Unit 2 :User Interface Components with Swing

AWT and Swing :

- AWT and Swing are used to develop window-based applications in Java.
- Awt is an abstract window toolkit that provides various component classes like Label, Button, TextField, etc., to show window components on the screen. All these classes are part of the Java.awt package.

- On the other hand, Swing is the part of JFC (Java Foundation Classes) built on the top of AWT and written entirely in Java. The javax.swing API provides all the component classes like JButton, JTextField, JCheckbox, JMenu, etc.

AWT:

- AWT stands for Abstract Window Toolkit.
- It is a platform-dependent API to develop GUI (Graphical User Interface) or window-based applications in Java.
- It was developed by heavily Sun Microsystems In 1995.
- It is heavy-weight in use because it is generated by the system's host operating system.
- It contains a large number of classes and methods, which are used for creating and managing GUI.

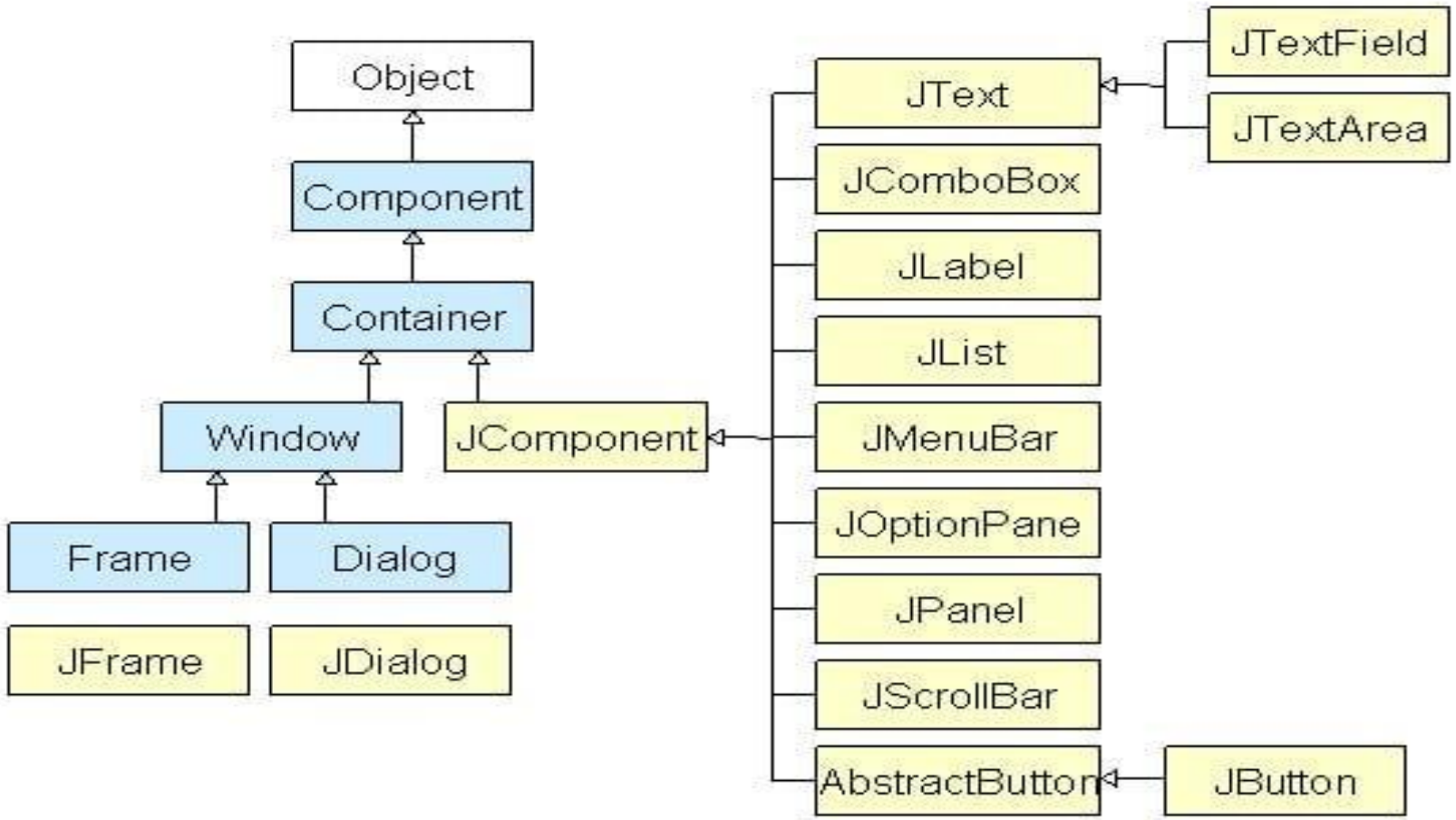
Swing:

- Swing is a lightweight Java graphical user interface (GUI) that is used to create various applications.
- Swing has platform-independent components.
- It enables the user to create buttons and scroll bars.
- Swing includes packages for creating desktop applications in Java.
- Swing components are written in Java language.
- It is a part of Java Foundation Classes(JFC).

Difference between AWT and Swing :

AWT	Swing
Java AWT is an API to develop GUI applications in Java	Swing is a part of Java Foundation Classes and is used to create various applications.
The components of Java AWT are heavy weighted.	The components of Java Swing are light weighted.
Java AWT has comparatively less functionality as compared to Swing.	Java Swing has more functionality as compared to AWT.
The execution time of AWT is more than Swing.	The execution time of Swing is less than AWT.
The components of Java AWT are platform dependent.	The components of Java Swing are platform independent.
MVC pattern is not supported by AWT.	MVC pattern is supported by Swing.
AWT provides comparatively less powerful components.	Swing provides more powerful components.
AWT components require java.awt package	Swing components requires javax.swing package
AWT is a thin layer of code on top of the operating system.	Swing is much larger swing also has very much richer functionality.
AWT stands for Abstract windows toolkit .	Swing is also called as JFC(java Foundation classes). It is part of oracle's JFC.
Using AWT , you have to implement a lot of things yourself .	Swing has them built in.

Java Swing Class Hierarchy Diagram :



Swing Components/Control and Containers :

- A component is an independent visual control and Java Swing Framework contains a large set of these components which provide rich functionalities and allow high level of customization. They all are derived from JComponent class.
- All these components are lightweight components. This class provides some common functionality like pluggable look and feel, support for accessibility, drag and drop, layout, etc.

- A container holds a group of components. It provides a space where a component can be managed and displayed.

Containers are of two types:

1.Top level Containers

- It inherits Component and Container of AWT.
- It cannot be contained within other containers.
- Heavyweight.
- Example: JFrame, JDialog, JApplet

2.Lightweight Containers

1. It inherits JComponent class.
2. It is a general purpose container.
3. It can be used to organize related components together.
4. Example: JPanel

1. JFrame :

- The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class.
- JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.
- It can be created in two ways:

1. **By creating object of the frame classs**

   Eg. JFrame F = new JFrame();

2. **By extending the frame class**

```
class ClassName extends JFrame{
}
```

Constructors:

- **JFrame() :**

  It constructs a new frame that is initially invisible.

- **JFrame(GraphicsConfiguration gc) :**

  It creates a Frame in the specified GraphicsConfiguration of a screen device and a blank title.

- **JFrame(String title) :**

  It creates a new, initially invisible Frame with the specified title.

- **JFrame(String title, GraphicsConfiguration gc) :**

  It creates a JFrame with the specified title and the specified GraphicsConfiguration of a screen device.

Methods of the JFrame class :

1. setVisible():

- This method is used to control whether a component will be displayed on the screen or not.
- It takes a Boolean value as an argument.

2. setDefaultCloseOperation()

- This method specifies many options for the close Button. We can choose any one of the given constants to determine the close Button's nature when someone clicks on it.

- Constants are given below :

i. **JFrame.DO_NOTHING_ON_CLOSE –** "As the name suggests, it neglects the click, and nothing happens when someone clicks on the Close Button. "

ii. **JFrame.HIDE_ON_CLOSE –**" As the name suggests, it only hides the frame when someone clicks on the Close Button, but the application still remains on the running state. "

iii. **JFrame.DISPOSE_ON_CLOSE –**"As the name suggests, it disposes the frame when someone clicks on the Close Button, but the application still remains on the running state. "

iv. **JFrame.EXIT_ON_CLOSE –**"As the name suggests, it exits the application when someone clicks on the Close Button and removes the program from memory permanently. "

3. setSize():

- As the name suggests this method is used to set the size of the window displayed on the screen .
- By default the size is passed to the **setSize()** method is ” 0 pixel x 0 pixel “.
- it takes two integer type arguments. first argument specifies the width of the window and second argument specifies the height of the window .
- example – **setSize(100,200) .**

4. setLocation()

- This method is used to set the window’s location on the screen.
- The default location of the window is the top-left corner of the screen.
- It takes two integer type arguments. The first argument specifies the window’s location on the x-axis, and the second argument specifies the window’s location on the y-axis.

5. setBounds()

- This method is used to set the size and position(Location) of the window simultaneously.
- It takes four integer type arguments in which the first two arguments specify the window’s position on the x-axis and y-axis, and the last two arguments specify the size of the window in width and height.
- example –**setBounds(200,200,100,200)**.

6. setTitle()

- This method is used to set the title of the window.
- It takes the string/name of the window to be displayed on the screen as its argument.

7. setBackground()

Steps to Add Color :

- Import the **java.awt.*** package. This package makes the **setBackground()** method available.
- Since the Background Color can only be added to the container, we have to make a container object for the window jframe.
- This is done by calling the **getContentPane()** method through jframe.
- Now we have to call the **setBackground()** method using the container's object, which takes various static pre-defined fields of Color class.

8. setResizable()

- This method takes a boolean value as its parameter.

i. **setResizable(true) –** "allows the user to resize the window. "
ii. **setResizable(false)-** "restricts the user to resize the window. "

2. JLable :

- ❖ JLable is a component which displays a readable text or image in the swing container UI.
- ❖ The application user cannot edit the text, but text can be changed by application.
- ❖ The JLabel Contains 4 constructors. They are as following:

1. JLabel()
2. JLabel(String s)
3. JLabel(Icon i)
4. JLabel(String s, Icon i, int horizontalAlignment)

3. JTextField

- ❖ JTextField object is a text component that allows for the editing of a single line of text.
- ❖ The JTextField Contains 4 constructors. They are as following:

1. **JTextField**() : Creates a new TextField
2. **JTextField**(int cols) : Creates a new empty TextField with the specified number of columns.
3. **JTextField**(String str, int cols) : Creates a new TextField initialized with the specified text and columns.
4. **JTextField**(String str) : Creates a new TextField initialized with the specified text.

4. JPasswordField

- ❖ JPasswordField object is a text component specialized for password entry.
- ❖ Following are the constructors:

1. JPasswordField()
2. JPasswordField(int columns)
3. JPasswordField(String text)
4. JPasswordField(String text, int columns)

5. JTextArea :

- JTextArea object is a text component that allows editing of a multiple lines of text.
- Following are the constructors:

1. JTextArea()
2. JTextArea(String s)
3. JTextArea(int row, int column)
4. JTextArea(String s, int row, int column)

6. JButton :

- JButton class provides functionality of a button. It is used to create button component.
- Following are the constructors:

1. JButton() : It creates a button with no text and icon.
2. JButton(String s) : It creates a button with the specified text.
3. JButton(Icon i) : It creates a button with the specified icon object.

7. JCheckBox :

- The JCheckBox class is used to create a checkbox.
- It is used to turn an option on (true) or off (false). Clicking on a CheckBox changes its state from "on" to "off" or from "off" to "on "
- Following are the constructors:

1. JCheckBox() : Creates an initially unselected check box button with no text, no icon.
2. JChechBox(String s) : Creates an initially unselected check box with text.
3. JCheckBox(String text, boolean selected) : Creates a check box with text and specifies whether or not it is initially selected.

8. JRadioButton :

- The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options.
- It should be added in ButtonGroup to select one radio button only.
- Following are the constructors:

1. JRadioButton()
2. JRadioButton(String s)
3. JRadioButton(String s, boolean selected)

9. JComboBox :

- JComboBox class is used to show popup menu of choices.
- Choice selected by user is shown on the top of a menu.

- Following are the constructors:

1. JComboBox() : Creates a JComboBox with a default data model.
2. JComboBox(Object[] items) : Creates a JComboBox that contains the elements in the specified array.

10. JTable:

- The JTable class is used to display data in tabular form.
- It is composed of rows and columns.
- Following are the constructors:

1. JTable() : Creates a table with empty cells.
2. JTable(Object[][] rows, Object[] columns): Creates a table with the specified data.

10. JList:

- The object of JList class represents a list of text items. The list of text items can be set up so that the user can choose either one item or multiple items.

- Following are the constructors:

1. JList() : Creates a JList with an empty, read-only, model.
2. JList(ary[] listData): Creates a JList that displays the elements in the specified array.
3. JList(ListModel<ary> dataModel) : Creates a JList that displays elements from the specified, non-null, model.

11. JOptionPane

- The JOptionPane class is used to provide standard dialog boxes such as message dialog box, confirm dialog box and input dialog box.
- These dialog boxes are used to display information or get input from the user.
- Following are the constructors:

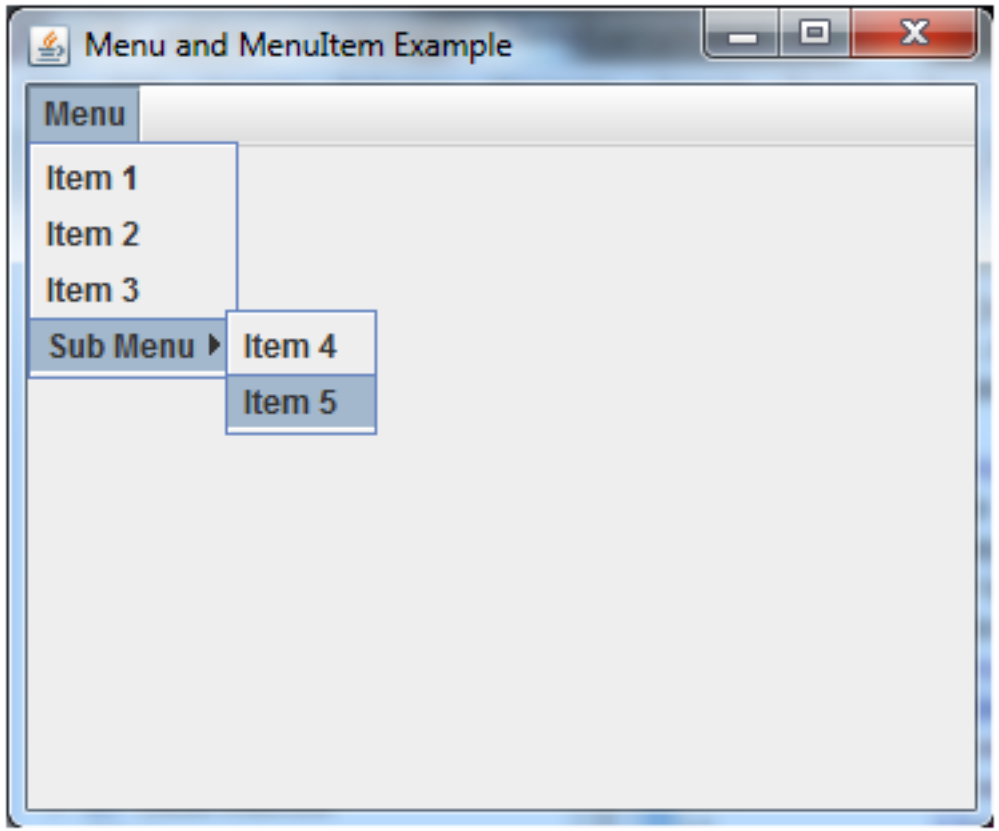
1. JOptionPane() **:** It is used to create a JOptionPane with a test message.
2. JOptionPane(Object message) : It is used to create an instance of JOptionPane to display a message.

12. JMenuBar, JMenu, JMenuItem:

- JMenuBar class used to display menubar on the window or frame.
- Jmenu beasically represents menu. It contains several JMenuIteam object.

Following are the constructors:

- JMenuBar()
- JMenu(String name)
- JMenu()



13. JTree

- Swing provides a JTree component that allows you to create tree-like structures for displaying and organizing data.
- You can use DefaultMutableTreeNode objects to represent nodes in the tree, and these nodes can have child nodes.
- JTree is commonly used for file explorers, navigation menus, and hierarchical data visualization.

e.g.

```
DefaultMutableTreeNode depNode = new DefaultMutableTreeNode("Department");
JTree jTree = new JTree(depNode);
```

14. JFileChooser:

- Swing provides a JFileChooser component for creating file dialog boxes that allow users to select files or directories.
- It's a convenient way to implement file open and save functionality in your Swing applications.

```
JFileChooser fileChooser = new JFileChooser();
int returnValue = fileChooser.showOpenDialog(null);
if (returnValue == JFileChooser.APPROVE_OPTION) {
   File selectedFile = fileChooser.getSelectedFile();
}
```


15. Color Chooser:

- Swing provides a JColorChooser component for creating color selection dialogs.
- You can use it to allow users to pick colors for various elements in your GUI.

```
Color selectedColor = JColorChooser.showDialog(this, "Choose a Color", Color.WHITE);
```

Layout Management

- A layout manager determines the location and size of components placed into a container.
- Different layout manager classes use different algorithms for determining size and location.
- Even if you do not use the layout manager, the components are still positioned by the default layout manager.
- LayoutManager is an interface that is implemented by all the classes of layout managers.

There are the following classes that represent the layout managers:

1. BorderLayout
2. FlowLayout
3. GridLayout
4. GridBagLayout
5. GroupLayout

BorderLayout

- The BorderLayout is used to arrange the components in **five regions**: **north, south, east, west, and center.** Each region (area) may contain one component only. It is the default layout of a frame or window. The BorderLayout provides five constants for each region:

There are two useful constructors.

a. **new BorderLayout()**
b. **new BorderLayout(int hgap, int vgap)**

When components are added to a container that uses a BorderLayout, they should be added with the following kind of statement.

container.add(component, *whr*);

The parameter *whr* should be one of the following.

BorderLayout.NORTH
BorderLayout.EAST
BorderLayout.SOUTH
BorderLayout.WEST
BorderLayout.CENTER

FlowLayout

- The FlowLayout arranges the components in a directional flow, either from left to right or from right to left. Normally all components are set to one row, according to the order of different components. If all components can not be fit into one row, it will start a new row and fit the rest in.

Constructors :

1. FlowLayout(): construct a new FlowLayout object with center alignment and horizontal and vertical gap to be default size of 5 pixels.
2. FlowLayout(int align): construct similar object with different settings on alignment.
3. FlowLayout(int align, int hgap, int vgap): construct similar object with different settings on alignment and gaps between components.

(For the constructor with the alignment settings, the possible values could be: LEFT, RIGHT, CENTER, LEADING and TRAILING.)

GridLayout

- The Java GridLayout class is used to arrange the components in a rectangular grid. One component is displayed in each rectangle.

Constructors :

1. **GridLayout():** creates a grid layout with one column per component in a row.
2. **GridLayout(int rows, int columns):** creates a grid layout with the given rows and columns but no gaps between the components.
3. **GridLayout(int rows, int columns, int hgap, int vgap):** creates a grid layout with the given rows and columns along with given horizontal and vertical gaps.

GridBagLayout

- **GridBagLayout** is a more flexible layout manager, which allows the components to be vertical, horizontal, without specifying the components to be the same size.
- GridBagLayout extends the capabilities of the GridLayout.
- GridBagLayout also allows the component to span to multiple columns or rows.
- The **GridBagConstraints** decides where the component to be displayed and how the component should be positioned.

Constructors : GridBagLayout()

Here are properties of ***GridBagConstraints*** and their descriptions:

1. gridx,gridy :
   Specify the row and column from top to bottom and from left to right starting from zero. For example gridx = 0, gridy = 0 is the top left cell of the grid.

2. gridwidth, gridheight :
   Specify number of rows (gridheight) and columns (gridwidth) to which a component can span. The default value of *gridwidth* and *gridheight* is 1.
3. fill :

This property is used to resolve whether and how to resize the component when the component's display region is larger than the component's requested size. The values for fill property are: **NONE**, **VERTICAL**, **HORIZONTAL** and **BOTH**

4. ipadx, ipady :

The ipadx, ipady property are used to set the internal padding of the component.

5. weightx, weighty

These properties are used to specify how to distribute space between rows(weighty) and columns(weightx).

GroupLayout

- **GroupLayout** groups its components and place them hierarchically in a container. The grouping is done by an instance of the Group class.
- Group is an abstract class and two concrete classes that implement this class are **SequentialGroup** and **ParallelGroup**.
- **SequentialGroup** positions its elements sequentially one after the other while **ParallelGroup** aligns its elements on top of each other.
- The **GroupLayout** class provides methods such as **createParallelGroup**() and **createSequentialGroup**() to create groups.
- GroupLayout treats each axis independently. That is, there is a group representing the horizontal axis, and a group representing the vertical axis. Each component must exist in both a horizontal and vertical group.

Constructors : GroupLayout(Container host)

Working with 2D Shapes :

- paintComponent is a method in the JComponent class (which is a superclass of many Swing components such as JPanel) that is called automatically by Swing to paint the component on the screen. When a component needs to be displayed, Swing invokes paintComponent to render the component's appearance on the screen.

- Graphics2D is a subclass of the Graphics class in Java that provides more advanced 2D rendering capabilities than the basic Graphics class.

- It allows you to draw lines, shapes, and images with various effects, such as gradients, alpha transparency, and antialiasing**.**

Applet :

- Applet is a special type of program that is embedded in the webpage to generate the dynamic content. It runs inside the browser and works at client side.
- To create an applet, a class must class extends **java.applet.Applet** class.

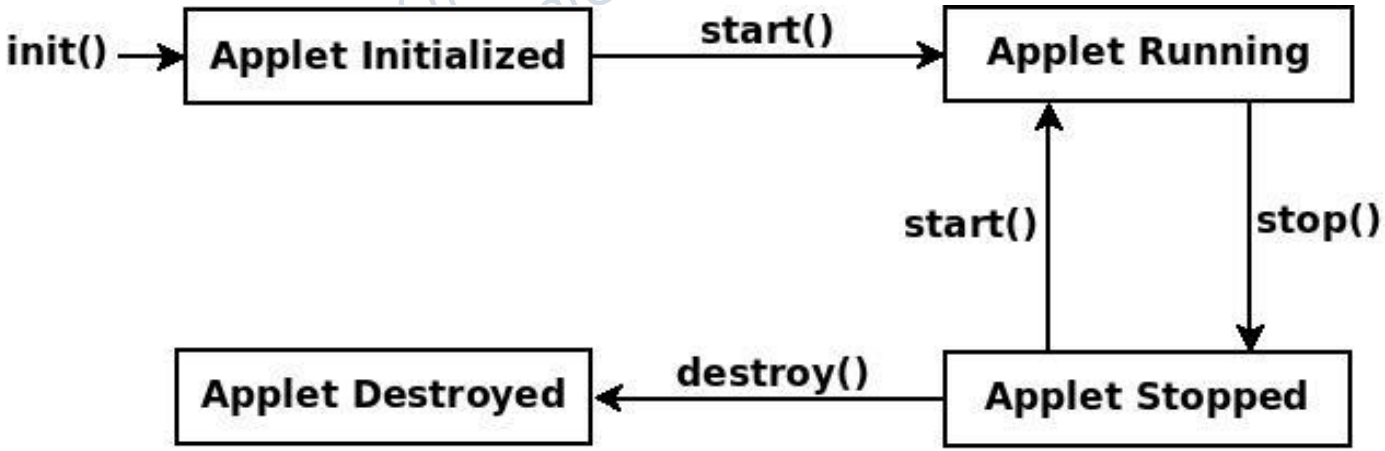
- An Applet class does not have any main() method. It is viewed using JVM. The JVM can use either a plug-in of the Web browser or a separate runtime environment to run an applet application.
- JVM creates an instance of the applet class and invokes init() method to initialize an Applet.

Advantages of Applet :

- They are very secure.
- It works at client side so less response time.
- Applets can be executed by browsers running under different platforms.

Lifecycle of Java Applet :

1. Applet is initialized
2. Applet is running
3. Applet is stopped.
4. Applet is destroyed.



1. **Applet Initialized**:
   Objects regarding Applet get initialized in this phase. An init() method is called for the same purpose.
2. **Applet Running**:
   Applets are embedded in a webpage. When a part of webpage which consists of applet is shown on a screen, then applet is in Running phase. A start() method is called, which takes applet to the Running phase.

3. **Applet Stopped**:
   When a webpage in minimized or webpage is scrolled down/up; so that, applet is disappeared from the screen, then applet is in Stopped phase. Stop() method is called when applet is disappeared from the screen.

4. **Applet Destroyed:**
   When you closed the webpage which consists of applet, applet is get destroyed. A destroy() method is called when applet is destroyed.

Lifecycle methods for Applet:

1. init(): is used to initialized the Applet. It is invoked only once.

2. start(): is invoked after the init() method or browser is maximized. It is used to start the Applet.

3. paint(Graphics g): is used to paint the Applet. It provides Graphics class object that can be used for drawing oval, rectangle, arc etc.

4. stop(): is used to stop the Applet. It is invoked when Applet is stop or browser is minimized.

5. destroy(): is used to destroy the Applet. It is invoked only once.

Run Applet:

There are two ways to run an applet
1.By html file.
2.By appletViewer tool (for testing purpose).

Applet viewer :

- It is a command line program to run Java applets.
- It is included in the SDK.
- It helps you to test an **applet** before you run it in a browser.

Example of Applet with adding controls:

```
import java.applet.Applet;
import java.awt.Button;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;


public class AppletDemo extends Applet implements ActionListener{
Button btn1;
public void init() {
btn1 = new Button("Click Me...");
btn1.addActionListener(this);
add(btn1);
}
public void paint(Graphics g) {
g.drawString("Teaxs International College", 10, 10);
g.drawOval(50, 50, 50, 50);
}
public void actionPerformed(ActionEvent e) {
setBackground(Color.RED);
}
}
```

AppletDemo.html

```
<html>
<body>
<applet code=”AppletDemo.class" width="300" height="300">
</applet>
</body>
</html>
```