SRI VENKATESA PERUMAL
COLLEGE OF ENGINEERING & TECHNOLOGY
| AUTONOMOUS | ACCREDITED BY NAAC |
RVS Nagar, K.N Road, Puttur, Chittoor dist, AP. I www.svpcet.org

Department of Electronics&Computer Science and Engineering

Oops Using Java Learning Question Bank (20OE0501)

Regulation : R20 **Academic Year : 2022-23**
Semester : V SEM **Subject Code : 20OE0501**

UNIT – 1 : An Overview of Java : Evolution of java,Object Oriented Programming ,Data types,Arrays,Arithmetic Operators
Control Statements: Java's selection Statements,Iteration statements,Jump

SNO	QUESTIONS 2 MARKS	CO	PO	BL
1	Define Object Oriented Programming and java .	CO1	1,2,3	1
2	What are Data types?.	CO1	1,2,3	5
3	What are Arrays?	CO1	1,2,3	5
4	What are Arithmetic Operators?	CO1	1,2,3	1
5	What is Java's selection Statements?.	CO1	1,2,3	4
6	List Iteration statements	CO1	1,2,3	4
7	List Jump Statements.	CO1	1,2,3	2
8	What is Java Buzzwords?	CO1	1,2,3	2
9	How Java Changed the Internet ?	CO1	1,2,3	5
10	What is Java Lineage.	CO1	1,2,3	1

SNO	QUESTIONS 10 MARKS	CO	PO	BL
1	Explain Evolution of java .	CO1	1,2,3	2

2	Explain Object Oriented Programming?	CO1	1,2,3	2
3	Describe Java Data types with an examples?	CO1	1,2,3	3
4	Describe Java Arrays with an example programs?.	CO1	1,2,3	4
5	Describe the types of Arithmetic Operators of Java with examples.	CO1	1,2,3	2
6	Explain Java's selection Statements with an example	CO1	1,2,3	5
7	Explain Iteration statements with an example.	CO1	1,2,3	1
8	Describe Jump statements with an eample.	CO1	1,2,3	4

UNIT - 2: Introducing classes : Class Fundamentals,Declaring Objects,Introducing Methods,Constructors,Argument Passing,Returning Objects,Recursion,Using Object as Parameter,The this Keyword

SNO	QUESTIONS 2 MARKS	CO	PO	BL
1	Define Class	CO2	1,2,3,5	1
2	Define Object .	CO2	1,2,3,5	2
3	What is Method ?	CO2	1,2,3,5	5
4	What is Constructor ?	CO2	1,2,3,5	2
5	What is Argument Passing .	CO2	1,2,3,5	1
6	How Returning Objects.	CO2	1,2,3,5	1
7	What is Recursion .	CO2	1,2,3,5	1
8	What is Parameter .	CO2	1,2,3,5	2
9	What is object as a Parameter ?	CO2	1,2,3,5	3
10	What is this Keyword ?	CO2	1,2,3,5	4

SNO	QUESTIONS 10 MARKS	CO	PO	BL
1	Explain Class Fundamentals with an example?	CO2	1,2,3,5	2
2	Describe how to Declaring Objects and give an example?	CO2	1,2,3,5	6
3	Describe Methods with an example ?	CO2	1,2,3,5	3
4	Describe Java Constructors with an example.	CO2	1,2,3,5	6
5	Explain Argument Passing with an example	CO2	1,2,3,5	4
6.	Explain Returning Objects with an example	CO2	1,2,3,5	5
7	Describe Recursion with an example	CO2	1,2,3,5	3
8	Describe how Using Object as Parameter and this Keyword with an example?	CO2	1,2,3,5	2

Unit - III :

Inheritance:

Inheritance,Understanding Inheritance,The Object Class,Polymorphism,Understanding Polymorphism,Understanding Abstract classes,Understanding Interfaces,Packages,Access protection,Importing Packages

2 Marks Questions :

S.No	Question	CO	PO	BL
1.	What is Inheritance?	CO3	1,2,3,5	1
2.	What is Object ?	CO3	1,2,3,5	2
3.	What is Class ?	CO3	1,2,3,5	2
4.	Define Polymerphism	CO3	1,2,3,5	1
5.	Define Abstract Class	CO3	1,2,3,5	1
6.	Define Interface.	CO3	1,2,3,5	1
7.	What is Access protection ?	CO3	1,2,3,5	1

8.	How can Importing Packages?	CO3	1,2,3,5	2

10 Marks Questions :

S.No	Question	CO	PO	BL
1.	Explain the Inheritance with an example program.	CO3	1,2,3,5	2
2.	Explain Polymorphism with an example program.	CO3	1,2,3,5	1
3.	Explain Abstract classes with an example program	CO3	1,2,3,5	2
4.	Explain Interfaces with an example program.	CO3	1,2,3,5	2
5.	Explain Packages with an example program.	CO3	1,2,3,5	1
6.	Describe Access Protection..	CO3	1,2,3,5	2
7.	How to Importing a Package with an example.	CO3	1,2,3,5	1

Unit - IV

Exception Handling:

Exception handling Fundamentals,Using try and catch,Java Built-in Exceptions,Chained Exceptions

Introducing Stream:I/O basics,Stream,Reading Console input,Lifecycle of an Applet.

2 Marks Questions :

S.No	Question	CO	PO	BL
1.	Define Exception	CO4	1,2,3,5	2
2.	Define try block	CO4	1,2,3,5	2
3.	Define Catch block	CO4	1,2,3,5	3
4.	What is Java-Built in Exception ?	CO4	1,2,3,5	1
5.	Define Chained Exception.	CO4	1,2,3,5	2
6.	Define Stream	CO4	1,2,3,5	2

7.	Define Applet.	CO4	1,2,3,5	2
8.	Define Input system.	CO4	1,2,3,5	2
9.	Define Output system ?	CO4	1,2,3,5	2
10.	Define Reading Console input.	CO4	1,2,3,5	2

10 Marks Questions :

S.No	Question	CO	PO	BL
1.	Explain the Exception handling Fundamentals.	CO4	1,2,3,5	2
2.	Explain about the try and catch block with an example program.	CO4	1,2,3,5	2
3.	Explain Java Built-in Exceptions.	CO4	1,2,3,5	2
4.	Explain Chained Exceptions with an example program.	CO4	1,2,3,5	2
5.	Explain I/O basics and Stream	CO4	1,2,3,5	6
6.	Describe Reading Console input with an example program.	CO4	1,2,3,5	3
7.	Describe Applet Class with an example program	CO4	1,2,3,5	6
8.	Explain the Lifecycle of an Applet	CO4	1,2,3,5	4

7. Describe Applet Class with an example program ?

Ans:

An applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server. Applets are used to make the website more dynamic and entertaining.

1. All applets are sub-classes (either directly or indirectly) of *java.applet.Applet* class.
2. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.
3. In general, execution of an applet does not begin at main() method.
4. Output of an applet window is not performed by *System.out.println()*. Rather it is handled with various AWT methods, such as *drawString()*.

Creating Hello World applet :

Let's begin with the HelloWorld applet :

```
// A Hello World Applet
// Save file as HelloWorld.java

import java.applet.Applet;
import java.awt.Graphics;

// HelloWorld class extends Applet
public class HelloWorld extends Applet
{
   // Overriding paint() method
   @Override
   public void paint(Graphics g)
   {
      g.drawString("Hello World", 20, 20);
   }
```

}

Explanation:

1. The above java program begins with two import statements. The first import statement imports the Applet class from applet package. Every AWT-based(Abstract Window Toolkit) applet that you create must be a subclass (either directly or indirectly) of Applet class. The second statement import the Graphics class from AWT package.
2. The next line in the program declares the class HelloWorld. This class must be declared as public because it will be accessed by code that is outside the program. Inside HelloWorld, **paint( )** is declared. This method is defined by the AWT and must be overridden by the applet.
3. Inside **paint( )** is a call to *drawString( )*, which is a member of the Graphics class. This method outputs a string beginning at the specified X,Y location. It has the following general form:

void drawString(String message, int x, int y)

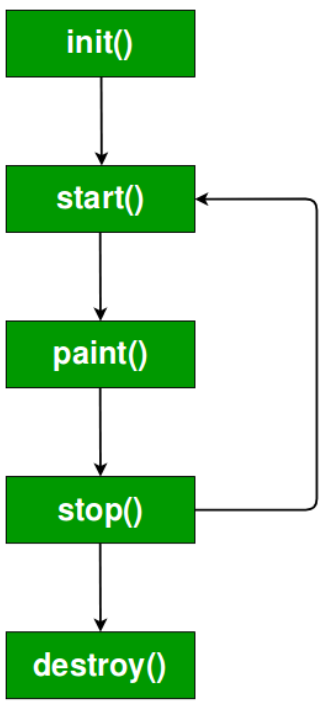
Here, message is the string to be output beginning at x,y. In a Java window, the upper-left corner is location 0,0. The call to *drawString( )* in the applet causes the message "Hello World" to be displayed beginning at location 20,20.

Output:

8. Explain the Lifecycle of an Applet ?

Ans:

Life cycle of an applet :



It is important to understand the order in which the various methods shown in the above image are called. When an applet begins, the following methods are called, in this sequence:

1. init( )
2. start( )
3. paint( )

When an applet is terminated, the following sequence of method calls takes place:

4. stop( )
5. destroy( )

Let's look more closely at these methods.

1. init( ) : The **init( )** method is the first method to be called. This is where you should initialize variables. This method is called **only once** during the run time of your applet.
2. start( ) : The **start( )** method is called after **init( )**. It is also called to restart an applet after it has been stopped. Note that **init( )** is called once i.e. when the first time an applet is loaded whereas **start( )** is called each time an applet's HTML document is displayed onscreen. So, if a user leaves a web page and comes back, the applet resumes execution at **start( )**.
3. paint( ) : The **paint( )** method is called each time an AWT-based applet's output must be redrawn. This situation can occur for several reasons. For example, the window in which the applet is running may be overwritten by another window and then uncovered. Or the applet window may be minimized and then restored. **paint( )** is also called when the applet begins execution. Whatever the cause, whenever the applet must redraw its output, **paint( )** is called.
The **paint( )** method has one parameter of type Graphics. This parameter will contain the graphics context, which describes the graphics environment in which the applet is running. This context is used whenever output to the applet is required. Note: This is the only method among all the method mention above, which is parameterized. It's prototype is public void paint(Graphics g)
where g is an object reference of class Graphic.
4. stop( ) : The **stop( )** method is called when a web browser leaves the HTML document containing the applet—when it goes to another page, for example. When **stop( )** is called, the applet is probably running. You should use **stop( )** to suspend threads that don't need to run when the applet is not visible. You can restart them when **start( )** is called if the user returns to the page.
5. destroy( ) : The **destroy( )** method is called when the environment determines that your applet needs to be removed completely from memory. At this point, you should free up any resources the applet may be using. The **stop( )** method is always called before **destroy( )**.

Unit - V

Event Handling:

The Delegation event model-Events,Event sources,Event Listeners,Event classes,Handling mouse events,Inner classes

2 Marks Questions :

S.No	Question	CO	PO	BL
1.	Define Delegation event model	CO5	1,2,3,5	2
2.	Define Events	CO5	1,2,3,5	2
3.	Define Event sources.	CO5	1,2,3,5	1
4.	Define Event Listeners	CO5	1,2,3,5	1
5.	Define Event classes.	CO5	1,2,3,5	2
6.	Define Inner classes.	CO5	1,2,3,5	1

10 Marks Questions :

S.No	Question	CO	PO	BL
1.	Explain Delegation event model and Events.	CO5	1,2,3,5	2
2.	Explain Event sources and Event LIsteners.	CO5	1,2,3,5	4
3.	Explain Event classes with an example.	CO5	1,2,3,5	2
4.	Explain Handling mouse events with an example.	CO5	1,2,3,5	2
5.	Describe Inner classes with an example.	CO5	1,2,3,5	4

3.Explain Event classes with an example.

Ans:

Event Classes

The classes that represent events are at the core of Java's event handling mechanism. Thus, a discussion of event handling must begin
event classes. It is important to understand, however, that Java defines several types of events and that not all event classes

Event Class	Description
ActionEvent	Generated when a button is pressed, a list item is double-clicked, or a menu item is selected.
AdjustmentEvent	Generated when a scroll bar is manipulated.
ComponentEvent	Generated when a component is hidden, moved, resized, or becomes visible.
ContainerEvent	Generated when a component is added to or removed from a container.
FocusEvent	Generated when a component gains or loses keyboard focus.
InputEvent	Abstract superclass for all component input event classes.
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.
KeyEvent	Generated when input is received from the keyboard.
MouseEvent	Generated when the mouse is dragged, moved, clicked, pressed, or released; also generated when the mouse enters or exits a component.
MouseWheelEvent	Generated when the mouse wheel is moved.
TextEvent	Generated when the value of a text area or text field is changed.
WindowEvent	Generated when a window is activated, closed, deactivated, deiconified, iconified, opened, or quit.

Eg:

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

class EventClass extends Frame implements ActionListener {

   TextField textField;

   GFGTop()
   {
      // Component Creation
      textField = new TextField();

      // setBounds method is used to provide
      // position and size of the component
      textField.setBounds(60, 50, 180, 25);
 Button button = new Button("click Here");
      button.setBounds(100, 120, 80, 30);

      // Registering component with listener
      // this refers to current instance
      button.addActionListener(this);

      // add Components
      add(textField);
      add(button);

      // set visibility
      setVisible(true);
   }
 public void actionPerformed(ActionEvent e)
   {
      // Setting text to field
      textField.setText("EventClass!");
   }

   public static void main(String[] args)
   {
     new EventClassTop();
   }
}
```

4.Explain Handling Mouse Event with an example?

Ans:

```
// Demonstrate the mouse event handlers.



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; // coordinates of mouse



public void init() {



addMouseListener(this);
```

```
addMouseMotionListener(this);

}

// Handle mouse clicked.

public void mouseClicked(MouseEvent me) {

// save coordinates

mouseX = 0;

mouseY = 10;

msg = "Mouse clicked.";

repaint();

}

// Handle mouse entered.

public void mouseEntered(MouseEvent me) {

// save coordinates

mouseX = 0;
```

```
mouseY = 10;

msg = "Mouse entered.";

repaint();

}

// Handle mouse exited.

public void mouseExited(MouseEvent me) {

// save coordinates

mouseX = 0;

mouseY = 10;

msg = "Mouse exited.";

repaint();

}

// Handle button pressed.

public void mousePressed(MouseEvent me) {
```

```
// save coordinates

mouseX = me.getX();

mouseY = me.getY();

msg = "Down";

repaint();

}

// Handle button released.

public void mouseReleased(MouseEvent me) {

// save coordinates

mouseX = me.getX();

mouseY = me.getY();

msg = "Up";

repaint();

}
```

```
// Handle mouse dragged.

public void mouseDragged(MouseEvent me) {

// save coordinates

mouseX = me.getX();

mouseY = me.getY();

msg = "*";

showStatus("Dragging mouse at " + mouseX + ", " + mouseY);

repaint();

}

// Handle mouse moved.

public void mouseMoved(MouseEvent me) {

// show status

showStatus("Moving mouse at " + me.getX() + ", " + me.getY());

}
```

```
// Display msg in applet window at current X,Y location.

public void paint(Graphics g) {

g.drawString(msg, mouseX, mouseY);

}

}
```

OutPut: