Q.1 Attempt the following (any TWO) [10]
Q.1(a) Explain the Java event delegation model. [5]

(A) • Java Event Delegation model defines standard and consistent mechanisms to generate and process events.
• The source generates an event and sends it to one or more listeners.
• The listener simply waits until it receives an event.
• Once the event is received, the listener processes the event and returns.
• The advantage of this design is that the application logic that processes events is cleanly separated from the user interface logic that generates those events.
• The user interface element is able to ‘delegate’ the processing of an event to a separate piece of code.
• In this event model, listeners must register with a source in order to receive an event notification.
• This provides an important benefit: notifications are sent only to listeners that want to receive them.
• Events: in the delegation model, an event is an object that describes a state change in a source. It can be generated as a consequence of a person interacting with the elements in a graphical user interface. Example: Pressing a button.
• Event Source: Is an object that generates an event. This occurs when the internal state of that object change in some way. A source must register listeners in order to receive notifications about a specific type of event. Example: public void addTypeListener(TypeListener el)
• Event Listeners – a Listener is an object that is notified when an event occurs. It has two requirements, first, it must have been registered with one or more sources to receive notifications about specific type of events and second, it must implement methods to receive and process these notifications.

Example:
```java
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/*
<applet code="SimpleKey" width=300 height=300>
</applet>
*/
public class SimpleKey extends Applet
    implements KeyListener
{
    String msg = " " ;
    int X = 10, Y = 20;

    public void init() 
    {
        addKeyListener(this);
    }

    public void KeyPresses(KeyEvent ke) 
    {
        showStatus("Key Down");
    }
```
public void KeyReleased(KeyEvent ke)
{
    showStatus("Key Up");
}

public void KeyTyped(KeyEvent ke)
{
    msg += ke.getKeyChar();
    repaint();
}

public void paint(Graphics g)
{
    g.drawString(msg, X, Y);
}

Q.1(b) What are inner classes? Explain with examples. [5]
(A) Inner Classes:
Inner classes are class within Class. Inner class instance has special relationship with Outer class. This special relationship gives inner class access to member of outer class as if they are the part of outer class.

```java
import java.applet.*;
import java.awt.event.*;

<applet code="InnerClassDemo width=200 height=200"></applet>
```

public class InnerClassDemo extends Applet
{
    public void init()
    {
        addMouseListener(new MyMouseAdapter());
    }

    class MyMouseAdapter extends MouseAdapter
    {
        public void mousePressed(MouseEvent me)
        {
            showStatus("Mouse Pressed");
        }
    }
}

Q.1(c) What are the different types of layout in Java? Explain GridLayout. [5]
(A) Enlist the following types:
1) FlowLayout  2) BorderLayout  3) GridLayout
4) CardLayout  5) BoxLayout    6) GridBagLayout
7) Grid Layout

**GridLayout**

A GridLayout object places components in a grid of cells. Each component takes all the available space within its cell, and each cell is exactly the same size. If the GridLayoutDemo window is resized, the GridLayout object changes the cell size so that the cells are as large as possible, given the space available to the container.
Layout Design

Sample Program or Code snippet:
```java
import java.awt.*;
import java.applet.*;

/*
 *<applet code="SimpleKey" width=300 height=300>
 */

public class gridLayoutDemo extends Applet
{
    static final int n = 4;
    public void init()
    {
        setLayout (new GridLayout(n , n));
        setFont (new Font("SansSerif", Font.BOLD, 24));
        for(int i=0; i<n; i++)
        {
            for(j=0; j<n; j++)
            {
                int k = i*n+j;
                if (k > 0)
                add (new Button (" + k"));
            }
        }
    }
}
```

Q.1(d) Write AWT based Java program that demonstrate the use of checkbox and [5]
radiobuttons.

(A) Public class CBGroup extends Applet implements Itemlistener(

```java
String msg = " ";
Checkbox winXP, win Vista, solaries, mac;
checkboxGroup cbg;
public void init ( )
{
    cbg = new checkboxGroup( );
    winXP = new checkbox ("Windows XP", true);
    winVista = new checkbox ("Windows Vista");
    Solaries = new checkbox ("Solaries", cbg, true);
    mac = new checkbox ("mac", cbg false);
    add (winXP);
    add (winVista);
    add (Solaries);
```
add (Mac)
winXP.addItemListener (this);
winVista.addItemListener (this);
Solaris.addItemListener (this);
mac.addItemListener (this);

Public void itemStateChanged [ItemEvent] 
{
    repaint();
} 

public void paint(Graphics g) {
    msg = " current selection;"
    msg = cbg.getSelected.index () getLabels
    g.drawString (msg, 6, 100);
}

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.

(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) How do divide frame window in two parts? Explain with code specification. [5]

(A) A frame window can be divided into two parts using JSplitPane component.
- A frame can be divided into two parts either vertically or horizontally.
- In the following code, a frame is divided into two parts horizontally.
- Firstly, the object of JSplitPane is created.
- Then its setOrientation() method is used to set the orientation, either Horizontal or Vertical.

```java
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class JSplitPaneDemo {
    public static void main(String[] args) {
        JFrame frame = new JFrame("JSplitPane demo");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JSplitPane sp = new JSplitPane();
        sp.setOrientation(JSplitPane.HORIZONTAL_SPLIT);
        frame.getContentPane().add(sp, BorderLayout.CENTER);
        frame.setSize(500,500);
        frame.setVisible(true);
    }
}
```

Q. 2(c) Compare and contrast AWT and Swing. [5]

(A) Comparison between AWT and Swing:

<table>
<thead>
<tr>
<th></th>
<th>AWT</th>
<th>Swing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1)</td>
<td>Stands for Abstract Windows Toolkit</td>
<td>Swing is also called as JFC's (Java foundation classes)</td>
</tr>
<tr>
<td>2)</td>
<td>AWT components are called HeavyWeight components</td>
<td>Swings are called light weight component because swing components sits on the top of AWT components and do the work.</td>
</tr>
<tr>
<td>3)</td>
<td>AWT components require java.awt package.</td>
<td>Swing components require javax.swing package.</td>
</tr>
<tr>
<td>4)</td>
<td>AWT components are platform dependent.</td>
<td>Swing components are platform independent.</td>
</tr>
<tr>
<td>5)</td>
<td>This feature is not supported in AWT</td>
<td>One can have different look and feel in Swing</td>
</tr>
<tr>
<td>6)</td>
<td>Using AWT, one has to implement a lot of things oneself.</td>
<td>Swing has them built in.</td>
</tr>
</tbody>
</table>

Q. 2(d) Write a Java program to create different tabs using JTabbedPane. [5]

(A) import javax.swing.JTabbedPane;
- import javax.swing.*;
- import java.awt.*;
- import java.awt.event.*;
- public class JTabbedPaneDemo extends JFrame
  {
    public static void main(String args[])
    {
        JTabbedPaneDemo d = new JTabbedPaneDemo();
        JTabbedPane p = new JTabbedPane();
        p.addTab("Cities", new CitiesPanel());
    }
```
p.addTab("Colors", new ColorsPanel());
d.add(p);
d.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
d.setSize(500,500);
d.setVisible(true);
}
}
class CitiesPanel extends JPanel
{
    public CitiesPanel()
    {
        JButton b1 = new JButton("New York");
        add(b1);
    }
}
class ColorsPanel extends JPanel
{
    public ColorsPanel()
    {
        JButton b1 = new JButton("Blue");
        add(b1);
    }
}

Q.3 Attempt the following (any TWO) [10]
Q.3(a) Write a short note on JSAPI (JAVA SERVLET API). [5]

(A) • Servlet API is used to create HTTP servlets, or any kind of servlets. Servlets use
classes and interfaces from two packages: javax.servlet and javax.servlet.http.
• The Java servlet API is the class library using which requests can be processed and
responses can be constructed dynamically.
• The API thus helps define the expected interactions of a Web container and a Servlet.
• Every servlet must implement the javax.servlet.Servlet interface. Most servlets
implement it by extending one of two special classes: javax.servlet.GenericServlet or
• Unlike a regular Java program, and just like an applet, a servlet does not have a main( )
method. Instead, certain methods of a servlet are invoked by the server in the process
of handling requests. Each time the server dispatches a request to a servlet, it invokes
the servlet’s service( ) method.

Packages

<table>
<thead>
<tr>
<th>Packages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>javax.servlet</td>
<td>The javax.servlet package contains a number of classes and interfaces that describe and define the contracts between a servlet class and the runtime environment provided for an instance of such a class by a conforming servlet container.</td>
</tr>
<tr>
<td>javax.servlet.http</td>
<td>The javax.servlet.http package contains a number of classes and interfaces that describe and define the contracts between a servlet class running under the HTTP protocol and the runtime environment provided for an instance of such a class by a conforming servlet container.</td>
</tr>
</tbody>
</table>

Interfaces in javax.servlet package
There are many interfaces in javax.servlet package. They are as follows:
Servlet
ServletRequest
Q. 3(b) Explain the life cycle phases of servlet. [5]

(A) Servlet Lifecycle

The servlet lifecycle consists of the following phases:

i) Loaded    ii) Instantiated    iii) Initialized
iv) Service request  v) Destroyed  vi) Finally garbage collected
i) **Loaded** : As soon as there is a request for a servlet, it is loaded on the JVM, if it does not exist. A servlet is only loaded once.

ii) **Instantiated** : After loading the servlet, it is instantiated i.e. an object of the requested servlet class is created to service the request.

iii) **Initialized** : Post instantiation, the servlet’s `init(ServletConfig config)` method is called in order to initialize the servlet.

   The `init()` method is defined as:
   ```java
   public void init(ServletConfig config) throws ServletException
   ```

   During initialization, the servlet has access to two objects:
   - `ServletConfig`
   - `ServletContext`

   The following are the most common tasks that are implemented in the `init()` method:
   - Reading initialization parameters
   - Reading configuration data from persistent resource like a config file
   - Initializing a database driver
   - Opening a JDBC connection
   - Writing of information to a network resource

iv) **Service Request** : Only after the completion of the `init()` method, the `service()` method is invoked.

   Once the server has loaded and initialized the servlet, it is able to handle client requests through the `service()` method. Each client request to the `service()` method is run in a separate servlet thread.

   The `service()` method is defined as follows:
   ```java
   public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
   ```

   The `ServletRequest` object helps in accessing the original request data and the `ServletResponse` object provides methods that help in building a response.

v) **Destroyed** : Post servicing the request, the servlet can be unloaded by calling the `destroy()` method. All resources which were allocated by `init()` should be released by `destroy()`. The `destroy()` method is defined as

   ```java
   public void destroy() throws ServletException
   ```
This method is only called once in the lifetime of a servlet. The most common tasks in the destroy() method are:

- Synchronizing cleanup tasks such as closing open resources or closing a connection pool
- Informing other applications that the servlet will no longer be available

Q.3(c) What are servlets?  What are the advantages of servlets over CGI?  

(A) A servlet is a program that extends the functionality of a web server. Servlets receive a request from a client, dynamically generate the response and then send the response containing an HTML or XML document to the client. Java servlet technology allows defining HTTP specific Servlet classes.

1) **Efficient**: With traditional CGI, a new process is started for each HTTP request. If the CGI program itself is relatively short, the overhead of starting the process can dominate the execution time. With servlets, the Java virtual machine stays running and handles each request with a lightweight Java thread, not a heavyweight operating system process.

2) **Convenient**: Servlets have an extensive infrastructure for automatically parsing and decoding HTML form data, reading and setting HTTP headers, handling cookies, tracking sessions, and many other such high-level utilities. In CGI, this has to be explicitly done.

3) **Powerful**: Servlets support several capabilities that are difficult or impossible to accomplish with regular CGI. Servlets can talk directly to the Web server, whereas regular CGI programs cannot, at least not without using a server-specific API.

4) **Portable**: Servlets are written in the Java programming language and follow a standard API and thus are portable themselves.

5) **Inexpensive**: A number of free or very inexpensive Web servers are good for development use or deployment of low- or medium-volume Web sites. This is in contrast to many of the other CGI alternatives, which require a significant initial investment for the purchase of a proprietary package.

6) **Secure**: One of the main sources of vulnerabilities in traditional CGI stems from the fact that the programs are often executed by general-purpose operating system shells. So, the CGI programmer must be careful to filter out characters such as backquotes and semicolons that are treated specially by the shell. Implementing this precaution is harder, and weaknesses stemming from this problem are constantly being uncovered in widely used CGI libraries.

A second source of problems is the fact that some CGI programs are processed by languages that do not automatically check array or string bounds. Servlets suffer from neither of these problems. Even if a servlet executes a system call (e.g., with Runtime.exec) to invoke a program on the local operating system, it does not use a shell to do so. And array bounds checking and other memory protection features are a central part of the Java programming.
Q. 3(d) Write a servlet application to find the sum of digits of the number entered by the user through the HTML form. [5]

(A) add.html
<html>
<head>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
<form action="Addition" method="post">
  Number - <input type=text name=t1>
  <input type =submit>
</form>
</body>
</html>

Addition.java
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");
    //String n1="10";
    int n2 = Integer.parseInt(n1);
    try
    {
      int i=0;
      int sum=0,tot, rem;
      int no=1;
      while (n2>0)
      {
        rem = n2 % 10;
        tot = tot+ rem
        n2=n2/10;
      }
      out.println(tot);
    }
    finally
    {
      out.close();
    }
  }
}
Q.4 Attempt the following (any TWO) [10]

Q.4(a) Write a JDBC program that inserts values in database. [TABLE NAME : Employee, FIELDS : Empid, Name, Dept, Designation] [5]

(A) import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

//login, password, roll
public class InsertRecord extends JFrame
{
    public InsertRecord()
    {
        try
        {
            final JTextField entry = new JTextField(30);
            final JTextField t1 = new JTextField(30);
            final JTextField t2 = new JTextField(30);
            final JTextField t3 = new JTextField(30);
            final JTextField t4 = new JTextField(30);
            JButton b1 = new JButton("Insert");
            final Container con = getContentPane();
            con.setLayout(new FlowLayout());
            con.add(b1);
            con.add(t1);
            con.add(t2);
            con.add(t3);
            con.add(t4);
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection conn = DriverManager.getConnection("jdbc:odbc:shree1", ",", ",");
            System.out.println("connected");
            //public Statement statement = conn.createStatement();
            final Statement statement = conn.createStatement();
            //________________-
            //Insert a new record
            B1.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent event)
                {
                    try
                    {
                        String queryString = "insert into Employee (Empid,name, dept, Designation) values (" + t1.getText() + "," + t2.getText() + "," + t3.getText() + "," + t4.getText() + ");"
                        statement.executeUpdate(queryString);
                    }
                    catch (SQLException ex)
                    {
                    }
                }
            });
            System.out.println("Record inserted.");
        }
    }
}
Q.4(b) What is the difference between a traditional JSP page and a JSP document? [5]
Quote an example.
(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]
```html
<%@ page language="java" session="true" %>
<%! 
public java.util.Date.PrintDate()
{ 
  return (new java.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]
```html
<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()
</jsp:declaration>
```

})}
catch (Exception ex)
{

}

public static void main(String[] args)
{
  InsertRecord n = new InsertRecord();
  n.setTitle("Navigation");
  n.setSize(500, 500);
  n.setVisible(true);
}
Q.4(c) Explain scrollable and updatable resultsets in JDBC. [5]

(A) scrollable result set allows the cursor to be moved to any row in the result set. Back and forward.

```java
try {
    // Create an insensitive scrollable result set
    Statement stmt = connection.createStatement
        (ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
    // Create a sensitive scrollable result set
    stmt = connection.createStatement
        (ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);
} catch (SQLException e) {
}
```

Scrollable result set

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class ScrollableResultSetExample {
    public static void main(String[] args) {
        Connection connection = null;
        try {
            // Load the MySQL JDBC driver
            String driverName = "com.mysql.jdbc.Driver";
            Class.forName(driverName);
            // Create a connection to the database
            String serverName = "localhost";
            String schema = "test";
            String url = "jdbc:mysql://" + serverName + "/" + schema;
            String username = "username";
            String password = "password";
```
connection = DriverManager.getConnection(url, username, password);
    System.out.println("Successfully Connected to the database!");
}  
catch (SQLException e)
{
    System.out.println("Could not connect to the database " + e.getMessage());
}
try
{
    ResultSet.TYPE_SCROLL_SENSITIVE directive)
    Statement statement =
    connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
Resultset.CONCUR_READ_ONLY);
    ResultSet results = statement.executeQuery
("SELECT * FROM test_table");
    System.out.println("Cursor position " + results.getRow() + ",
    is before first ? " + results.isBeforeFirst());
    results.next();
    System.out.println("Cursor position " + results.getRow() + ",
    is first ? " + results.isFirst());
    results.last();
    System.out.println("Cursor position " + results.getRow() + ",
    is last ? " + results.isLast());
    results.afterLast();
    System.out.println("Cursor position " + results.getRow() + ",
    is after last ? " + results.isAfterLast());
    results.absolute(3);
    System.out.println("Cursor position " + results.getRow());
    results.absolute(-1);
    System.out.println("Cursor position " + results.getRow() + ",
    is last ? " + results.isLast());
}  
catch (SQLException e)
{
    System.out.println("Could not retrieve data from the database "
    + e.getMessage());
}

Updatable ResultSet:
Along with scrollable result sets, JDBC 2.0 also introduces the concept of updatable result
sets—result sets you can change.

An updatable result set enables you to perform in-place changes to a result set and have them
take effect using the current transaction. An overnight interest-assignment process for a bank
is an example of such a potential batch process. It would read in an accounts balance and
interest rate and, while positioned at that row in the database, update the interest.
JDBC 2.3 result sets have two types of concurrency: ResultSet.CONCUR_READ_ONLY and ResultSet.CONCUR_UPDATEABLE. You pass the concurrency type ResultSet.CONCUR_UPDATEABLE as the second argument to createStatement(), or the third argument to prepareStatement() or prepareCall()::

PreparedStatement stmt = conn.prepareStatement(
    *SELECT acct_id, balance FROM account*,
    ResultSet.TYPE_SCROLL_SENSITIVE,
    ResultSet.CONCUR_UPDATEABLE);

The most important thing to remember about updatable result sets is that you must always select from a single table and include the primary key columns. If you don’t, the concept of the result set being updatable is nonsensical. After all, updatable result set only constructs a hidden UPDATE for you. If it does not know what the unique identifier for the row in question is, there is no way it can construct a valid update.

Q.4(d) Explain any five implicit objects in Java Server Pages.  
(A)

JSP implicit objects are predefined Java objects that JSP contains makes available to each JSP page. During the application development, programmer can use the implicit objects directs within the JSP page without implicit duration.

- JSP Implicit Objects are also known as pre-defined variables of JSP
- Request : It is the HttpServletRequest object that can be used to Fetch the information from the client.
- Response : It is the HttpServletResponse object that can be used to send the response back to the client.
- Out : It is the PrintWriter object used to send output data to the client. i.e. out.flush( ); used to flush the stream output to the client.
- Session :
  (i) It is the ServletContent object associated with the application.
  (ii) It is created during the initialization phase of the JSP page. It will be removed by jspDestroy( ) method.
- Config : It is the ServletConfig object associated with the page. It can also get the information about the server class file generated by JSP page. Eg. config.getServletName( ) - returns the servletName.

Q.5 Attempt the following (any TWO)  
Q.5(a) Explain the life cycle of JSF in detail.

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 contain 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) Explain the architecture of Enterprise Java Beans. [5]

(A) Enterprise Beans architecture defines a model for the development and deployment of reusable Java Server components.

The EJB Architecture is composed of:
1) An Enterprise Bean Server.
2) Enterprise Bean Container that runs on these server.
3) Enterprise Beans that run in these containers
4) Enterprise bean clients
5) Other systems such as Java Naming and Directory interface (JNDI) and Java Transaction Service (JTS)

**Enterprise Bean Server:**
- Is a component transaction server. It supports the EJB server side component model for developing and deploying distributed enterprise-level applications in a multi-tiered environment.

**It provides:**
- The framework for creating, deploying and managing middle-tier logic.
- An environment that allows the execution of applications developed using Enterprise JavaBeans [EJB] components.

**The EJB server takes care of:**
- Managing and coordinating the allocation of resources to the applications.
- Security
- Threads
- Connection pooling
- Access to a distributed transaction management service.

**Enterprise Bean Container:**
- The EJB server provides one or more containers for the enterprise beans, which is called an EJB container.
- It manages the enterprise beans contained in it.

**It is responsible for:**
- Registering the object
- Providing a remote interface for the object
- Creating and destroying object instances
- Checking security for the object
- Managing the active state for the object

**Enterprise Beans:**
- Business components developed using the EJB architecture are called as Enterprise Beans.
- It is a server-side component that encapsulates the code that fulfils the purpose of the application. They can be combined with other components and rapidly produce a custom application.

**Enterprise Bean Clients:**
- An EJB client is a stand-alone application that provides the User interface logic on a client machine.
- It makes calls to remote EJB components on a Server. The EJB client needs to be informed about:
  - How to find the EJB server
  - How to interact with the EJB components.
Q.5(c) Write the benefits of using EJB. [5]

(A) 1) Complete focus only on Business Logic:
   - Enterprise Beans live and run in the server under the control of an EJB container.
   - The EJB container provides all the big infrastructure services such as Security, Concurrency, Transaction management, Networking, Resource management, Persistence, Massaging and Customization during deployment.
   - Developer can use these services with minimal effort and time, thus making writing an enterprise bean as simple as writing a Java class.
   - EJB model separates system level services from the business logic. This allows the server vendor to concentrate on system level functionalities while the developer can concentrate more on only the business logic for the domain specific application.

2) Reusable Components:
   - Each EJB is a reusable building block.
   - An EJB can be reused by assembling them in several different applications. For each application, making simple changes in deployment descriptor without the source code can customize its behavior with the underlying services.

3) Portable:
   - EJB use Java language, which is portable across multiple platforms.
   - The components can run on any platform and are completely portable across any vendor's EJB-compliant application server.
   - The EJB environment automatically maps the component to the underlying vendor-specific infrastructure services.

4) Fast building of Application:
   - The EJB architecture simplifies building complex enterprise applications.
   - With the component-based EJB architecture, the development, enhancing the functionalities and maintenance of complex enterprise applications becomes easier.
   - With its clear definition of roles and well defined interfaces, the EJB architecture promotes and supports team-based development and lessens the demands on individual developers.

5) Presents One business Logic Having Many Presentation Logics:
   - An enterprise bean typically performs a business process or represents a business entity and it is independent of the presentation logic. The EJB model allows the business developer to concentrate on business logic while Web page designer concentrates on formatting the output.
   - This separation makes it possible to develop multiple presentation logic for the same business process or to change the presentation logic of a business process without needing to modify the code that implements the business process.

6) Distributed Deployment:
   - The Enterprise beans, the business components of distributed system are deployed across multiple servers on a network. The client on the same machine as enterprise bean or client located on different system on the network can call the methods on the bean.

7) Application Interoperability:
   - EJB architecture is mapped to standards followed by CORBA, which is an industrial standard. Hence, it is relatively simple to craft an EJB and make it work with components developed in a different like VC++ and so on using CORBA. The EJB’s client view interface serves as a well-defined integration point between components built using different programming languages.

Q.5(d) What are the different types of enterprise beans? Explain. [5]

(A) Session Beans
   - A session bean object is a short lived object that executes on behalf of a single client. It is a temporary, logical extension of a client application that runs on the Application Server.
A shared bean does not represent the shared data in a database, but obtains a data snapshot. However a session bean can update data.

Session beans
- Execute for a single client
- Can be transaction aware
- Do not represent directly shared data in an underlying database, although they may access and update this data.
- Are not persistent in a database.
- Are removed if the container crashed and the client has to establish a new session.

Session beans are of three types

- **Stateless Session Beans**: these beans do not remember anything about the client between the calls of the method. It forgets about the client once the business process completes or executes. They can be used across multiple clients. These beans can implement a web service.

- **Stateful Session Beans**
  - can remember conversation between the client of the application and application itself across method calls.
  - They store data that is client specific. When a client calls the methods again the bean remembers the client's previous method call.
  - Only a single client can use a stateful session bean at a time. The state is retained for the duration of the client bean session.
  - If the client removes the bean or terminates, the session ends and the state disappears. Shopping cart is a typical example of Stateful Session Beans.

- **Singleton Session Beans**
  - Are instantiated once per application and exist for the lifecycle of the application.
  - These are helpful where a single enterprise bean instance is shared across and concurrently accessed by clients.
  - There is only one singleton bean per application. There is a pool of stateless session beans, any of which may respond to a client request.

- **Message Driven Beans**
  - Is an enterprise bean that allows enterprise application to process messages asynchronously.
  - It acts as a JMS (Java Message Service) message listener, which is smaller to an event listener except that it receives messages instead of events.
  - The messages may be sent by any component, an application client, another enterprise bean or a Web component, a JMS application or system that does not use Java EE technology.

- **Message Driven beans**
  Do not have home or component interface
  - Do not have business methods but define a message listener method, which the EJB container invokes to deliver messages.
  - Do not hold any state calls of the message listener method.
  - Could be a New customer Notification subscriber.
  - Are relatively short-lived.

Do not represent directly shared data in the database, but they can access and update this data.
Q. 6 Attempt the following (any TWO) [10]
Q. 6(a) Explain the architecture of hibernate framework in detail. [5]

(A) 

Conference Object:
- This object represents a configuration or properties file for hibernate. It is usually created once during application initialization.
- The configuration object reads the properties to establish a database connection. A Configuration object is spawned to create a SessionFactory.

SessionFactory:
- The SessionFactory is created with the help of a Configuration object during the application start up. It serves as a factory for spawning Session objects when required.
- Typically, it is created once and kept alive for later use.
- The applications that require interacting with multiple databases, multiple SessionFactory objects are created.

Session:
- Session objects are lightweight and inexpensive to create. They provide the main interface to perform actual database operations.
- All the persistent objects are saved and retrieved with the help of a Session object.
- Typically, session objects are created as needed and destroyed when not required.

Transaction:
- A transaction represents a unit of work the database.
- Any kind of modifications initiated via the session object are placed within a transaction.
- A session object helps creating a Transaction object.
- Transaction objects are used for a short time and are closed by either committing or rejecting.

Query:
- Persistent objects are retrieved using a Query object.
- Query objects allow using SQL or Hibernate Query Language (HQL) queries to retrieve the actual data from the database and create objects.
Criteria:
- Persistent objects can also be retrieved using a Criteria object.
- Criteria uses an object/method means of constructing and executing a request to retrieve objects.

Q.6(b) Explain web.xml and struts.xml files.

(A) Web.xml
- the web.xml web application descriptor file represents the core of the java web application, so it is appropriate that it is also part of the core of Struct Framework.
- in this file, Struts defines its Filedisptatcher, the Servler filter class that initializes the Struts framework and handles all requests.
- This filter can contain initialization parameters that affect what, if any, additional configuration files are loaded and how the framework should behave.
- Struts also provides an ActionContextCleanUp class that handles special cleanup tasks when other filters need access to an initialized Struts framework.

Key Initialization Parameters:
- Config – a comma delimited list of XML configuration files to load.
- actionPac scan for actkages – a comma delimited list of Java packages Actions.
- configProviders – a comma delimited list of Java classes that implement the ConfigurationProvider interface that should be used for building the configuration.
- loggerFactory – The class name of the LoggerFactory implementation.
- * – any other parameter are treated as framework constants.

Struts.xml
- The Struts framework uses this configuration file to initialize its own resources. These resources include:
  - Interceptors that can preprocess and postprocess a request.
  - Action classes that can call business logic and data access code.
  - Results that can prepare views using JavaServer Pages, Velocity and FreeMarker templates.
- The struts.xml is the core configuration file for the framework and it should be present in the class path of web application.
- This file allows to break big struts.xml file into small files and configuration files to be included as needed.

Example:
```xml
<struts>
  <include file="file1.xml" />
  <include file="file2.xml" />
</struts>
```

Q.6(c) What is Value stack in struts? State and explain the Execution Flow of value stack.

(A) Value stack is a stack of objects.
- It is a storage area that holds all of the data associated with the processing of a Request.

Flow of Value stack
- The framework receives a request and decides on the Action the URL maps to
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.

Q.7 Attempt the following (any THREE) [15]
Q.7(a) Write a Java program to handle the mouse related events. [5]

(A) 

```java
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
/
/applet code="MouseEvents" width=300 height=100>
</applet>
*/

public class MouseEvents extends Applet implements MouseListener,
MouseMotionListener
{

String msg = " ";
int mouseX = 0, mouseY = 0;
public void init()
{
    addMouseMotionListener(this);
}
//handle mouse dragged
```
public void mouseDragged(MouseEvent me)
{
   // save coordinates
   mouseX = me.getX();
   mouseY = me.getY();
   msg = "*";
   showStatus("Dragging mouse at " + mouseX + ", " + mouseY);
}

// handle mouse moved
public void mouseMoved(MouseEvent me)
{
   // show status
   showStatus("Moving mouse at " + me.getX()+ ", " + me.getY());
}

// Display message in the applet window at current X, Y location
public void paint(Graphics g)
{
   g.drawString(msg, mouseX, mouseY);
}

Q.7(b) Write a Java program using swing components to display tabbed pane with three [5] tabs for First for First Year, Second for Second Year and Third for Third year. Every tab displays a list of all subjects of the respective years.

(A) public class JTabbedExample extends JFrame {

    Public JTabbedExample( ) {
        setTitle("JTabbed pane Example");
        setSize(300, 300);
        JTabbedPane jtp = new JTabbedPane();
        getContentPane() .add(jtp);
        JPanel p1 = new JPanel();
        JPanel p2 = new JPanel();
        JPanel p3 = new JPanel();
        JLabel b1 = new JLabel();
        b1.setText("C++", "ECT", "FIT", "Maths");
        p1.add(b1);
        JLabel b2 = new JLabel();
        b2.setText("MOS", "ASQL", "JDS", "OOPS", "MM");
        p2.add(b2);
        JLabel b3 = new JLabel();
        b3 . setText("NS", "Advjava", "dotNet", "ST", "IT", "PM");
        p3.add(b3);
        jtp.addTab("Fy", p1); jtp.addTab("Sy", p2)
        jtp.addTab("Ty", p3);
    }

    Public static void main(string args[])
    {
        JTabbed Example tab = new JTabbedExample( );
    }
}
Q.7(c) What are HttpServletRequest and HttpServletResponse?

(A) HttpServletRequest interface provides methods for extracting HTTP parameters from the query string or the request body depending on the type of request such as GET or POST.

This interface extends ServletRequest interface to provide request information for HTTP Servlets.

HttpServletRequest interface includes:
Support for:
cookies, session tracking
Access to HTTP header information.
It also parses incoming HTTP form data and stores it as Servlet parameters.

HttpServletRequestResponse interface provides an OutStream for returning binary data such as a GIF or JPEG and PrintWriter for returning text output.

HttpServletRequestResponse interface to provide HTTP protocol-specific functionality including response headers and status codes.

HttpServletRequest and HttpServletResponse interfaces give full access to all information about the request and allows controlling the output sent to the client as the response to the request.

Q.7(d) Explain the concept of "Character Quoting Conventions" in JSP.

(A) • Because certain character sequences are used to represent start and stop tags, the developer sometimes needs to escape a character so the JSP engine does not interpret it as part of a special character sequence.

• In a scripting element, if the character '%>' needs to be used, escape the greater than sign with a backslash.

  `<% String message = "This is the %> message" ; %>

• The backslash before the expression acts as an escape character, informing the JSP engine to deliver the expression verbatim instead of evaluating it.

• Character Quoting And Data Conventions:

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;\%</code></td>
<td>Used in template text (static HTML) where you really want <code>&quot;%&quot;</code>.</td>
</tr>
<tr>
<td><code>%\&gt;</code></td>
<td>Used in scripting elements where you really want &quot;%.&quot;</td>
</tr>
<tr>
<td><code>\</code></td>
<td>A single quote in an attribute that uses single quotes. Remember, however, that you can use either single or double quotes, and the other type of quote will then be a regular character.</td>
</tr>
<tr>
<td><code>\&quot;</code></td>
<td>A double quote in an attribute that uses double quotes. Remember, however, that you can use either single or double quotes, and the other type of quote will then be a regular character.</td>
</tr>
<tr>
<td><code>%\&gt;</code></td>
<td>% in an attribute.</td>
</tr>
<tr>
<td><code>&lt;\%</code></td>
<td>`% in an attribute.</td>
</tr>
<tr>
<td><code>\</code></td>
<td>Used as a delimiter.</td>
</tr>
</tbody>
</table>

Q.7(e) Write a session bean code specificaiton that calculate simple interest. Assume the principal, term and rate of interest is entered by the user and the inputs is passed through a servlet.

(A) import java.rmi.RemoteException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
This class contains the implementation for the `calculateSimpleInterest' method exposed by this Bean. It includes empty method bodies for the methods prescribe by the SessionBean interface; these don’t need to do anything in this simple example.

```java
public class InterestBean implements SessionBean {
    public double calculateSimpleInterest(double principle, double rate, double periods) {
        System.out.println ("Someone called `calculateSimpleInterest!");
        return periods * principle * (rate/100);
    }
}
```

Q.7(f) "Action is a heart and soul of the Struts framework". Discuss. [5]

(A) Action is the heart and soul of the Struts framework. It processes input and interacts with other layers of the application.

Roles of Action

- **Perform As a model:**
  - Action performs as a Model by encapsulating the actual work to be done for a given request based on the input parameters.
  - Encapsulation is done using the execute( ) method. The code spec inside this method should only hold the business logic to serve a Request.
    ```java
    public String execute() {
        setWelcomeMessage(MESSAGE + getUserName());
        return "SUCCESS";
    }
    ```

- **Serves as a Data Carrier:**
  - Action serves as a data carrier from Request to the View.
  - Action being the Model component of the framework carries the data around.
  - The data that it requires is held locally which makes it easy to access using JavaBeans properties during the actual execution of the business logic.

- **Helps determine the results:**
  - Action determines the Result that will render the View that will be returned in the request’s response. This is achieved by returning a control string that selects the result that should be rendered.

- **Single Or Multiple Results:**
  - The most basic action performs the required task and always results a single result.
  - An action can also return different results depending on the complexity of the business logic.