterprise Java Beans. [5]

(A) Enterprise JavaBeans provide develops with a distriuted, object oriented, component based architecture.



Session Bean :

- Session beans are non-persistent enterprise beans, that encapsulate business logic onto which a client can take action programmatically across the local, remote or web service client views.
- Mapping of client and session is one to one i.e. each client is provided with its own session object.

Entity Bean :

- Entity beans are enterprise beans that contain persistent data can be saved is various persistent data stores.
- Each entity bean carries its own identity.
- Entity beans that delegate their persistence to their EJB container are called container – managed persistence (CMP) entity beans.

Message-driven bean

- Message-driven beans are enterprise beans that receive and process. Java messaging services (JMS) messages.
- Unlike session or entity beans, message driven beans have no interfaces.
- Message driven beans allows asynchronous communication between the queue and the listener and provide separation between message processing and business logic.

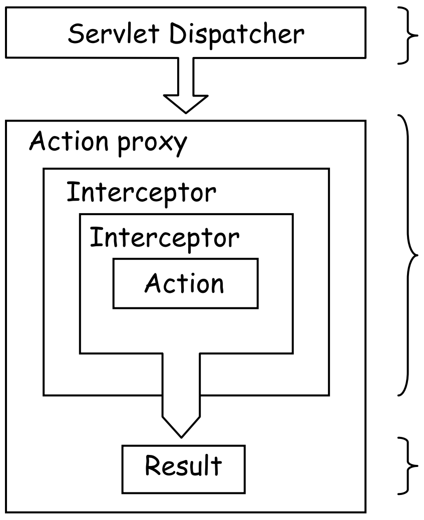
Q.7(f) What are interceptors in struts? Explain the execution flow of interceptors. [5]

(A) **Interceptor** : Interceptors allow executing separate functionality other than the action element of strut application, Interceptors can be used to

(i) Provide pre-processing logic before the action is called.
(ii) Provide post processing logic after the action is called.
(iii) Catching exceptions so that alternate processing can be performed.

Interceptors are used for the following requirements :

- It can be used to implement the exception handling. file uploading lifecycle call backs and validation etc.
- Interceptor is pluggable; to decide exactly which features an action needed support.
- Customer interceptors can be mixed and matched with the framework interceptors.
- Interceptors have access to the environmental variables and execution properties.



1. ServletDispatch instantiates a new Action Proxy and call execute()
2. Interceptors intercept the request being sent to and returning from the Action
3. Once execution of the action is complete, the request is sent to a result to render the results.

Major struts & framework Interceptors

1. Alias : Allows parameters to have different name aliases across requests.
2. Create Session : Automatically creates an HTTP session if it does not exist.
3. Exception : Allowing automatic exception handling via. redirection.
4. file upload : facilitates easy file uploading.
5. Validation : Provides validation support formations.

❑ ❑ ❑ ❑ ❑