Page :
Date: / /

Programming Technology.
Tutorial - 3

Bhoj Raj Joshi:
181308
BECE - Day.
Assignment of Programming Technology By Bhoj Raj Joshi

1. Differentiate application with applet. Write a simple applet program that demonastrates the event handling.

Ans: The main difference betn application and applet is that the applet is a small java program that can be executed by a Java-compatible web browser while the application is a standalone program that can directly run on the machine. An applet is specifically designed to be executed within an 'HTML' document using a java web browser. It is a small program and does not affect the operating system or the hardware. An application is a standalone java program that can be directly run on the machine. It will perform the tasks defined in the program regardless of the computer architecture.

major difference betn Applet and Application is as

(a) Defination: An applet is a small application that performs one specific task that runs within the scope of a dedicated widget engine or a larger program, often as a plug-in. An application is a standalone program that is designed to run on a standalone machine to accomplish a task. This is the main difference between Applet and Application.

Scanned with CamScanner

Page :
Date: / /

(2) Program Type: While an applet is a small program an application is a large program.

(3) Creations: Another difference betn applet and application is that an applet is created by extending java.applet.Applet whereas an application is created by writing the program inside the main method.

(4) Read and Write Operations: Furthermore, an applet cannot read and write files on the local computer. But, an application can perform file reading and writing on the local computer.

(5) Execution: One other difference betn Applet and Application is that an applet is executed by any Java-compatible web browser. Whereas, application can be executed using Java Runtime Environment (JRE). It is the combination of JVM, Java class libraries and other files required to run Java applications.

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class EventApplet extends Applet implements
        ActionListener {
Button b;
Textfield tf;
public void init() {
tf = new Textfield();
```

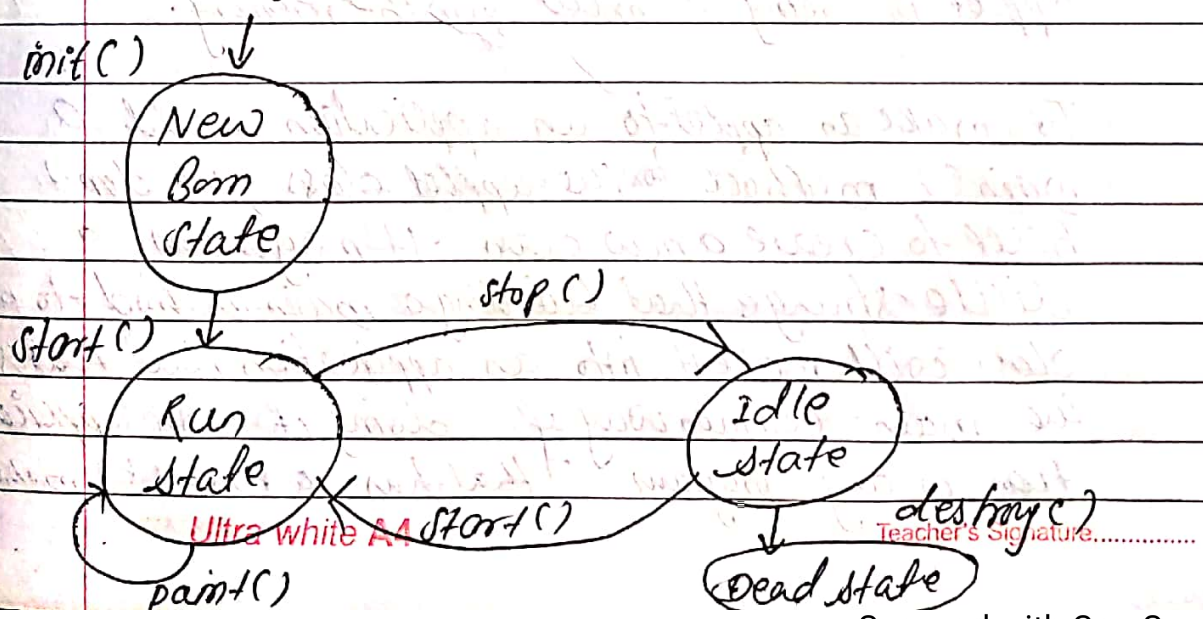

Scanned with CamScanner

Page :
Date: / /

```
tf.setBounds(30,40,150,20);
b= new Button("click");
b.setBounds (80,150,60,50);
add(b);
add(tf);
b.addActionListener(this);
setLayout(null);
}
public void actionPerformed(ActionEvent e){
tf.setText("welcome");
}
}
```

2. What are the five different methods of applet life-cycle? mention the steps involving conversation of applet to application with suitable example program.

Ans: There are five applet methods that are called by the applet container from the time the applet is loaded into the browser to the time it's terminated by the browser. This methods corresponds to various aspects of an applet's life-cycle.



Ultra white A4
Teacher's Signature.....
Scanned with CamScanner

Page :
Date: / /

1. public void init () :
Called once by the applet container when an applet is loaded for execution.

2. public void start ():
Called by the applet container after method init completes execution. In addition, if the user browses to another website and later returns to applet's HTML page, method start is called again.

3. public void paint (Graphics g):
Called by the applet container after method init and start. Method paint is also called when the applet needs to be repainted.

4. public void stop ():
The applet container call this method when the user leaves the applet's web page by browsing to another web page.

5. public void destroy ():
The applet container call this method when the applet is being removed from the memory.

To make an applet to an application, add a main() method to the applet class; we don't have to create a new class. It may seem a little strange that building a main method to a class will make it into an application, but that's the main common way of doing it. An application is any program that has a main () method.

Scanned with CamScanner

Page :
Date: / /

This new main() does what a browser does.

I. Create windows (JFrame) to hold the applet.
II. Make the window's close box stop the applet.
III. Create a new applet object, and add it to the window.
IV. Start the applet by calling init(), then start().
V. Finalize the layout.
VI. Make the window (with the applet init) visible.

Examples

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
public class UserInputDemo extends Applet implements
        ActionListener.
{
int a=0, b=0;
String str1, str2, str;
TextField T1;
TextField T2;
TextField T3;
Button btnAdd;
public void init() {
T1 = new TextField (10);
T2 = new TextField (10);
T3 = new TextField (10);
btnAdd = new Button ("Add");
T1.setText ("0");
T2.setText ("0");
T3.setText ("0");
add (T1);
add (T2);
```


Scanned with CamScanner

Page :
Date: / /

```
add(T3);
add (btnAdd);
btnAdd.addActionListener(this);
}
public void actionPerformed(ActionEvent ae)
{
Str1 = T1.getText();
a = Integer.parseInt(Str1);
Str2 = T2.getText();
b = Integer.parseInt(Str2);
T3.setText("sum =" + (a+b));
}
}
```

3. Why applet tags are needed to run an applet? Explain the different html tags.

Ans:- The applet viewer makes it possible to run a Java applet without using a browser. Applet Viewer ignores any HTML that is not immediately relevant to launching an applet. However, it recognizes a wide variety of applet-launching syntax. The HTML code that Applet Viewer recognizes is useful part of coding.

An 'HTML' tag is a special word or letter surrounded by angle brackets, < and >. We use tags to create HTML elements, such as paragraphs or links. Many elements have an opening tag and a closing tag - for example, a p (paragraph) element has a tag, followed by the paragraph text, followed by a closing tag. Some

Scanned with CamScanner

Page :
Date: / /

elements don't have a closing tag. These are called empty elements. For example, the br elements for inserting line breaks is simply written <br>. If we are working with XHTML then we can write empty elements using self-closing tags. For example, <br/>.

Here we discuss about different HTML tags:

1. ~~CODEBASE~~ <article> : This element is used to define an independent piece of content in a document, that may be a blog, a magazine or a newspaper article.

2. <aside>: It specifies that article is slightly related to the rest of the whole page.

3. <audio>: It is used to play audio file in HTML.

4. <footer>: It defines a footer for a section.
5. <header>: It defines header for a section.
6. <main>: It defines the main content of a document.
7. <mark>: It specifies the marked or highlighted content.
8. <time>: It is used to define a date/time.
9. <video>: It is used to play video file in HTML.
10. <wbr>: It defines the possible line break.

4. How do you pass parameter to the applet? Explain with an example.

Ans:

Scanned with CamScanner

Page :
Date: / /

The Applet tag in HTML allows us to pass parameters to our applet. To retrieve a parameter, we need to use the getParameter() method. It returns the value of the specified parameter in the form of a String object. Thus, for numeric and boolean values, we will need to convert their string representations to their internal formats.

```
import java.awt.*;
import java.applet.*;
public class ParamDemo extends Applet {
String fontName;
int fontSize;
float leading;
boolean active;
public void start() {
String param;
fontName = getParameter("fontName");
if (fontName == null)
fontName = "Not found";
param = getParameter("fontSize");
try {
if (param != null)
fontSize = Integer.parseInt(param);
else
fontSize = 0;
} catch (NumberFormatException e) {
fontSize = -1;
}
param = getParameter("leading");
try {
```


Scanned with CamScanner

Page :
Date: / /

```
if (param != null)
leading = Float.valueOf(param).floatValue();
else
leading = 0;
} catch (NumberFormatException e) {
leading = -1;
}
param = getParameter("accountEnabled");
if (param != null)
active = Boolean.valueOf(param).booleanValue();
}
public void paint (Graphics g) {
g.drawString("Font name :" + fontName, 0, 10);
g.drawString("Font size : " + fontSize, 0, 26);
g.drawString("Leading :" + leading, 0, 42);
g.drawString("Account Active :" + active, 0, 58);
}
}
```

5. What is event handling? Can we handle event without delegation event model? Explain your understanding.

Ans: Event handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism has the code which is known as event handles that is executed when an event occurs.

Java uses the Delegation Event model to handle the events. This model defines the standard mechanism to generate and handle the events. It's concept is quite simple: a source generates an event and sends it to one or more listener. In this scheme, the listener simply waits until it receives an

Scanned with CamScanner

Page :
Date: / /

event. once an event is received, the listeners processes the event and then returns. The advantage of this design is that the application logic that processes events is clearly seperated from the user interface logic that generates those events. A user interface elements is able to "delegate" the processing of an event to a seperate a piece of code. In the delegation event model, listener must registes 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.

6. same as 5

7. Differentiate event source and event Listener? Describe different event classes.

Ans: The main difference betn event source and event Listener is : The event Listener is an object that is notified by the event source when an event occurs. The event Listener receives an event object when it is notified of the event, then uses the object to respond the event. The event source is required to provide methods that enable listeners to be registered and unregistered. The event source also is required to maintain a list of its registered listeners and be able to notify its listeners when an event occurs.

Also, Listeners are created by implementing one or more of the interfaces defined by the java.

Scanned with CamScanner

Page :
Date: / /

awt. event package. When an event occurs, the event source invokes the appropriate method defined by the listener and provides an event object as its arguments.

The classes that represent events are at the core of Java's event handling mechanism. Following are several commonly used event classes and provides a brief description of when they are generated.

1. ActionEvent: Generated when a button is pressed, a list item is double-clicked, or a menu item is selected.

2. AdjustmentEvent: Generated when a scrollbar is manipulated.
3. ComponentEvent: Generated when a component is hidden, moved, resized, or becomes visible.
4. ContainerEvent: Generated when a component is added to or removed from a container.
5. FocusEvent: Generated when a component gains or loses keyboard focus.
6. InputEvent: Abstract superclass for all components input event classes.
7. ItemEvent: Generated when a check box or list item is clicked; also occurs when a choice selection is made or a checkable menu item is selected or deselected.
8. KeyEvent: Generated when i/p is received from the keyboard.
9. MouseEvent: Generated when the mouse is dragged, moved, clicked, pressed or released; also generated when the mouse enters or exits a component.

Scanned with CamScanner

Page :
Date: / /

10. mouseWheelEvent: Generated when the mouse wheel is moved.
11. TextEvent: Generated when the value of a text area or text field is changed.
12. WindowEvent: Generated when a window is activated, closed, deactivated, deiconified, iconified, opened or quit.

8. How ActionEvent is generated? Explain JButton event handling with example.

Ans: An actionEvent is generated when a button is pressed, a list item is double-clicked, or a menu item is selected. ActionEvent class contains the following methods:

String getActionCommand() - used to obtain the command name name for the invoking ActionEvent object.

- int getModifiers() - This method returns an integer that indicates which modifies keys (ALT, CTRL, META, and/or SHIFT) were pressed.
- long getWhen() - Returns the time when the event was generated.

For example, when a button is pressed, an action event is generated that has a command name equal to the label on that button.

Following examples demonstrate the use of JButton in which whenever we click one of the buttons, the appropriate action listener changes the background color of the panel.

Teacher's Signature.....

Ultra white A4

Scanned with CamScanner

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class ButtonTest
{
public static void main (String[] args)
{
ButtonFrame = new ButtonFrame();
frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
frame.show();
}
}
class ButtonFrame extends JFrame
{
public ButtonFrame ()
{
setTitle ("ButtonTest");
setSize (WIDTH, HEIGHT);
ButtonPanel panel = new ButtonPanel();
Container contentPane = getContentPane();
ContentPane.add (panel);
}
public static final int WIDTH = 300;
public static final int HEIGHT = 200;
}
class ButtonPanel extends JPanel
{
public ButtonPanel()
{
JButton yellowButton = new JButton ("Yellow");
JButton blueButton = new JButton ("Blue");
```

Ultra white A4
Teacher's Signature.....
Scanned with CamScanner

Page :
Date: / /

```
JButton redButton = new JButton ("Red");
add (yellowButton);
add (blueButton);
add (redButton);
ColorAction yellowAction = new ColorAction (Color.yellow);
ColorAction blueAction = new ColorAction (Color.blue);
ColorAction redAction = new ColorAction (Color.red);
yellowButton.addActionListener (yellowAction);
blueButton.addActionListener (blueAction);
redButton.addActionListener (redAction);
}
private class ColorAction implements ActionListener
{
public ColorAction (Color c)
{
backgroundColor = c;
}
public void actionPerformed (ActionEvent event)
{
setBackground (backgroundColor);
repaint ();
}
private Color backgroundColor;
}
}
```

9. How is event generated when you click mouse? Explain mouse event handling mechanism with suitable programming code in Java.

Scanned with CamScanner

Page :
Date: / /

Ans: When mouse is clicked, mouse generates an event called mouse event represented by MouseEventClass. MouseEvent is handled by MouseListener. Whenever a GUI Components is interacted by the user, it generates an event. For example, if the user clicks over a button, the button generates an event known as ActionEvent. Similarly, when the user selects a check box, it generates an event and the event is known as ItemEvent. These two examples refer software components. Java also supports events generated by hardware components like mouse and keyboard etc. To capture mouseEvent, Java gives the programmer two interfaces known as MouseListener and MouseMotionListener.

Handling mouse event using the delegation event model is actually quite easy. Just follow these two steps

1. Implement the appropriate interface in the listener so that it will receive the type of event desired.
2. Implement code to register and unregister (if necessary) the listener as a receipient for the event notifications.

```
//Demonstrate the mouse event handling.
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
public class MouseEvents extends Applet implements
    MouseListener, MouseMotionListener {
String msg = " ";
int mouseX = 0, mouseY = 0
public void init () {
addMouseListener (this);
```

Ultra white A4

Teacher's Signature.....

Scanned with CamScanner

Page :
Date: / /

```
addmousemotionListener (this);
}
public void mouseClicked (mouseEvent me) {
mouseX=0;
mouseY=10;
msg = "mouse clicked";
repaint ();
}
public void mouseEntered (mouseEvent me) {
mouseX=0;
mouseY=10;
msg = "mouse entered.";
repaint ();
}
public void mouseExited (mouseEvent me) {
mouseX=0;
mouseY=10;
msg = "mouse exited";
repaint ();
}
public void ~~button~~ mousePressed (mouseEvent me){
mouseX= me.getX();
mouseY= me.getY();
msg = "Down";
repaint ();
}
public void mouseReleased (MouseEvent me) {
mouseX = me.getX ();
mouseY= me.getY ();
msg = "up";
repaint ();
}
```

Ultra white A4

Teacher's Signature.....

Scanned with CamScanner

```
public void mouseDragged(MouseEvent me) {
mousex = me.getx();
mousey = me.gety();
msg = "*";
showStatus("Dragging mouse at " + mousex + ", " +
mousey);
repaint();
}
public void mousemoved(MouseEvent me) {
showstatus("moving mouse at " + me.getx() + ", " +
me.gety());
}
public void paint(Graphics g) {
g.drawstring(msg, mousex, mousey);
}
}
```

10. What are the sources of an event? Describe different event listener interface of java.awt.event packages.

Ans: Following are some of the uses interface components that can generates the events.

1. Button: Generates action events when the button is pressed.
2. Check box: Generates item events when the check box is selected or deselected.
3. Choice: Generates item events when the choice is changed.
4. List: Generates action events when a menu item is selected; generates item events when an ~~checkable menu~~ item is selected or deselected.

Scanned with CamScanner

Page :
Date: / /

5. MenuItem: Generates action events when a menu item is selected; generates item events when a checkable menu item is selected or deselected.
6. Scroll bar: Generates adjustment events when the scroll bar is manipulated.
7. Text Components: Generates text events when the user enters a character.
8. Window: Generates a window events when a window is activated, closed, deactivated, deiconified, iconified, opened or quit.

Listeners are created by implementing one or more of the interfaces defined by the java.awt.event package. They are discussed below:

1. ActionListener: Defines one method to receive an action events.
2. AdjustmentListener: Defines one method to receive adjustment events.
3. ComponentListener: Defines four methods to recognize when a component is hidden, moved, resized, or shown.
4. ContainerListener: Defines two methods to recognize when a component is added to or removed from a container.
5. FocusListener: Defines two methods to recognize when a components gains or loses keyboard focus.
6. ItemListener: Defines one method to recognize when the state of an item changes.

Scanned with CamScanner

Page :
Date: / /

7. KeyListener: Defines three methods to recognize when a key is pressed, released or typed.
8. MouseListener: Defines five methods to recognize when the mouse is clicked, enter a components, exists a Components, is pressed, or is released.
9. MouseMotionListener: Defines two methods to recognize when the mouse is dragged or moved.
10. MouseWheelListener: Defines one method to recognize when the mouse wheel is moved.
11. TextListener: Defines one method to recognize when a text value changes.
12. WindowFocusListener: Defines two methods to recognize when a window gains or loses i/p focus.
13. WindowListener: Defines seven methods to recognize when a window is activated, closed, deactivated, deiconified, iconified, opened or quit.

11. How swing is differs from AWT? What are the advantages of swing?

Ans:

Swing	AWT
1. Java swing components are platform independent.	1. AWT components are platform dependent.
2. swing components are lightweight.	2. AWT components are heavyweight.
3. swing supports pluggable look and feel.	3. AWT doesn't support pluggable look and feel.
4. swing provides more powerful components such	4. AWT provides less components than swing.

Ultra white A4

Teacher's Signature.....

Scanned with CamScanner

	as tables, lists, scrollpanes, colorchoosers, tabbedpane etc.	5. AWT doesn't follow MVC where model represent data, views represents presentation and controller acts as an interface betn model and view.
5.	Swing follows MVC.	

Advantages of swings are as

1. Light weight - swing components are independent of native operating system's API as swing API controls are rendered mostly using pure JAVA code instead of underlying operating system calls.

2. Rich controls: swing provides a rich set of advanced controls like Tree, TabbedPane, Sliders, Colorpickers, table controls.

3. Highly customizable: swing controls can be customized in very easy way as visual appearance is independent of internal representation.

4. Pluggable look and feel: SWING based GUI Application look and feel can be changed at run time based on available values.

12. Explain the mechanism of seperating application code with GUI in swing?

Ans: swing API is set of extensible GUI components to ease developer's life to create JAVA based Front/end/GUI Applications. It is built on top of AWT API and acts as replacement of AWT API as it has almost every controls corresponding to

Scanned with CamScanner

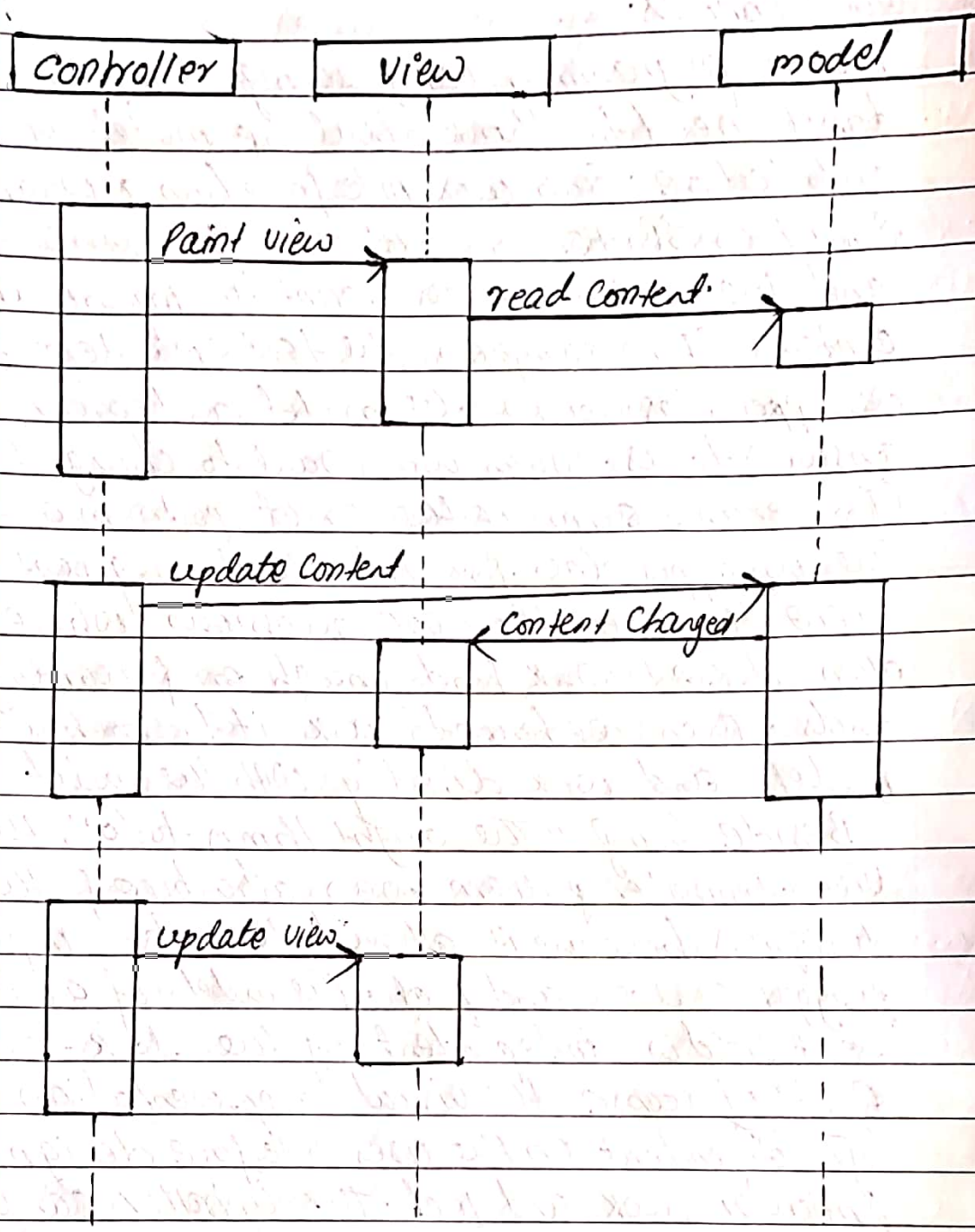
Page :
Date: / /

AWT Controls.

As a programmer using swing components, we generally don't need to think about the model-view controller architecture. Each uses interface has a wrapper class (such as JButton or JTextField) that stores the model and the view. When we want to inquire about the contents (for example, the text in a text field), the wrapper class asks the model and returns the answer to us. When we want to change the view (for example, move the caret posn in a text field), the wrapper class forwards that request to the view. However, there are occasions where the wrapper class doesn't work hard enough on forwarding the commands. Then we have to ask it to retrieve the model and work directly with the model.

Beside being "The right things to do", the model-view-controller pattern was attractive for the swing designers because it allowed them to implement pluggable look and feel. The model of a button or text field is independent of the look-and-feel. But of course the visual representation is completely dependent on the uses interface design of a particular look and feel. The controller can vary as well.

Scanned with CamScanner



13. What is Java servlet? How do you differentiate java applet, java servlet and java serves pages?

Ans:- A Java servlet is a Java software components that extends the capabilities of a serves. Although servlets can corresponds to many types of requests, they most commonly implements web containers

Scanned with CamScanner

Page :
Date: / /

for hosting web applications on web servers and thus qualify as a server-side servlet web API.

web applications built with Java includes Java applets, Java servlets and Java server pages (JSP). Java applets run on the client computer. Javascript, which is a different language than Java despite its similar names, also runs on the client computer as part of the web browser. Java servlets and Java server pages (JSP) run on the server. Java server pages (JSP) which are a dynamic version of Java servlets. servlets must be compiled before they can run, just like a normal Java program. In contrast, JSP code is embedded with the corresponding HTML and is compiled "on the fly" into a servlet when the page is requested. This flexibility can make it easier to develop web applications using JSP than with Java servlets.

24. Why servlet is important in Java? What are the advantages of using servlet in your application?

Ans: servlet is important in Java because of the following reason:

→ The performance is much better.

→ servlet is used when we do not like to create a seperate process to handle each and every request of client.

→ Developers can use servlets when they have to take advantages of all the features of Java.

→ Java class libraries which are available to a servlet can communicate with applets, databases,

Scanned with CamScanner

Page :
Date: / /

or other software via RMI and socket mechanism.

Advantages of using servlet over application is as

1. Servlets performance is significantly better. It execute within the address space of a web server. It is not necessary to create a seperate process to handle each client request.

2. Servlets are platform-independent because they are written in Java.

3. The Java security ~~images~~ manager on the server enforces a set of instructions to protect the resources on a server machine.

4. The full functionality of the Java class libraries is available to a servlet. It can communicate with applet, database, or other softwares via RMI and socket mechanism.

15. How multiple clients are handled in Java servlets? Describe its life cycle.

Ans: multiple clients are handled in Java servlets:

(a) The servlet container creates one object of that servlet class.

(b) Servlet container starts 4 threads on that object representing 4 requests on per request basis.

(c) Servlet Container creates 4 sets of requests

Scanned with CamScanner

response objects on one set per each request. Every thread can be identified through its "thread name." Every object can be identified with its unique number called "Hashcode". To get the Hashcode of any object call Hashcode() of an that object.

Servlet Life cycle:

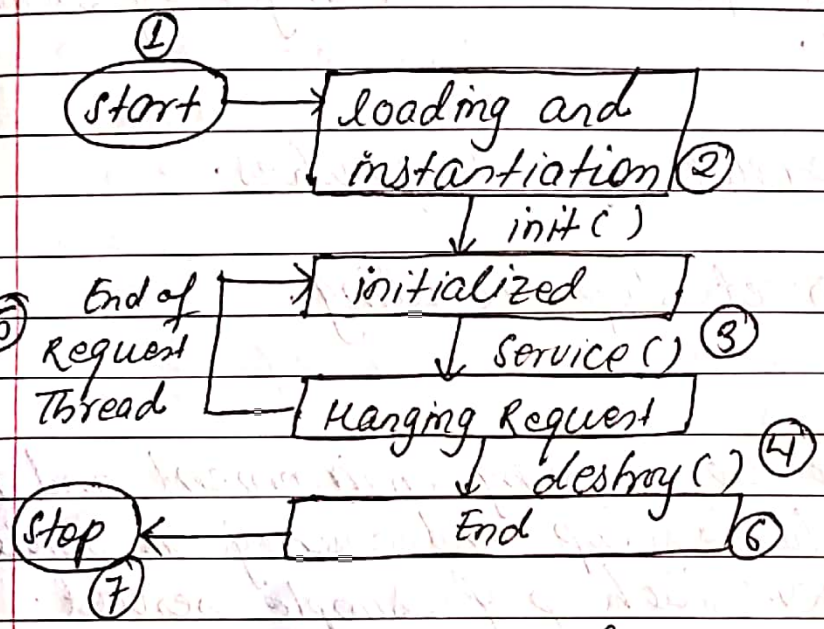


Fig: Block diagram of servlet Life cycle

1. Start: Execution of servlet begins.
2. Loading and instantiation void init(): It is called when servlet is first loaded. This methods lets you initialized servlet.

3. Initialized void service(): The purpose of this method is to serve a request we can call it as many times as we like.

4. Handling request and destroying servlet:

Scanned with CamScanner

Page :
Date: / /

Java applications must be 1st determined what code is needed to execute the request URL to provide a response. To destroy servlet void destroy method is used at the end of servlet life cycle.

5. End of Request Thread: when service() finishes its task, either the thread ends or returns to the thread pool that is manged by servlet container.

6. End: servlet lifecycle finishes.

7. Stop: servlet stop executing.

16. What are the necessary environment and interface required to develop and run java servlet. Explain with the help of simple servlet.

Ans:- To become familiar with the key servlet concepts, we will begin by building and testing a simple servlet. The basic steps are the following:

1. Create and compile the servlet source code. Then, copy the servlet's class file to the proper directory, and add the servlet's name and mappings to the proper web.xml file.
2. Start Tomcat.
3. Start a web browser and request the servlet. Let us examine each of these steps in details:

Scanned with CamScanner

Page :
Date: / /

Create and compile the servlet source code :-
To begin, create a file named HelloServlet.java that contains the following program

```
import java.io.*;
import javax.servlet.*;
public class HelloServlet extends GenericServlet {
public void service (ServletRequest request, ServletRes-
ponse response) throws ServletException, IOException {
response.setContentType ("text/html");
PrintWriter pw = response.getWriter();
pw.println ("<B> Hello!");
pw.close();
  }
}
```

Let's look closely at this program. First note, that it imports the javax.servlet package. Next, the program defines HelloServlet as a subclass of GenericServlet. The GenericServlet class provides functionality that simplifies the creation of a servlet. For example, it provides version of init() and destroy(), which may be used as is. We need supply only the service() method. Inside HelloServlet, the service() method is overriden. This method handles request from a client. Note that the first argument is a ServletRequest object. This enables the servlet to read data that is provided via the client request. The second argument is a ServletResponse object. This enables the servlet to formulate a response for the client.
To call to setContentType() establishes the MIME type of the HTTP response. In this program, the

Teacher's Signature.....

Ultra white A4

Scanned with CamScanner

Page :
Date: / /

MIME type is text/html. This indicates that the browser should interpret the content as HTML source code. Next, the getWriter() method obtains a printWriter. Anything written to this stream is sent to the client as part of the HTTP response. Then println() is used to write some simple HTML source code as part of the HTTP response.

Compile this source code and place the HelloServlet.class file in the proper Tomcat directory. Also add HelloServlet to the web.xml file

Start Tomcat:
Tomcat must be running before you try to execute a servlet.

Start a Web Browser and Request the servlet:
Start a web browser and enter the URL shown
http://localhost:8080/servlets-examples/servlet/HelloServlet

17. Why do we need servlet, ServletRequest and ServletResponse interfaces while developing java servlet? List the methods defined by each of the above interfaces.

Ans: All servlets must implements the servlet interface.
The servletRequest interface enables a servlet to obtain information about a client request.
The servletResponse interface enables a servlet

Scanned with CamScanner

Page :
Date: / /

to formulate a response for a client.

The Servlet Interface method:
It declares the init(), service(), and destroy() methods that are called by the server during the life cycle of a servlet. A method is also provide that allows a servlet to obtain any initialization parameters. The getServletConfig() method is called by the servlet to obtain initialization parameters. A servlet developers overrides the getServletInfo() method to provide a string with useful information. The server invokes this method also.

The ServletRequest Interface:

1. Object getAttribute(String attr): Returns the value of the attribute named attr.

2. String getCharacterEncoding(): Returns the character encoding of the request.

3. int getContentType(): Returns the size of the request. The value -1 is returned if the size is unavailable.

4. String getContentType(): Returns the type of the request. A null value is returned if the type cannot be determined.

5. ServletInputStream getInputStream() throws IOException: Returns a ServletInputStream that can be used to read binary data from the request. An IllegalSt-

Ultra white A4
Teacher's Signature.....

Scanned with CamScanner

Page :
Date: / /

ateException is thrown if getReader() has already been invoked for this request.

6. String getParameter(String pname): Returns the value of parameter named pname.

7. Enumeration getParameterNames(): Returns an enumeration of the parameter names for this request.

8. String[] getParameterValues(String name): Returns an array containing values associated with the parameter specified by name.

9. String getProtocol(): Returns a description of the protocol.

10. BufferedReader getReader() throws IOException: Returns a buffered reader that can be used to read text from the request. An IllegalStateException is thrown if getInputStream() has already been invoked for this request.

11. String getRemoteAddr(): Returns the string equivalent of the client IP address.

12. String getRemoteHost(): Returns the string equivalent of the client host name.

13. String getSchema(): Returns the transmission scheme of the URL used for the request.

Ultra white A4

Teacher's Signature.....

Scanned with CamScanner

Page :
Date: / /

For example, "http", "ftp".

14. String getServerName(): Returns the name of the server.

15. int getServerPort(): Returns the port number.

The ServletResponse Interface:

1. String getCharacterEncoding(): Returns the character encoding for the response.

2. ServletOutputStream getOutputStream() throws IOException: Returns a ServletOutputStream that can be used to write binary data to the response. An IllegalStateException is thrown if getWriter() has already been invoked for this request.

3. PrintWriter getWriter() throws IOException: Returns a PrintWriter that can be used to write character data to the response. An IllegalStateException is thrown if getOutputStream() has already been invoked for this request.

4. void setContentLength(int size): sets the content length for the response to size.

5. void setContentType(String type): sets the content type for the response to type.

Scanned with CamScanner

Page :
Date: / /

18. Write a java program to demonstrate the way to read parameters from the servlet.

Ans:-
```
import java.io.*;
import java.util.*;
import javax.servlet.*;
public class PostParametersServlet extends GenericServlet{
public void service(ServletRequest request, ServletRe-
ponse, response)
throws ServletException, IOException {
PrintWriter pw = response.getWriter();
Enumeration e = request.getParameterNames();
while (e.hasMoreElements()) {
String pname = (String) e.nextElement();
pw.print(pname + "=");
String pvalue = request.getParameter(pname);
pw.println(pvalue);
}
pw.close();
}
}
```

19. What is filter in servlet? write a simple servlet name as AddServlet with the url pattern /addservlet which gets the two request parameter (num1 and Num2) from jsp page and add those values and redirect along with summed value and send response to the client.

Ans:- A filter is an object that is invoked at the preprocessing and postprocessing of a request. It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and

Scanned with CamScanner

Page :
Date: / /

decryption, input validation etc. The servlet filter is pluggable, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet.

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class AddServlet extends HttpServlet {
public void service (HttpServletRequest req, HttpServletR-
esponse res) throws ServletException, IOException {
int firstNumber = Integer.parseInt (req.getParameter
("number1"));
int secondNumber = Integer.parseInt (req.getParameter
("number2"));
int answer = firstNumber + secondNumber;
req.setAttribute ("value", answer);
RequestDispatcher request = req.getRequestDispatcher
("sqServlet");
request.forward (req, res);
}
}

public class SquareServlet extends HttpServlet {
public void service (HttpServletRequest req, Httpser-
vletResponse res) throws IOException {
```


Scanned with CamScanner

Page :
Date: / /

```
int value = (int) req.getAttribute("value");
int finalAnswer = value * value;
res.getWriter().print(finalAnswer);
    }
}
```


Scanned with CamScanner

Page :
Date: / /

20. How do you read and write session information in java servlet? What are the methods defined by HttpSession interface.

Ans: The time interval in which two systems (i.e. the client and the server) communicate with each other can be termed as session. In simpler terms, a session is a state consisting of several requests and response betn the client and the server. It is known fact that HTTP and web servers are both stateless. Hence, the only way to maintain the state of the user is by making use of technologies that implements session tracking. Session tracking in servlets can be implemented by number of methods, cookies being one of them. However they have multiple disadvantages:

- only textual information can be kept by them.
- If cookies are disabled by a user, the web application is unable to make use of them.
- Not more than 40kb data can be contained by a single cookie.
- Another way to implement session tracking is by creating sessions with unique session ids for every user in a Java servlet.

The HTTP session interface enables a servlet to read & write the state information that is associated with an HTTP session.

1. object getAttribute (String Attr): Returns the value associated with the name passed in attr. Returns null if attr is not found.

2. Enumeration getAttributeNames (): Returns an

Ultra white A4

Teacher's Signature.....

Scanned with CamScanner

Page :
Date: / /

enumeration of the attribute names associated with the session.

3. long getCreationTime() : Returns the time (in milliseconds since midnight, Jan 1, 1970, GMT) when the ~~client last made a request for this~~ session was created.

4. String getId() : Returns the session ID.

5. long getLastAccessedTime() : Returns the time (in milliseconds since midnight, Jan 1, 1970, GMT) when the client last made a request for this session.

6. void invalidate() : Invalidates this session and removes it from the context.

7. boolean isNew() : Returns true if the server is created the session and it has not yet been accessed by the client.

8. void removeAttribute (String attr) : Removes the attribute specified by attr from the session.

9. void setAttribute (String attr, object val) : Associate the value passed in val with the attribute name passed in attr.

Scanned with CamScanner

Page :
Date: / /

21. How can you exchange information between different servlets? Explain each with suitable section of code.

Ans: The servletcontext is a shared memory segment for web applications. When an object is placed in the servletcontext, it exists for the life of a web application, unless it is explicitly removed or replaced. There is one servletcontext per web application and all servlet share it. It can be retrieved via getServletContext() method. ServletContext has an "application" scope, and can also be used to pass information betn servlets within the same application, via methods setAttribute("name", object) and getAttribute("name").

Example i.e. source code

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SecondServlet extends HttpServlet {
public void doGet(HttpServletRequest request,
HttpServletResponse response)
try {
response.setContentType("text/html");
PrintWriter out = response.getWriter();
HttpSession session = request.getSession();
String name = (String) session.getAttribute("username");
String email = (String) session.getAttribute("emailid");
out.print("Name = " + name);
out.print("Email = " + email);
out.close();
```

Ultra white A4

Teacher's Signature.....

Scanned with CamScanner

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

22. What are the difference between get and post method? Explain the usage of cookies.

Ans: Difference betn get and post method is as

	GET	POST
1.	In case of get request, only limited amount of data can be sent because data is send in header.	1. In case of post request, large amount of data can be sent because data is sent in body.
2.	Get request is not secured because the data is exposed in URL bar.	2. Post request is secured because data is not exposed in URL bar.
3.	Get request can be bookmarked.	3. Post request cannot be bookmarked.
4.	Get request is more efficient and used more than post.	4. Post request is less efficient and used less than get.

Scanned with CamScanner

Page :
Date: / /

The cookie class encapsulate a cookie. A cookie is stored on a client and contains state information cookies are valuable for tracking user activities. For example, assume that a uses visits an online store. A cookie can save the users name, address, and other information. The user do not need to enter this data each time he or she visits the store. A servlet can write a cookie to a users machine via the addCookie() method of the HttpServletResponse interface. The data for that cookie is then included in the header of the HTTP response that is sent to the browser.
The names and values of cookie are stored on the users machine. Some of the information that is saved for each cookie includes the following
(a) The name of the cookie.
(b) The value of the cookie.
(c) The expiration date of the cookie.
(d) The domain and path of the cookie.

The expiration date determines when this cookie is deleted from the users machine. If an expiration date is not explicitly assigned to a cookie, it is deleted when the current browser session ends. Otherwise, the cookie is saved in a file on the users machine. The domain and path of the cookie determines when it is included in the header of an HTTP request. If the user enters a URL whose domain and path match these values, the cookie is then supplied to the web browser. Otherwise it is not. There is one constructor for cookie.

Scanned with CamScanner

Page :
Date: / /

23. What are the advantages of JSP? Explain different standard Tag Library of JSP.

Ans: Advantages of JSP are as follows:

(a) Vs. Active Server Pages (ASP): The advantages of JSP are two fold. First, the dynamic part is written in java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-microsoft web servers.

(b) Vs. Pure servlets: It is more convenient to write (and modify!) regular HTML than to have plenty of println statments that generates the HTML.

(c) Vs. Server-Side Includes (SSI): SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.

(d) Vs. Javascript: Javascript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

(e) Vs. Static HTML: Regular HTML, of course, cannot contain dynamic information.

Scanned with CamScanner

Page :
Date: / /

The Java server Pages Standard Tag Library (JSTL) is a collection of useful JSP tags which encapsulates the core functionality common to many JSP applications. JSTL has supports for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags.

1. <c:out> : Like <%= ..... >, but for expressions.
2. <c:set> : sets the result of an expression evaluation in a 'scope'.
3. <c:remove> : Removes a scoped variable.
4. <c:catch> : Catches any Throwable that occurs in its body and optionally exposed it.
5. <c:if> : Simple conditional tag which evaluates its body if the supplied condition is true.
6. <c:choose>: Simple Conditional Tag that establishes a context for mutually exclusive conditional operations, marked by <when> and <otherwise>
7. <c:when> : Subtag of <choose> that includes its body if its ~~has~~ condition evaluates to true.
8. <c:otherwise>: Subtag of <choose> that follows the <when> tags and runs only if all of the prior conditions evaluated to 'false'.
9. <c:forTokens>: Iterates over tokens, seperated by the supplied delimeters.
10. <c:param>: Adds a parameter to a containing 'import' tags URL.
11. <c:redirect>: Redirects to new URL.
12. <c:url>: Creates a URL with optional query parameters.

Scanned with CamScanner

Page :
Date: / /

~~24~~ 25. Identify the following JSP Tags : <%%>, <%@%>, <%!%>, <%=%>. Also explain them.

Ans: 1. <%%> : scriptlet
Blocks of Java code can be embeded in a scriptlet. The syntax for a scriptlet is as follows: <%
Java code
%>
The following scriptlet invokes the incrementCount() method and then o/ps the value in count:

```
<%
out.println("The counter's value is" + count
  + "<br/>");
incrementCount();
%>
```

2. <%@%>: directives
In General terms, directives instruct the compiler how to process a JSP program. Example includes the definition of our own tags, including the source code of other files, and importing packages.
The syntax is as:

```
<%@
page import = "java.util.*, java.sql.*"
%>.
```

3. <%!%>: element declaration
The declaration tags allows us to define variable and methods. Variable declarations are compiled

Teacher's Signature.....

Ultra white A4

Scanned with CamScanner

Page :
Date: / /

as instance variables for a class that corresponds to the JSP page. Declarations are defined with the syntax

```
<%!
Declarations
%>
```

For example, the following defines an instance variable named count and a method named incrementCount() that increments the count variable.

```
<%!
  private int count = 0;
  private void incrementCount ()
  {
  count ++;
  }
%>
```

4. <%=%> : Expression

We can access variables defined in declarations with expression. The syntax to embed an expression is as follows:

```
<%=
Expression
%>
```

Expressions are embeded directly into the HTML. The web browser will display the value of the expression in place of the tag. For example, we can output the value of the count variable in bold type with the following piece of HTML:

The value of count is <%=count%>.

Page :
Date: / /

26. Explain the mechanism of conversion from JSP to Servlet with reference to life cycle method of Servlet and JSP.

Ans: Each JSP page is internally translated into an equivalent Servlet. Every JSP page or tags written in the page will be internally converted into an equivalent java code by the container and this process is called translation.

There are two phases in JSP:

(a) Translate phase.

(b) Request processing Phase.

(a) Translate phase: It is processing of converting JSP into an equivalent servlet and then generating class file of the servlet.

(b) Request Processing Phase: It is process of executing service() of jsp and generating response data on to the browser.

When first time a request to a jsp then translation phase occurs first and then after service phase will be executed. From next request to the jsp, only request processing phase is got executed but translation phase is not because jsp is already translated.

In following two cases only translation occurs;

(a) When 1st request is a given to the jsp.

(b) when jsp is modified.

If a request is given to the jsp after it has modified then again translation phase is executed first and after that request processing phase will

Ultra white A4

Teacher's Signature.....

Scanned with CamScanner

Page :
Date: / /

be executed.

notes:

a) If we shutdown and restart the servlet then only request processing phase will be executed, Because when we shutdown the server java and class files are not deleted from the server.

b) If we remove the class file from server then partial translation occurs.

c) If we remove both java and class file from server then again translation occurs.

d) When a jsp is translated into an equivalent servlet then that servlet extends some base class provided by the server, but internally it extends HttpServlet.

27. Why do we need to handle parameters in JSP? Explain with suitable examples.

Ans: Here we are handling a JSP page (file.jsp) to the main JSP page (index.jsp) using <jsp:include> action tag. To pass the parameters from the index to the file page we are using <jsp:param> action tag. We are passing three parameters firstname, middlename and lastname, for each of these parameters we need to specify the corresponding parameter name and parameter value. The same we can extract at the included page using the expression language or expression tag and implict object.

index.jsp
<html>

Scanned with CamScanner

Page :
Date: / /

```
<head>
<title>JSP include with parameters example</title>
</head>
<body>
<jsp:include page="field.jsp">
<jsp:param name="firstname" value="Bhej" />
<jsp:param name="middlename" value="Raj" />
<jsp:param name="lastname" value="Jashi" />
</jsp:include>
</body>
</html>
```

file.jsp

```
${param.firstname}<br>
${param.middlename}<br>
${param.lastname}
```

O/P

Bhej
Raj
Jashi

As we can see that the include page displayed the parameters value passed from the main page. For displaying the parameters on the file.jsp page, we have used the expression language (${}).

SHOT ON MI A2
MI DUAL CAMERA

Ultra white A4

Teacher's Signature.....

Scanned with CamScanner

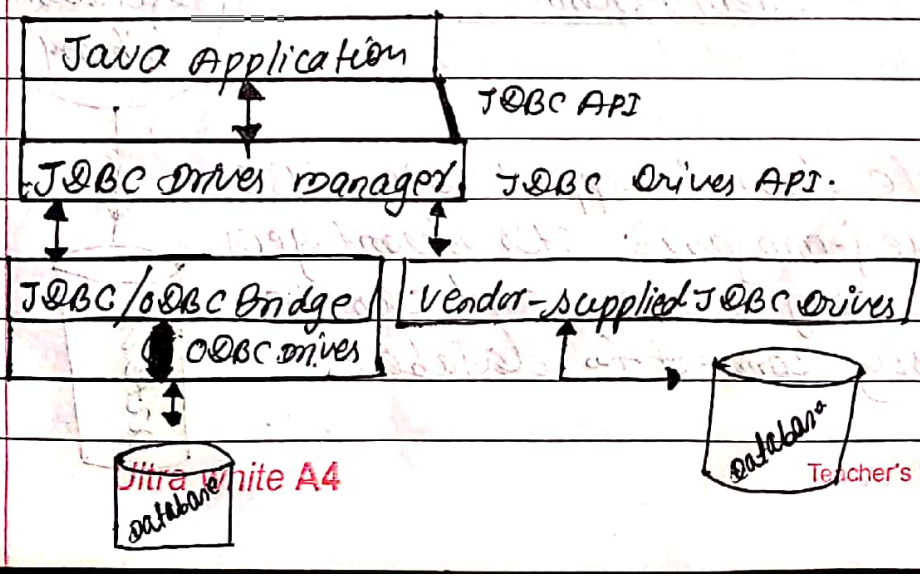
Page :
Date: / /

28. What is JDBC? Explain how the Java program access the database.

Ans: Java database connectivity (JDBC) is an application programming interface (API) for the programming language java, which defines how a client may access a database. It is a java-based data access technology used for Java database connectivity.

Java programs communicates with databases and manipulate their data using the Java Database Connectivity (JDBC) API. A JDBC driver enables Java applications to connect to a database in a particular DBMS and allows us to manipulate the database using the JDBC API. Most popular database management System now provide JDBC drivers. There are also many third-party JDBC drivers available.

The JDBC consists of two layers. The top layer is the JDBC API. The API communicates with the JDBC manager driver API, sending it the various SQL statements. The manager should communicate with the various third-party drivers that actually connect to the database and return the information from the query or perform the action specified by the query.



Ultra White A4
Teacher's Signature.....
Scanned with CamScanner

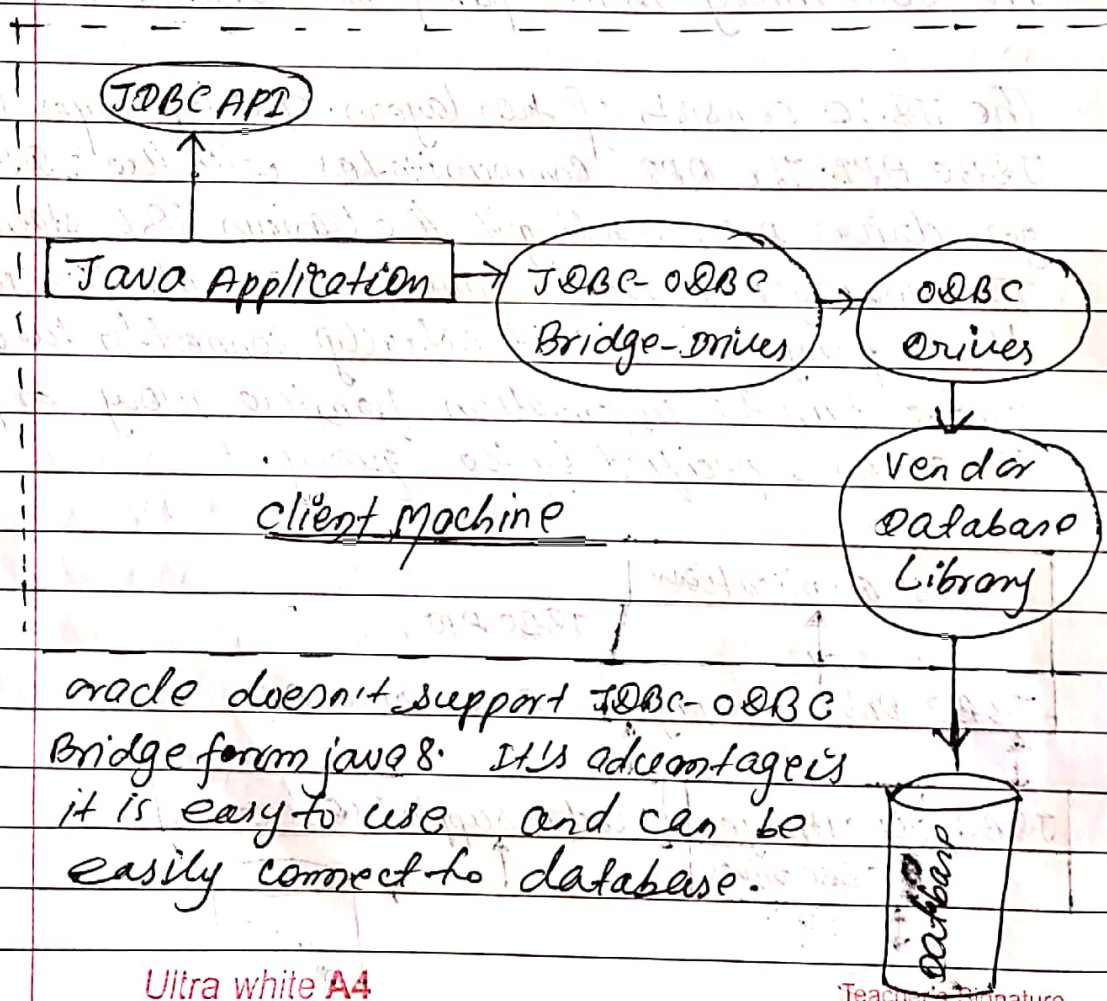
Page :
Date: / /

29. What are the types of JDBC drivers? Explain the use of JDBC drivers.

Ans: JDBC drives implementation vary because of the wide variety of operating systems and hardware platforms in which java operates. Sun has divided the implementation types into four categories, Type 1, 2, 3, and 4 which is explained below

Type 1: JDBC-ODBC Bridge Driver

The JDBC-ODBC bridge drives uses ODBC driver to connect to the database. The JDBC-ODBC bridge drives converts JDBC method calls into the ODBC function calls. This is now discouraged because of the 100% pure java drives.

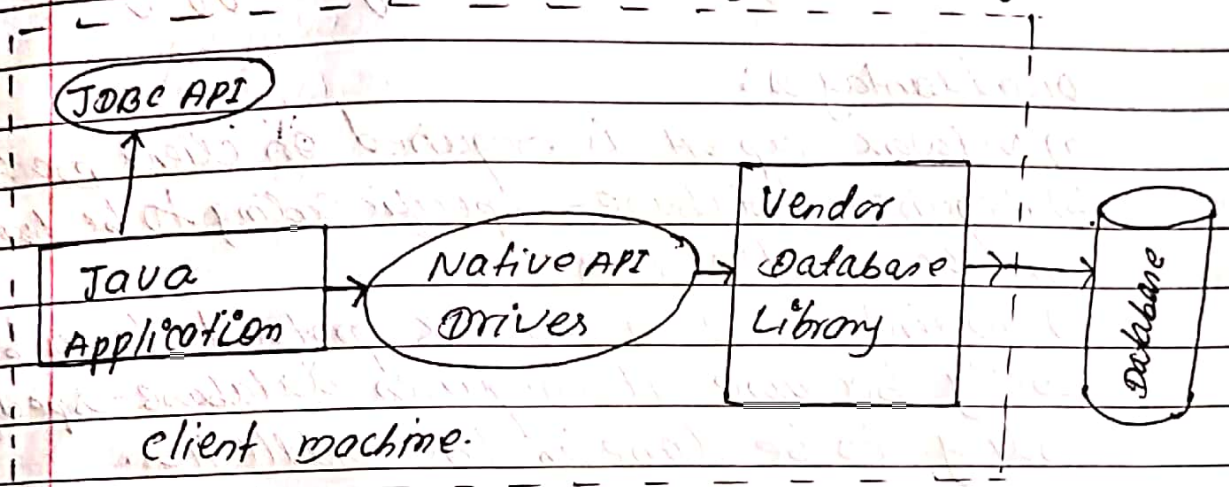


oracle doesn't support JDBC-ODBC Bridge form java 8. Its advantage is it is easy to use and can be easily connect to database.

Scanned with CamScanner

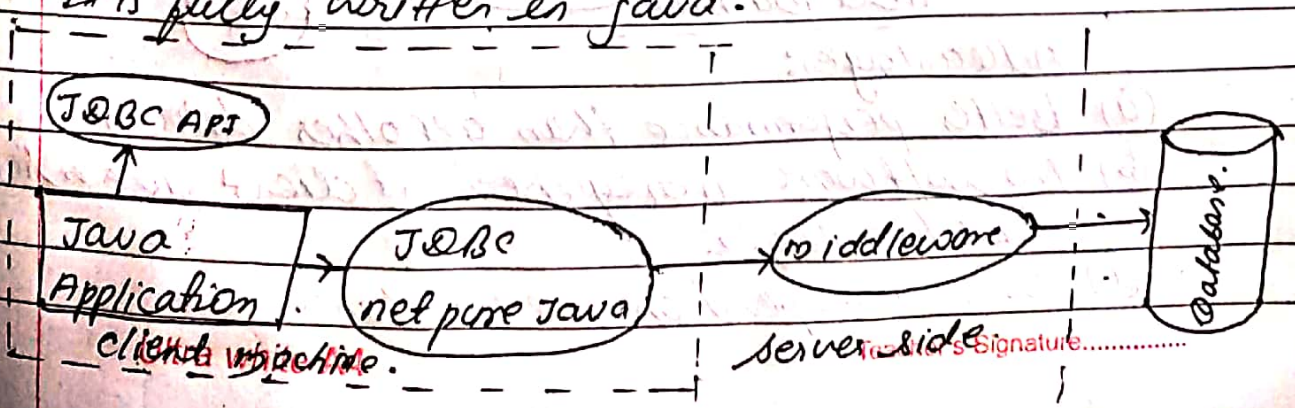
Page :
Date: / /

Type2: JDBC Native API: The native API drives uses the client side libraries of the database. The driver converts JDBC methods calls into native calls of the database API. It is not entirely written in java.



Advantage:
. Performace upgraded than JDBC-ODBC bridge driver.
Disadvantages:
• The Native drivers needs to be installed on the each client machine.
• The vendor client library needs to be installed on client machine.

Type3: JDBC-Net pure Java: The Network protocol driver i.e. JDBC-Net pure Java uses middleware (applicate server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol. It is fully written in java.



Teacher's Signature.....

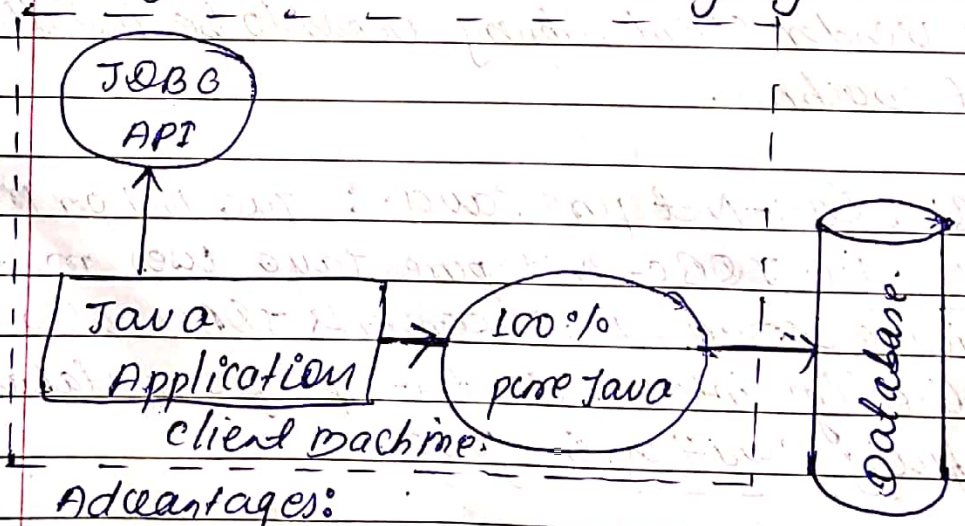
Scanned with CamScanner

Page :
Date: / /

Advantage:
a) No client side library is required because of application servers that can perform many tasks like auditing, load balancing, logging etc.

Disadvantages:
a) Network support is required on client machine.
b) Requires database-specific coding to be done in the middle tier.
c) Maintenance of Network protocol drivers becomes costly because it requires database-specific coding to be done in the middle tier.

Type 4: 100% pure Java (Thin driver):-
The 100% pure Java converts JDBC calls directly into the vendor-specific database protocol. That is why it is known as 100% pure Java. It is fully written in Java language.



Advantages:
(a) Better performance than all other drivers.
(b) No software is required at client side or server side.
Disadvantages:
(a) Drivers depend on the Database.

Scanned with CamScanner

Page :
Date: / /

32. What are the difference between JDBC and ODBC.

Ans: Here we go JDBC Vs ODBC.

	JDBC	ODBC
a.	JDBC is object oriented.	a) ODBC is procedural oriented.
b.	JDBC is used to provide connection betn 'JAVA' & database (oracle, Sybase, DB2, ms-access.	b) ODBC is used to provide connection betn front-end application (other than java) and backend (database like ms-access)
c.	JDBC is for java applications.	c. ODBC is for microsoft.
d.	JDBC is used by java programmers to connect to database.	d. ODBC is used betn application of different types.
e.	JDBC allows SQL-based database access for EJB persistence and for direct manipulation from CORBA, DJB or other server objects.	e. With small "bridge" program, we can use the JDBC interface to access ODBC accessible database.
f.	JDBC is multi-threaded.	f. ODBC is not multi-threaded.
g.	JDBC is faster.	g. ODBC is slower.
h.	JDBC API is natural Java interface and is built on ODBC, & therefore JDBC retains some of the basic feature of ODBC.	h. ODBC makes use of pointers which have been removed totally from java.

Scanned with CamScanner