UNIT – V

1. a) What are the key differences between Swing and AWT?

Java AWT	***Java Swing***
AWT is an API to develop GUI applications in Java.	Swing is a part of Java Foundation Classes and is used to GUI applications.
AWT provides limited set of components.	Swing provides additional components with additional features.
Components are platform dependent.	Components are platform independent
Components of AWT are heavy weighted.	The components of Java Swing are lightweight
Execution Time is more than Swing.	Execution Time is less than AWT.
Java AWT has comparatively less functionality as compared to Swing.	Java Swing has more functionality as compared to AWT.
MVC pattern is not supported by AWT	MVC pattern is supported by Swing.
AWT components require java.awt package.	Swing components requires javax.swing package.

b)Explain the following methods:

i) drawRect() - The drawRect() method in Java is a part of the Graphics class, which is used for drawing the outline of a rectangle.

Syntax: ***drawRect(int x, int y, int width, int height)***

x: The x-coordinate of the top-left corner of the rectangle.

y: The y-coordinate of the top-left corner of the rectangle.

width: The width of the rectangle.

height: The height of the rectangle.

(ii) drawOval() - The drawOval() method in Java is a part of the Graphics class, which is used for drawing the outline of an Oval inscribed within a specified rectangle.

Syntax: drawOval(int x, int y, int width, int height)

x: The x-coordinate of the top-left corner of the bounding rectangle of the oval.

y: The y-coordinate of the top-left corner of the bounding rectangle of the oval.

width: The width of the bounding rectangle of the oval.

height: The height of the bounding rectangle of the oval.

iii) drawLine() - The drawLine() method in Java is part of the Graphics class, which is used for drawing a straight line between two points.

Syntax: drawLine(int x1, int y1, int x2, int y2)

x1: The x-coordinate of the starting point of the line.
y1: The y-coordinate of the starting point of the line.
x2: The x-coordinate of the ending point of the line.
y2: The y-coordinate of the ending point of the line.

iv) drawArc() - The drawArc() method in Java is part of the Graphics class, which is used for draw the outline of an arc.

Syntax: drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)

x: The x-coordinate of the upper-left corner of the rectangle that bounds the arc.

y: The y-coordinate of the upper-left corner of the rectangle that bounds the arc.

width: The width of the rectangle that bounds the arc.

height: The height of the rectangle that bounds the arc.

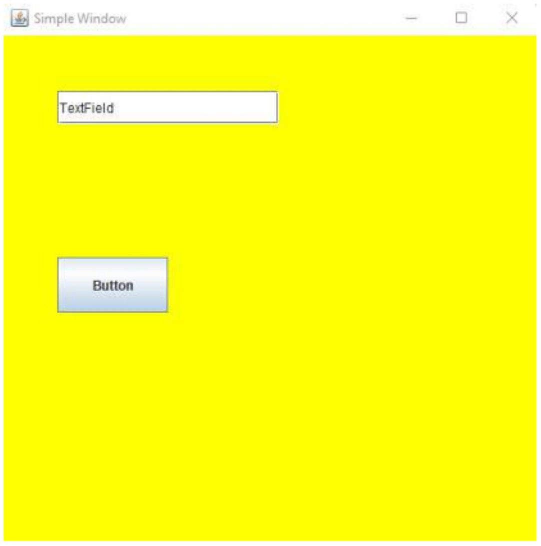
startAngle: The starting angle of the arc in degrees,

arcAngle: The angular extent of the arc in degrees, measured from the starting angle.

2. Write a Swing program that displays a simple window with a button and a text field.

Program:

```
import javax.swing.*;

import java.awt.*;

public class SimpleWindow{

        public static void main(String[] args){

        Frame f = new JFrame ("Frame 1");

        f.setSize(500,500); //Setting the frame Size

        f.getContentPane().setBackground(Color.yellow); //Setting the frame Color

        f.setVisible(true); //Setting the Visibility

        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //Close frame on Click


       JTextField textField = new JTextField("TextField"); //Creation of TextField

       textField.setBounds(50,50,200,30); //Textfield Location and size

       JButton button = new JButton("Button"); //Creation of a Button

       button.setBounds(50,200,100,50); //Button Location and size

       f.add(textField); //Adding textfield to frame

      f.add(button); //Adding Button to frame

     }

}
```



3. Explain the various components in Swing with an example.

Swing is a part of Java's standard library for creating graphical user interfaces (GUIs). It provides a rich set of components for building user interfaces. Here are some of the key Swing components, explained with a comprehensive example:

1. **JFrame:** The main window.
2. **JPanel:** A container that can hold and organize other components.
3. **JLabel:** A display area for a short text string or an image.
4. **JButton:** A button that can trigger an action when clicked.
5. **JTextField:** A single-line text input field.
6. **JTextArea:** A multi-line text input area.
7. **JCheckBox:** A box that can be checked or unchecked.
8. **JRadioButton:** A button in a group of radio buttons, where only one button in the group can be selected.
9. **JComboBox:** A drop-down list of items.
10. **JList:** A list of items.
11. **JMenuBar, JMenu, JMenuItem:** Components for creating menus.

Program:

```
import javax.swing.*;

import java.awt.*;

public class SwingComponentsExample {
   public static void main(String[] args) {
      JFrame f = new JFrame("Swing Components Example");
      f.setSize(500, 400);
      f.getContentPane().setBackground(Color.black);
      f.setLayout(null); // Using null layout for absolute positioning
      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JLabel l = new JLabel("This is a JLabel");
      l.setBounds(50, 20, 200, 30);
      f.add(l);

      JButton b = new JButton("Click Me");
      b.setBounds(50, 60, 200, 30);
      f.add(b);
```

```
        JTextField t = new JTextField("Enter text here");
        t.setBounds(50, 100, 200, 30);
        f.add(t);

        JCheckBox c = new JCheckBox("Check me");
        c.setBounds(50, 230, 200, 30);
        f.add(c);

        f.setVisible(true);
    }
}
```

4. What is an applet in java? How to create an applet? Explain the life cycle of an applet with an example for each.

- An applet is a small application designed to be executed within a web browser or an applet viewer. It is a special kind of Java program that can be embedded in an HTML page.
- Applets are primarily used for providing interactive features to web applications that cannot be provided by HTML alone. They have a graphical user interface (GUI) and can respond to user input.
- An applet can be run in the context of a web page using a Java-enabled browser or an appletviewer tool that comes with the Java Development Kit (JDK).

Creation of an applet:

1. ***Create the Applet Class:***
   - Extend the Applet class.
   - Override the init(), start(), stop(), and destroy() methods if necessary.
   - Implement the paint() method to display graphics.

2. ***Compile the Applet:***
   - Use the javac compiler to compile the applet source code.

3. ***Create an HTML File:***
   - Embed the compiled applet class in an HTML file using the <applet> tag or the <object> tag (recommended for newer browsers).
4. ***Run the Applet:***
   - Use an applet viewer or a web browser to run the applet.

Life Cycle of an Applet:

The applet life cycle can be defined as the process of how the applet is created, started, stopped, and destroyed during the entire execution of its application. It basically has five core methods and these 5 methods are defined in the Applet class, and you can override them to provide specific behavior for your applet. The main lifecycle methods are:

1. ***init()***

The init() method is called once when the applet is first loaded. It is used for initialization and setup purposes, such as loading resources or initializing variables.

2. ***start()***

The start() method is called after init(), and it is called each time the applet's HTML page is displayed. It is used to start or resume any execution, such as starting animations or threads.

3. ***stop()***

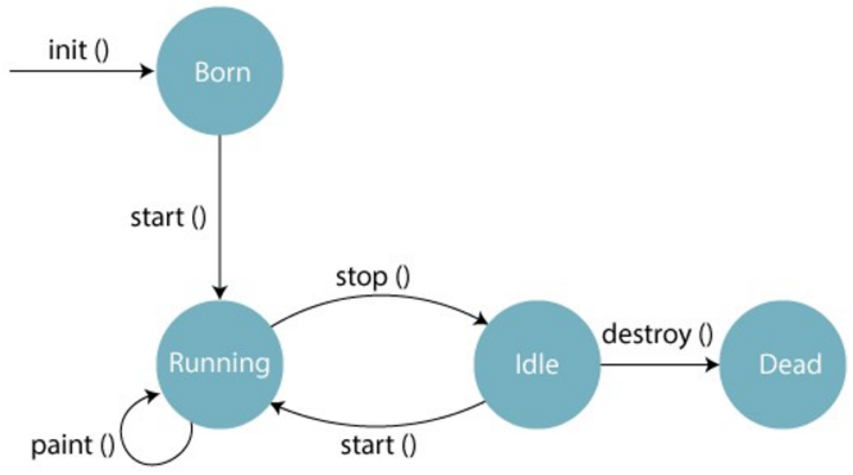
The stop() method is called when the applet's HTML page is no longer visible. It is used to suspend any execution, such as stopping animations or threads.

4. ***destroy()***

The destroy() method is called once when the applet is being destroyed. It is used for cleanup, such as releasing resources.

5. ***paint()***

The paint() method is called to draw the applet's user interface. It is called when the applet first appears, and again whenever the applet needs to be redrawn, such as when it is resized or uncovered.



Program:

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

public class AppletLifecycleExample extends Applet {

   public void init() {
      System.out.println("Applet initialized.");
   }

   public void start() {
      System.out.println("Applet started.");
   }

   public void stop() {
      System.out.println("Applet stopped.");
   }

   public void destroy() {
      System.out.println("Applet destroyed.");
   }

   public void paint(Graphics g) {
      g.drawString("Applet is being painted.", 20, 100);
   }
}
```

5. Discuss the major differences between application and applet?

Parameters	Application	Applet
Definition	Applications are just like a Java program that can be executed independently in the computer without using the web browser.	Applets are small Java programs that are designed to be included with the HTML web browser for execution.
main () method	Requires a main() method for its execution.	Does not require the main() method for its execution instead init() method is required.
Installation	the installation of a Java application on the local computer is required.	The Java applet does not need to be installed beforehand.
Compilation	The “javac” command is used to compile application programs, which are then executed using the “java” command.	Applet programs are compiled with the “javac” command and run using either the “appletviewer” command or the web browser.
Execution	It cannot run on its own; it needs JRE to execute.	It can be executed using a Java-enabled web browser.
File access	It can easily access a file or data available on a computer system or device.	It cannot access the file or data found on any local system or computer.
Read and Write Operation	It supports the reading and writing of files on the local computer.	It does not support the reading and writing of files on the local computer.
Connectivity with server	It is possible to establish connections with other servers.	It cannot establish connection to other servers.
Security	Java applications are pretty trusted, and thus, come with no security concerns.	Java applets are less reliable. So, they need to be safe.

6. a) What is panel container in AWT?

In the Abstract Window Toolkit (AWT), a Panel is a simple container class that can hold and organize a collection of components. It is a subclass of Container, which is itself a subclass of Component. Panel provides a space in which an application can attach any other AWT components, including other containers.

Program to create a panel:

```
import java.awt.*;
public class PanelExample {
   public static void main(String[] args) {
      Frame f = new Frame("Panel Example"); // Creation of a frame
      frame.setSize(300, 200);
      Panel p = new Panel(); // Creation of a panel
      Button b1  = new Button("Button 1");  // Creation of a Button 1
      Button b2  = new Button("Button 2"); // Creation of a Button 2
      p.add(b1); // Add Button 1 to the panel
      p.add(b2); // Add Button 2 to the panel
      f.add(p); // Add the panel to the frame
      frame.setVisible(true); // Set frame size and make it visible
   }
}
```

b) Explain the concept of Graphics in applet.

Graphics in an applet refers to the capability to draw lines, rectangles, ovals, polygons, and text, as well as setting colors and fonts, and images on the applet's window using the Graphics class provided by the Java Abstract Window Toolkit (AWT).

java.awt.Graphics class provides many methods for graphics programming. The Graphics class includes various methods for drawing shapes and text. Some common methods include:

drawLine(int x1, int y1, int x2, int y2): Draws a line between two points.

drawRect(int x, int y, int width, int height): Draws a rectangle.

drawString(String str, int x, int y): Draws a string of text.

setColor(Color c): Sets the current drawing color.

fillRect(int x, int y, int width, int height): Fills a rectangle with the current color.

7. Explain MVC architecture with respect to swing components with suitable examples.

- The Model-View-Controller (MVC) architecture is a design pattern used to separate an application into three interconnected components: Model, View, and Controller.
- This separation helps in managing complexity, improving flexibility, and facilitating unit testing.
- Swing, the GUI toolkit for Java, adheres to the MVC architecture, though in a slightly modified form often referred to as the Model-Delegate architecture.

MVC Components in Swing

- **Model**: Represents the data and the business logic of the application. It notifies the view of any data changes.
- **View**: Represents the presentation layer (UI) and displays the data to the user. It listens to model changes to update the UI accordingly.
- **Controller**: Handles user input and updates the model. It interprets the user actions, updates the model, and refreshes the view.

Example: JButton in MVC Architecture

Let's take the JButton as an example to explain the MVC architecture in Swing.

1. **Model**: ButtonModel is the model for JButton. It contains the state of the button, such as whether it is pressed, armed, or selected.
2. **View**: The view is the visual representation of the JButton, which is rendered by the LookAndFeel classes.
3. **Controller**: The controller is the part of the JButton that handles user interactions, such as mouse clicks. This is typically implemented using event listeners.

8. Define AWT? Explain the various AWT controls available in java with suitable example.

- The Abstract Window Toolkit (AWT) is a set of APIs provided by Java for creating graphical user interfaces (GUIs) and graphical applications. It includes a collection of classes for creating and managing windows, dialog boxes, buttons, scrollbars, text fields, and other GUI components.
- AWT is part of the Java Foundation Classes (JFC) and provides the foundation for building platform-independent windowing, graphics, and user-interface applications.

AWT Controls: AWT provides a variety of controls, also known as components, to build user interfaces. Here are some of the most commonly used AWT controls:

1. **Label**: Displays a single line of read-only text.
2. **Button**: A clickable button.
3. **TextField**: A single-line text input field.
4. **TextArea**: A multi-line text input field.
5. **Checkbox**: A box that can be checked or unchecked.
6. **CheckboxGroup**: Groups multiple checkboxes to allow only one to be selected at a time (radio buttons).
7. **Choice**: A drop-down list of items.
8. **List**: A list of items where multiple selections are allowed.
9. **Scrollbar**: A horizontal or vertical scrollbar.
10. **Canvas**: A blank area where custom graphics can be drawn.
11. **Panel**: A container to group other components.
12. **Frame**: A top-level window with a title and border.
13. **Dialog**: A pop-up window that can be modal or non-modal.
14. **MenuBar, Menu, MenuItem**: Components for creating menu bars and drop-down menus.

Example for AWT Controls:

```
import java.awt.*;

public class AWTExample {
   public static void main(String[] args) {
      Frame f = new Frame("AWT Example"); // Creation of a frame
      frame.setSize(400, 300); // Setting frame size
      f.setBackground(Color.yellow); //Setting background color for frame
      f.setVisible(true); //Setting visibility of frame

      Panel p = new Panel(); // Creation of a panel to hold the components
      p.setSize(150,150); // Setting panel size
      p.setLocation(50,100); // Setting the location of a panel
      p.setBackground(Color.red); //Setting background color for panel

      Button b = new Button("Button"); // Creation of a button
      b.setBounds(10,50,70,70); // Setting location and dimensions of button

      Label l = new Label("Abstract Window"); // Creation of a label

      TextField t = new TextField("Java Program"); // Creation of a text field

      Checkbox c= new Checkbox("Choose Me"); // Creation of a checkbox
      c.setBounds(50,50,50,50); // Setting location and dimensions of checkbox

      p.add(b); //adding button to panel

      p.add(l); //adding label to panel

      p.add(t); //adding textfield to panel

      p.add(c); //adding checkkbox to panel

      f.add(p); //adding panel to frame

   }

}
```

9. Explain how to use the AWT package for adding graphical components to an applet.

Steps to Add Graphical Components to an Applet

1. **Import the AWT and Applet Packages**:

   Import the necessary classes from the java.awt and java.applet packages.

2. **Create the Applet Class**:

   Define a public class that extends Applet.

3. **Override the init() Method**:

   Override the init() method to set up the user interface. This method is called once when the applet is first loaded.

4. **Add Components to the Applet**:

   Create and add AWT components to the applet using the add() method.

Example Program:

```
import java.awt.*;
public class AWTExample {
   public static void main(String[] args) {
      Frame f = new Frame("AWT Example"); // Creation of a frame
      frame.setSize(400, 300); // Setting frame size
      f.setBackground(Color.yellow); //Setting background color for frame
      f.setVisible(true); //Setting visibility of frame

      Button b = new Button("Button"); // Creation of a button
      b.setBounds(10,50,70,70); // Setting location and dimensions of button

      Label l = new Label("Abstract Window"); // Creation of a label
      TextField t = new TextField("Java Program"); // Creation of a text field

      f.add(b); //adding button to panel
      f.add(l); //adding label to panel
      f.add(t); //adding textfield to panel
   }
}
```

10. How do we create an applet? Explain the different ways to execute the applet program in a browser.

Steps to Create an Applet:

1. **Import Required Packages**:
   Import the java.applet.Applet and java.awt.* packages.
2. **Define the Applet Class**:
   Create a public class that extends Applet.
3. **Override Methods**:
   Override methods like init(), start(), stop(), and destroy().
4. **Add Components and Logic**:
   Add components and any custom logic within the lifecycle methods.

Program:

```
import java.applet.Applet;
import java.awt.*;
public class SimpleApplet extends Applet {

   public void init() {
      setLayout(new FlowLayout());

      Label label = new Label("Hello, Applet!");
      add(label);

      Button button = new Button("Click Me");
      add(button);
   }

   public void start() {
      // Code to execute when the applet is started
   }

   public void stop() {
      // Code to execute when the applet is stopped
   }

   public void destroy() {
      // Code to execute when the applet is destroyed
   }
}
```

Ways to Execute the Applet:

There are two different ways to execute an apple

1. **Using Applet Viewer**:

   The Applet Viewer is a command-line utility that comes with the JDK and can be used to run and debug applets independently of a web browser.

   appletviewer SimpleApplet.html

2. **Using a Java-enabled html/browser**:

   - Create an HTML file with an <applet> tag.
   - Specify the code attribute with the name of your applet class (e.g., code=" SimpleApplet.class").
   - Set the width and height attributes to define the applet's size

Example:

```
<!DOCTYPE html>
<html>
<body>
<applet code=" SimpleApplet.class" width="400" height="100">
</applet>
</body>
 </html>
```