# Learning Java - A Foundational Journey

Session: 16

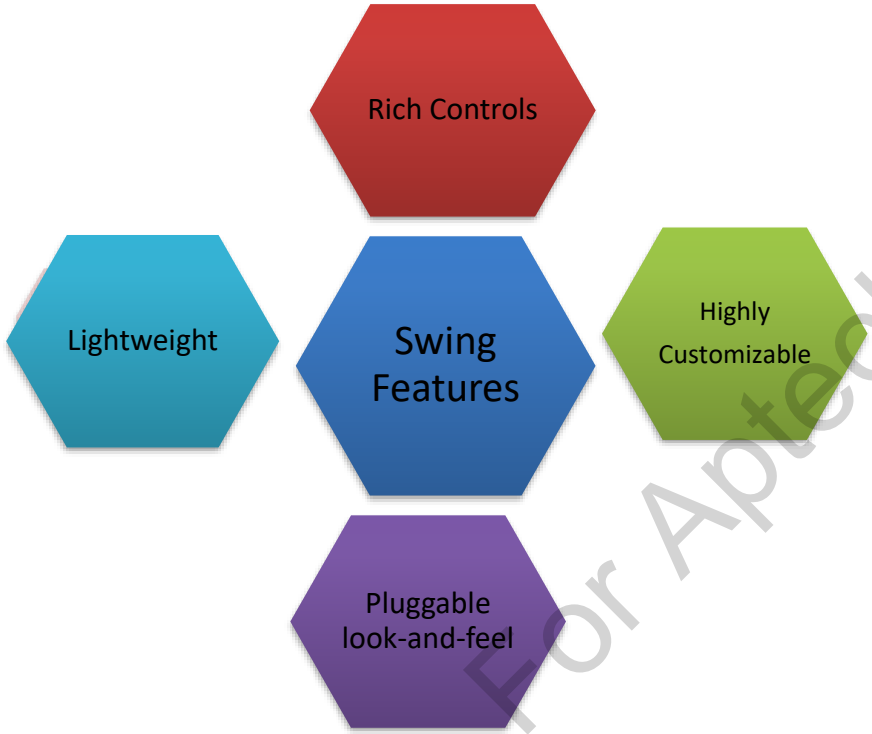

## Swing and JavaFX

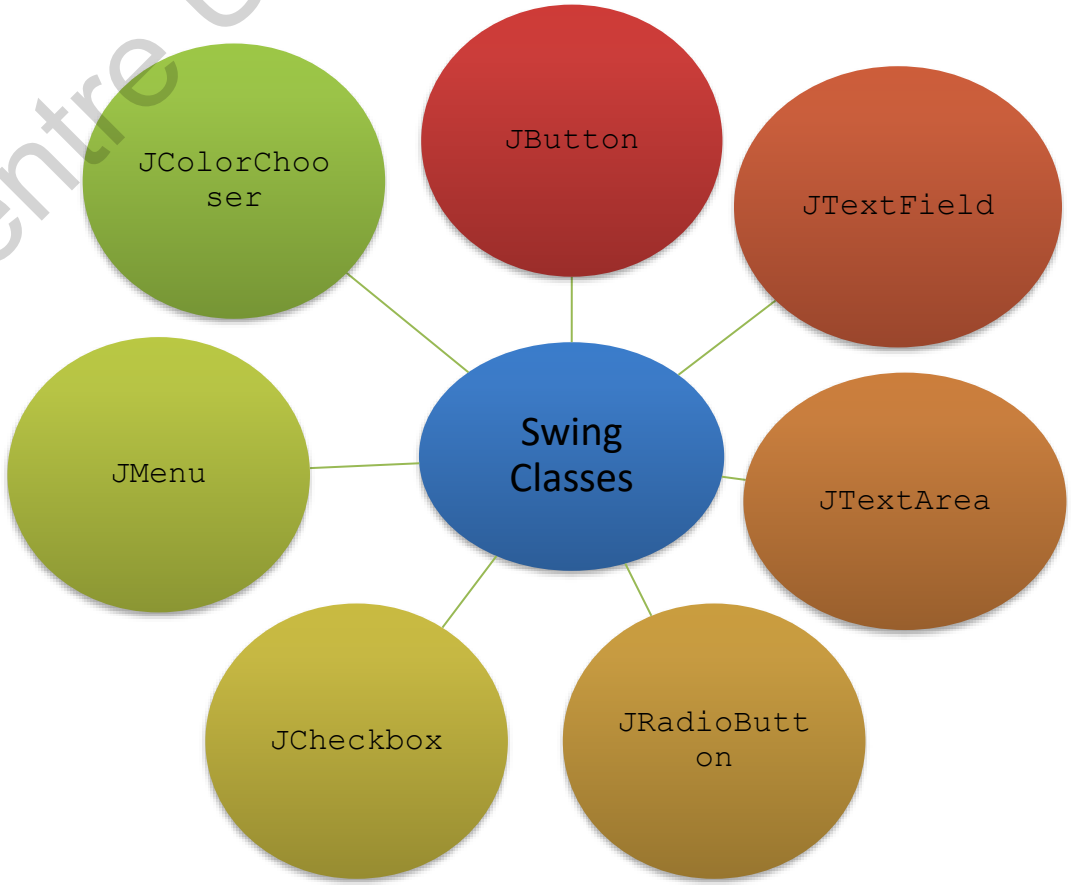
# Objectives

- Explain Swing Components
- Describe Layout Managers
- Describe JavaFX in detail

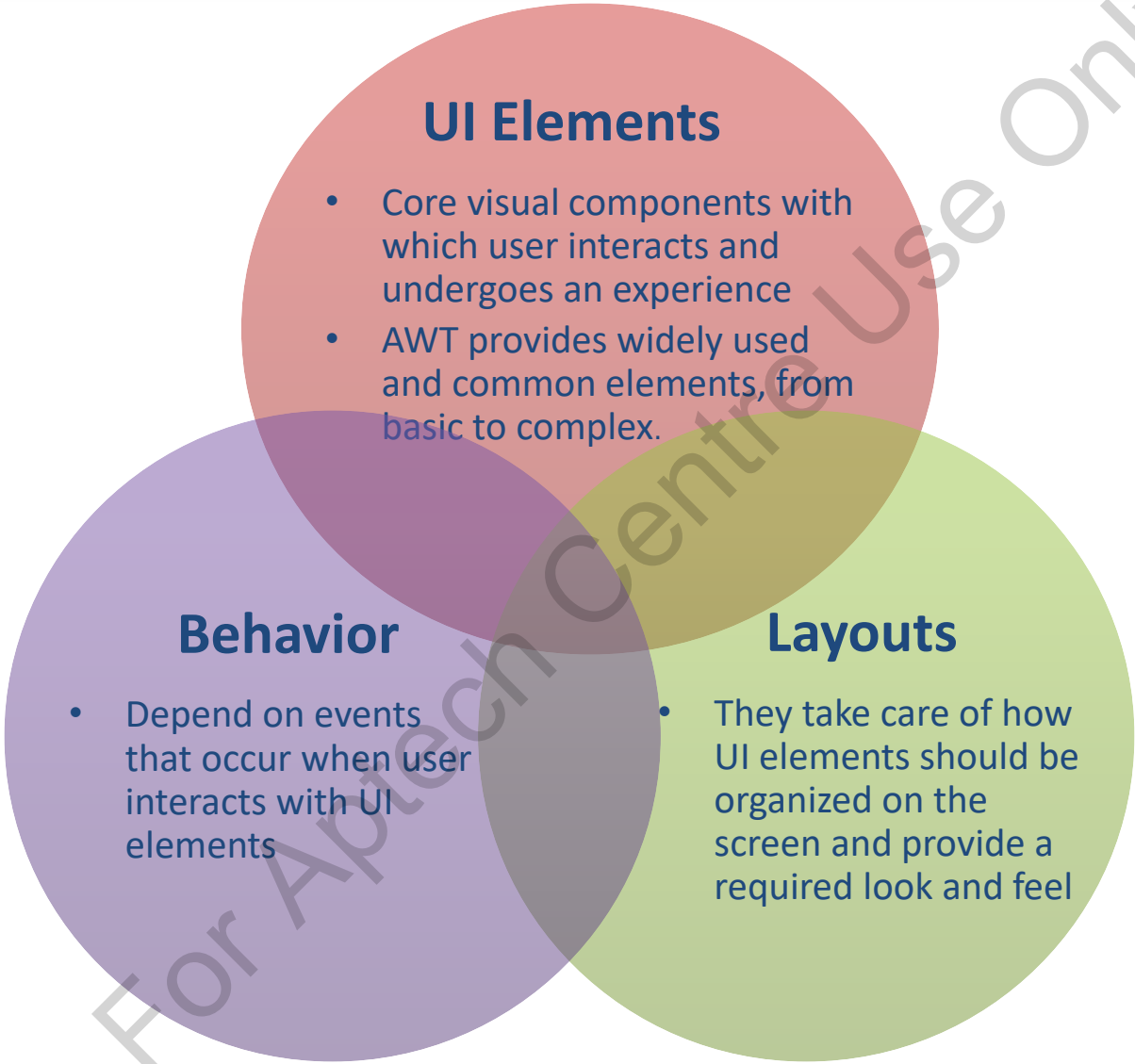
# Introduction to Swing

- Used for creating desktop applications with GUI features
- Completely written in Java

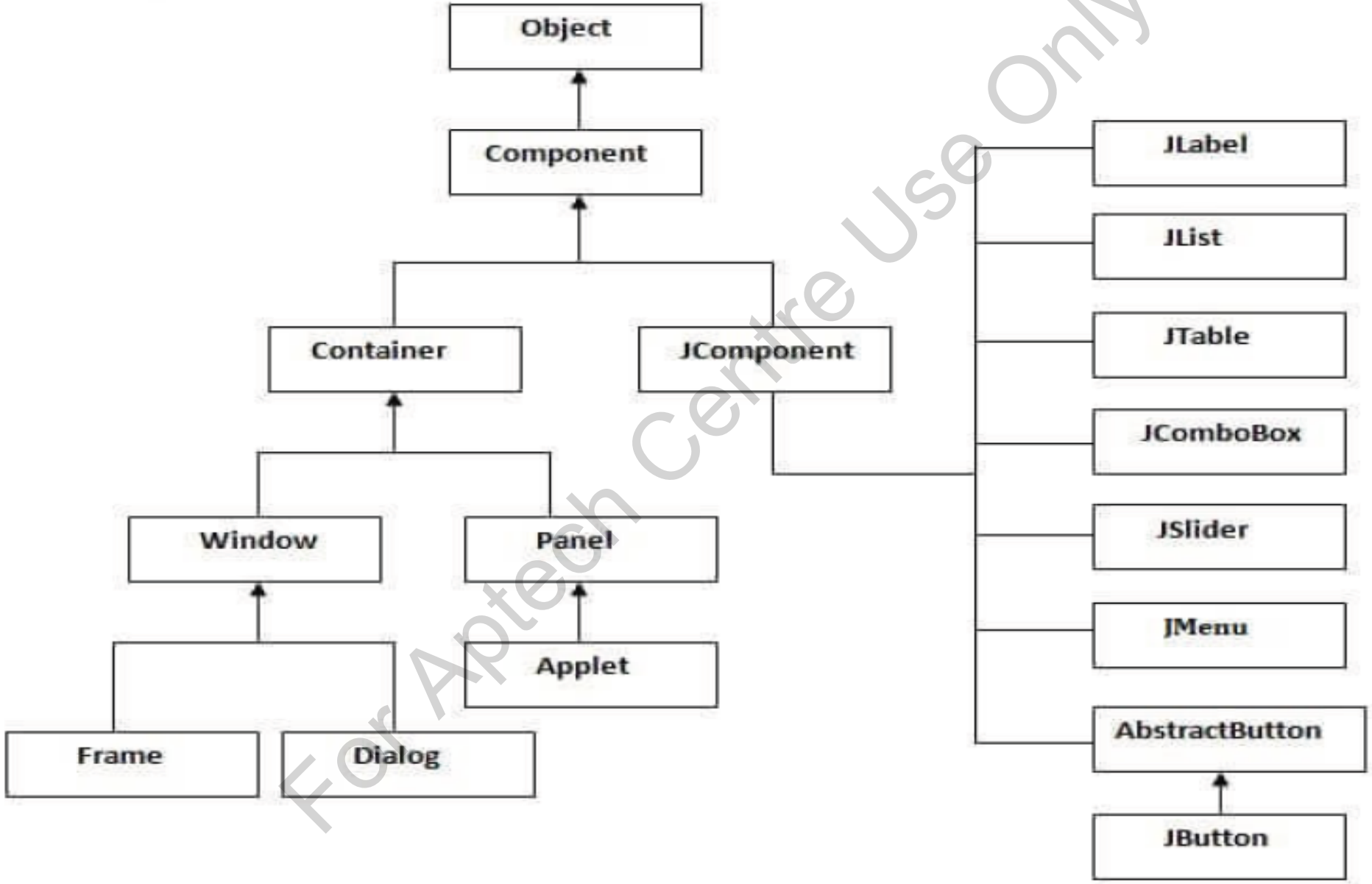


# Swing Controls



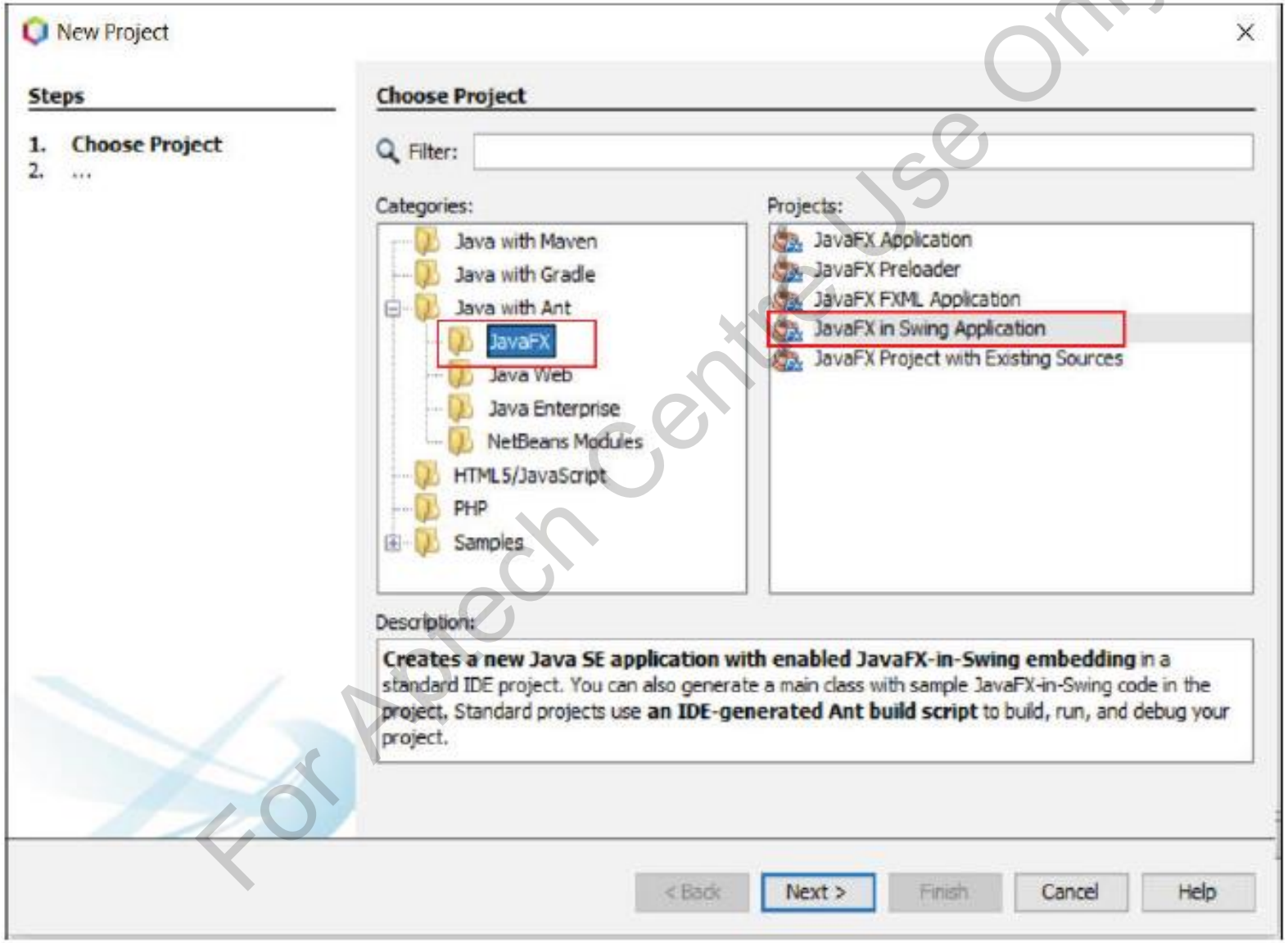
# Hierarchy of Swing Classes 1-2



# Hierarchy of Swing Classes 2-2

| Class | Description |
|---|---|
| Component | It is an abstract base class for non-menu user-interface controls of Swing. It can also represent an object with graphical representation. |
| Container | It can contain other Swing components. |
| JComponent | JComponent is a base class for all Swing UI components. In order to use a Swing component that inherits from JComponent, it must be a part of containment hierarchy and its root should be top-level Swing container. |

# Swing UI Elements 1-5



# Swing UI Elements 2-5

**JLabel:**

- Can display text, image, or both
- Label contents can be aligned by setting the vertical and horizontal alignment in its display area
- Text-only labels are leading edge aligned, whereas image-only labels are horizontally center aligned

```
package com.aptech.gui;
import javax.swing.*;
class JLabelDemo {
public static void main(String args[]) {
JFrame frame= new JFrame("Swing JLabel Example");
JLabel label1,label2;
label1=new JLabel("JLabel Example",JLabel.CENTER);
label1.setBounds(50,50, 250,30);
label2=new JLabel("Welcome to Aptech Jlabel
Example",JLabel.CENTER);
label2.setBounds(50,100, 250,30);
frame.add(label1);
frame.add(label2);
frame.setSize(400,400);
frame.setLayout(null);
frame.setVisible(true);
}
}
```

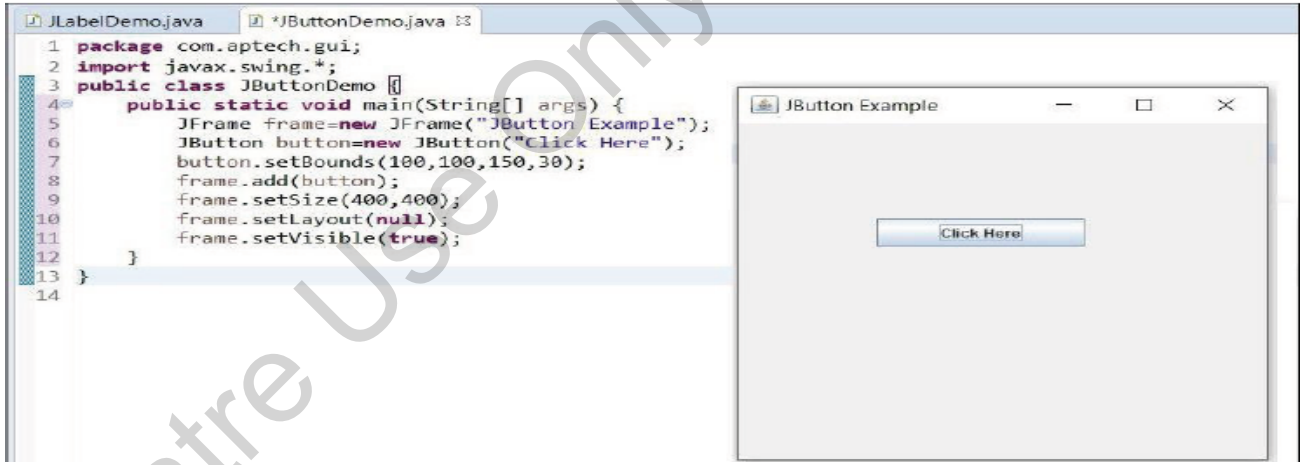
JLabelDemo.java

```
1  package com.aptech.gui;
2  import javax.swing.*;
3  class JLabelDemo
4  {
5  public static void main(String args[])
6          {
7              JFrame frame= new JFrame("Swing JLabel Example");
8              JLabel label1,label2;
9              label1=new JLabel("JLabel Example",JLabel.CENTER);
10             label1.setBounds(50,50, 250,30);
11             label2=new JLabel("Welcome to Aptech JLabel Example",JLabel.CENTER);
12             label2.setBounds(50,100, 250,30);
13             frame.add(label1);
14 frame.add(label2);
15             frame.setSize(400,400);
16             frame.setLayout(null);
17             frame.setVisible(true);
18         }
19 }
20
```

# Swing UI Elements 3-5

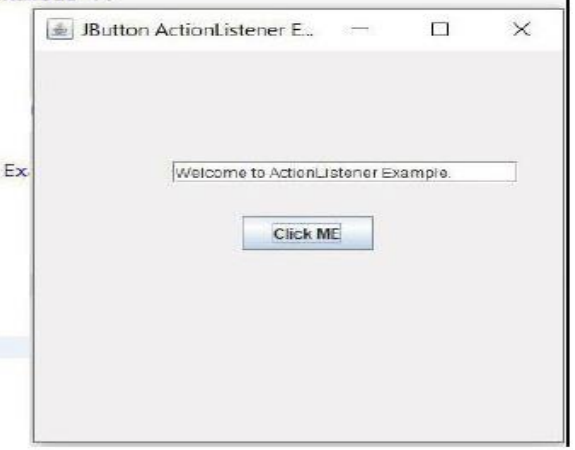
## JButton:

- Is an implementation of a push button.
- It has a label and generates an event when user clicks it.
- It can also have an image and inherits AbstractButton class, which implements the Accessible interface.




```
package com.aptech.gui;
import javax.swing.*;
public class JButtonDemo {
    public static void main(String[] args) {
      JFrame frame=new JFrame("JButton Example");
      JButton button=new JButton("Click Here");
      button.setBounds(100,100,150,30);
      frame.add(button);
      frame.setSize(400,400);
      frame.setLayout(null);
      frame.setVisible(true);
   }
}
```

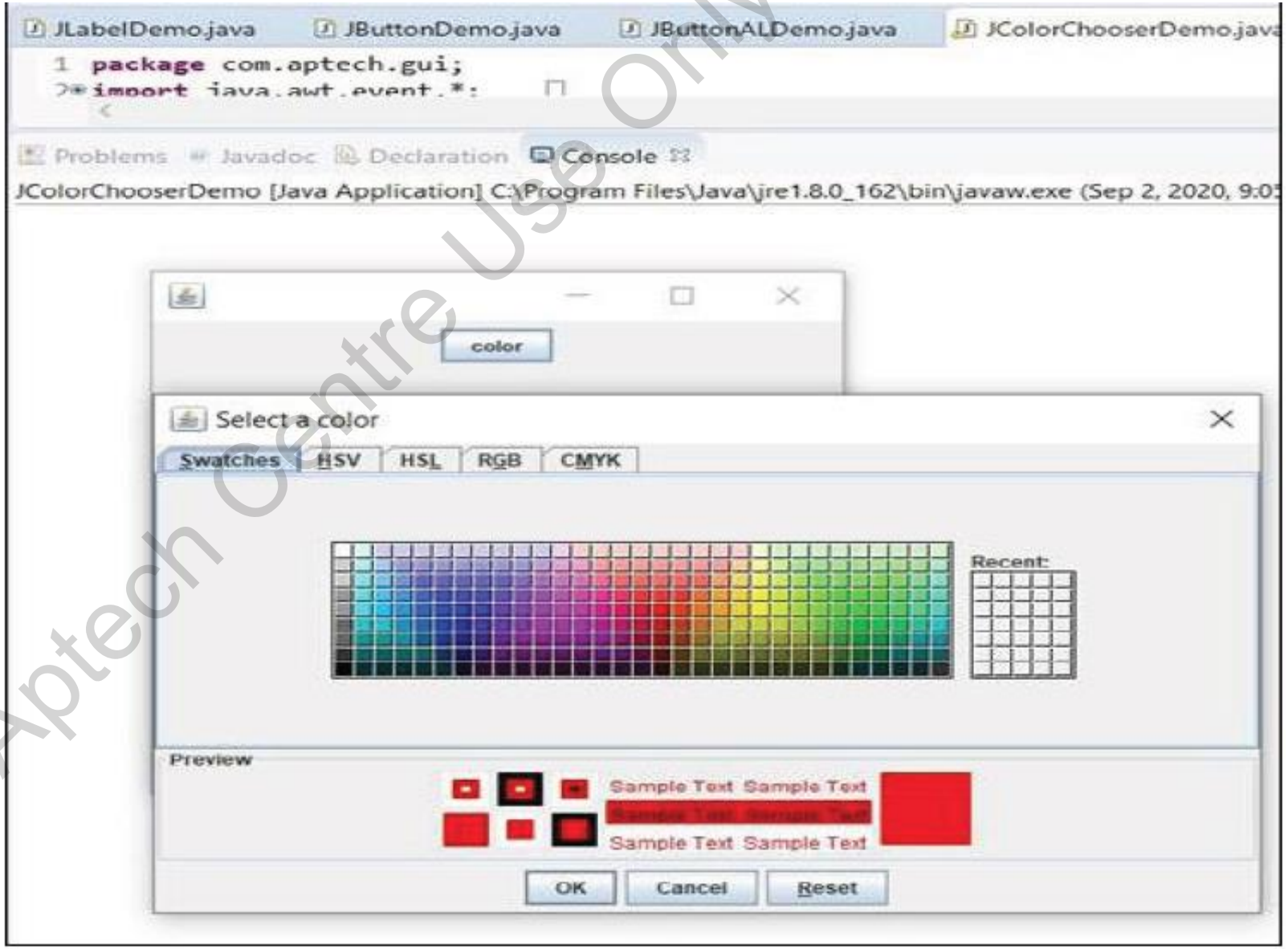
```
package com.aptech.gui;

import java.awt.event.*;
import javax.swing.*;
public class JButtonALDemo {
    public static void main(String[] args)
    {
        JFrame frame=new JFrame("JButton ActionListener Example");
        final JTextField textfield=new JTextField();
        textfield.setBounds(100,100, 250,20);
        JButton button=new JButton("Click ME");
        button.setBounds(150,150,95,30);
        button.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e){
            textfield.setText("Welcome to ActionListener Ex
        });
        frame.add(button);
        frame.add(textfield);
        frame.setSize(400,400);
        frame.setLayout(null);
        frame.setVisible(true);
    }
}
```



# Swing UI Elements 4-5

**JColorChooser:**

- It provides a pane of controls designed to allow a user to manipulate and select a color.
- `JColorChooser()` creates a color chooser pane with an initial color of white.



# Swing UI Elements 5-5

| Classes | Description |
|---|---|
| JCheckBox | It is a graphical component that can be in either an on (true) or off (false) state. Present in a group. |
| JRadioButton | It is a graphical component that can be in either an on (true) or off (false) state. Present in a group. |
| JList | JList component presents the user with a scrolling list of text items. |
| JComboBox | JComboBox component presents the user with a drop-down to show a menu of choices. |
| JTextField | JTextField object is a text component that allows editing of a single line of text. |
| JPasswordField | A JPasswordField object is a text component specialized for password entry. |
| JTextArea | A JTextArea object is a text component that allows editing of multiple lines of text. |
| ImageIcon | An ImageIcon control is an implementation of the Icon interface that paints icons from Images. |
| JScrollbar | Scrollbar control represents a scroll bar component in order to enable the user to select from range of values. |
| JOptionPane | JOptionPane provides set of standard dialog boxes that prompt users for a value or informs them of something. |
| JFileChooser | JFileChooser control represents a dialog window from which the user can select a file. |
| JProgressBar | JProgressBar represents a the progress bar displays the tasks percentage of completion, as the task progresses towards completion. |
| JSlider | JSlider lets the user graphically select a value by sliding a knob within a bounded interval. |
| JSpinner | JSpinner is a single line input field that lets the user select a number or an object value from an ordered sequence. |

# Layout Managers 1-2

A Layout manager helps to arrange position for all components within a container

A layout manager adapt to the dimensions of application window

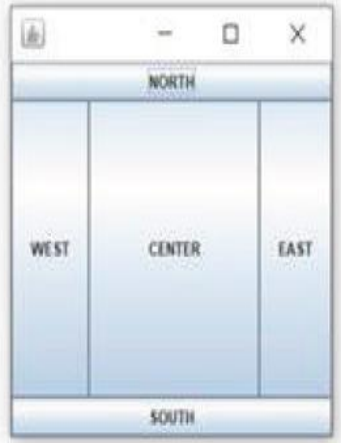
Each layout manager is an object of the class that implements the `LayoutManager` interface

| Layout Manager | Description |
|---|---|
| `BorderLayout` | It arranges components to fit in the five regions such as east, west, north, south, and center. |
| `CardLayout` | It treats each component in container as a card and only one card is visible at a time. |
| `FlowLayout` | It is the default layout and it arranges components in a directional flow. |
| `GridLayout` | It manages the components in the form of a rectangular grid. |
| `GridBagLayout` | This is the most flexible layout manager class, it aligns the component vertically, horizontally, or along their baseline having different sizes. |
| `GroupLayout` | It hierarchically groups the components in order to position them in a container. |
| `SpringLayout` | It positions children of its associated container according to a set of constraints. |

# Layout Managers 2-2

```
package com.aptech.gui; import
java.awt.*; import
javax.swing.*; public class
BorderLayoutDemo {
  JFrame frame;
  BorderLayoutDemo(){ frame=new JFrame();
      JButton button1=new
      JButton("NORTH");
      JButton button2=new JButton("SOUTH");
      JButton button3=new JButton("EAST");
      JButton button4=new JButton("WEST");
      JButton button5=new JButton("CENTER");
      frame.add(button1,BorderLayout.NORTH);
      frame.add(button2,BorderLayout.SOUTH);
      frame.add(button3,BorderLayout.EAST);
      frame.add(button4,BorderLayout.WEST);
      frame.add(button5,BorderLayout.CENTER);
      frame.setSize(300,300);
      frame.setVisible(true);
  } public static void main(String[]
  args) { new BorderLayoutDemo();
```

```
1  package com.aptech.gui;
2  import java.awt.*;
3  import javax.swing.*;
4
5  public class BorderLayoutDemo {
6
7      JFrame frame;
8      BorderLayoutDemo(){
9          frame=new JFrame();
10
11         JButton button1=new JButton("NORTH");;
12         JButton button2=new JButton("SOUTH");;
13         JButton button3=new JButton("EAST");;
14         JButton button4=new JButton("WEST");;
15         JButton button5=new JButton("CENTER");;
16
17         frame.add(button1,BorderLayout.NORTH);
18         frame.add(button2,BorderLayout.SOUTH);
19         frame.add(button3,BorderLayout.EAST);
20         frame.add(button4,BorderLayout.WEST);
21         frame.add(button5,BorderLayout.CENTER);
22
23         frame.setSize(300,300);
24         frame.setVisible(true);
25     }
26     public static void main(String[] args) {
27         new BorderLayoutDemo();
28     }
29
30 }
```



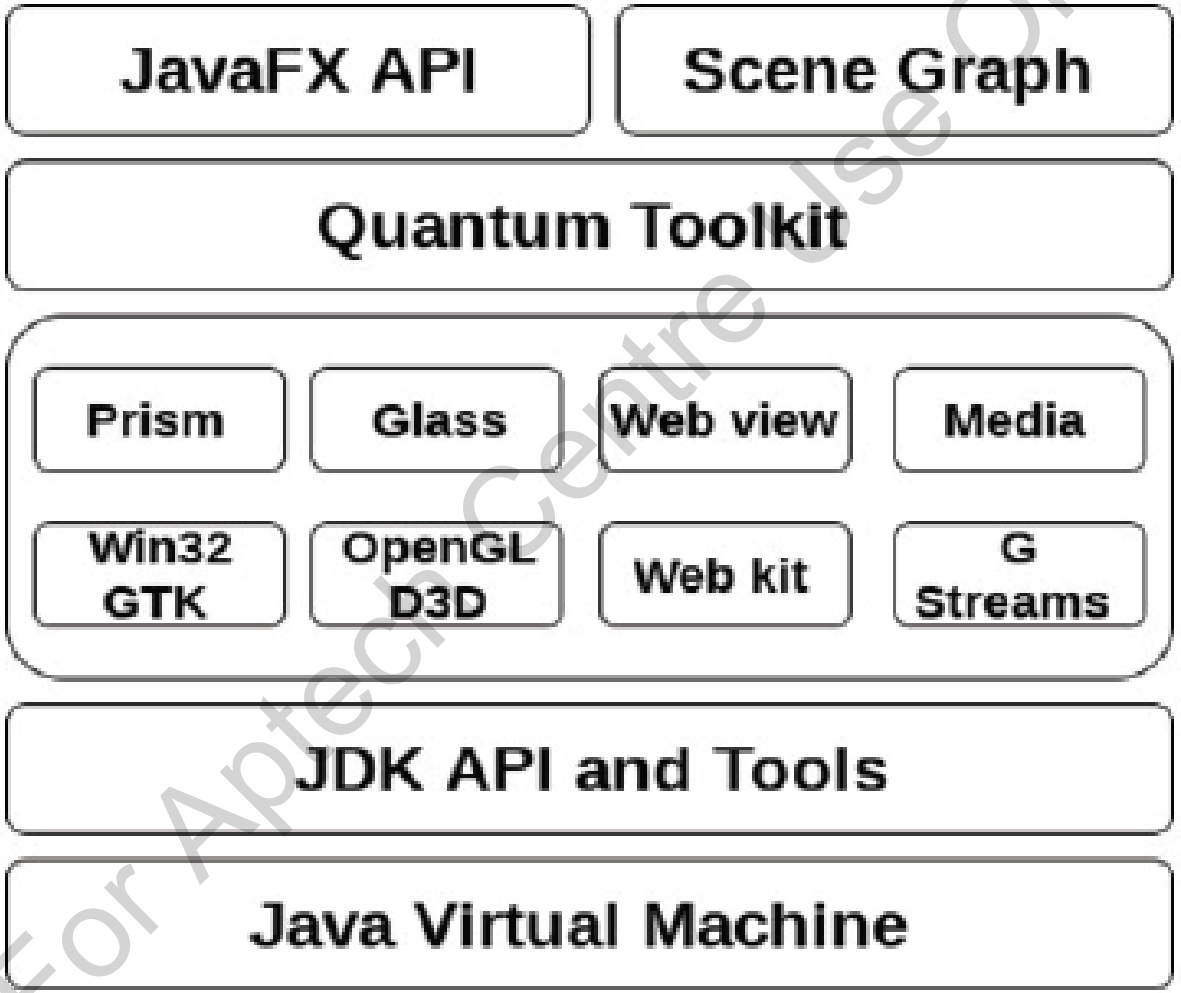
# JavaFX Packages 1-2

| Package Name | Description |
|---|---|
| javafx.animation | Provides set of classes that are responsible for transitions based animations |
| javafx.application | Provides application life-cycle methods |
| javafx.collections | Provides classes that can handle collections and related utilities |
| javafx.concurrent | Provides classes that are responsible for multitasking |
| javafx.embed.swing | Provides set of classes that can be used inside Swing code |
| javafx.embed.swt | Provides set of classes that can be used inside the Standard Widget Toolkit (SWT) code |
| javafx.event | Provides classes that deal with events and their handling |
| javafx.fxml | Contains set of classes that are responsible of loading hierarchy from markup |
| javafx.geometry | Provides 2D classes that contains methods to operate 2D geometry on the object |
| javafx.scene | Provides classes to deal with scene graph API |
| javafx.scene.canvas | Provides set of classes that deal with canvas |
| javafx.scene.control | Contains classes for all JavaFX components |
| javafx.scene.effect | Contains set of classes that apply the graphic effects to scene graph nodes |
| javafx.scene.image | Provides set of classes for loading and displaying images |

# JavaFX Packages 2-2

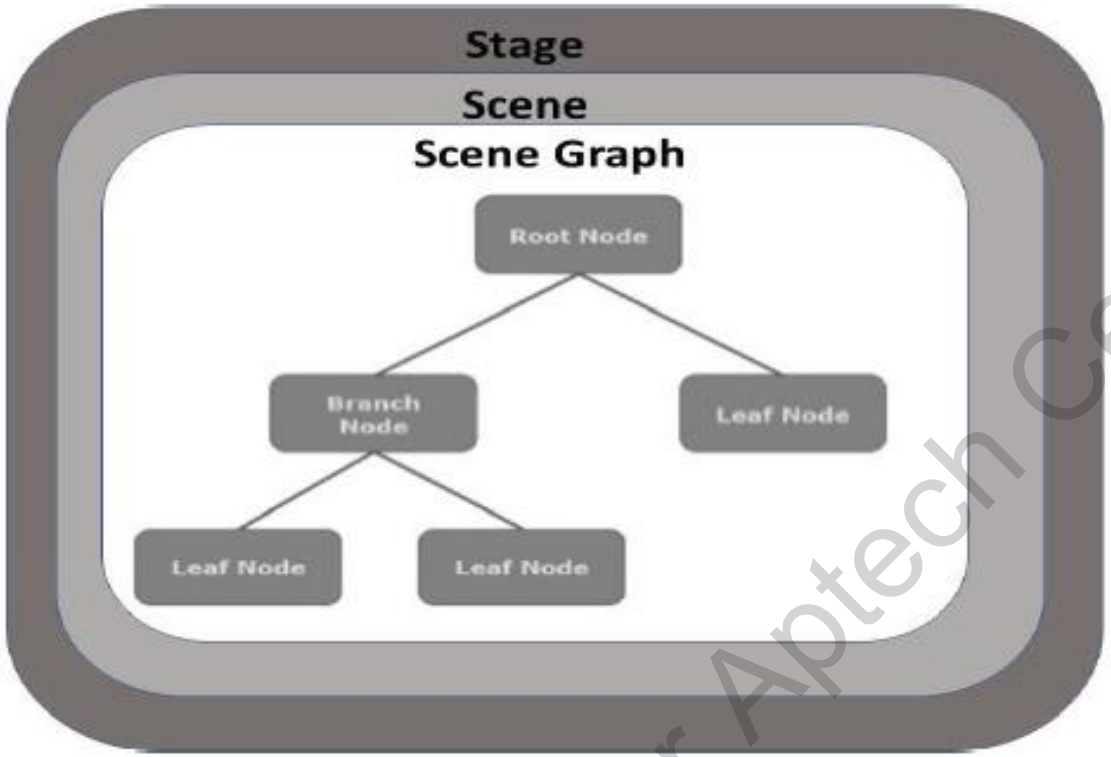
| Package Name | Description |
|---|---|
| javafx.scene.input | Provides set of classes for the mouse and keyboard events |
| javafx.scene.layout | Provides set of classes to support user interface layout |
| javafx.scene.shape | Provides set of 2D classes that performs the operations on objects related to 2D geometry |
| javafx.scene.text | Provides set of classes for fonts and rendering text nodes |
| javafx.scene.transform | Provides set of classes that are used to perform rotating, scaling, and shearing operations on objects |
| javafx.scene.web | Provides means for loading and displaying Web content |
| javafx.stage | Provides top level container classes for JavaFX content |
| javafx.util | Provides utilities classes |
| javafx.util.converter | Provides standard string converters for JavaFX |

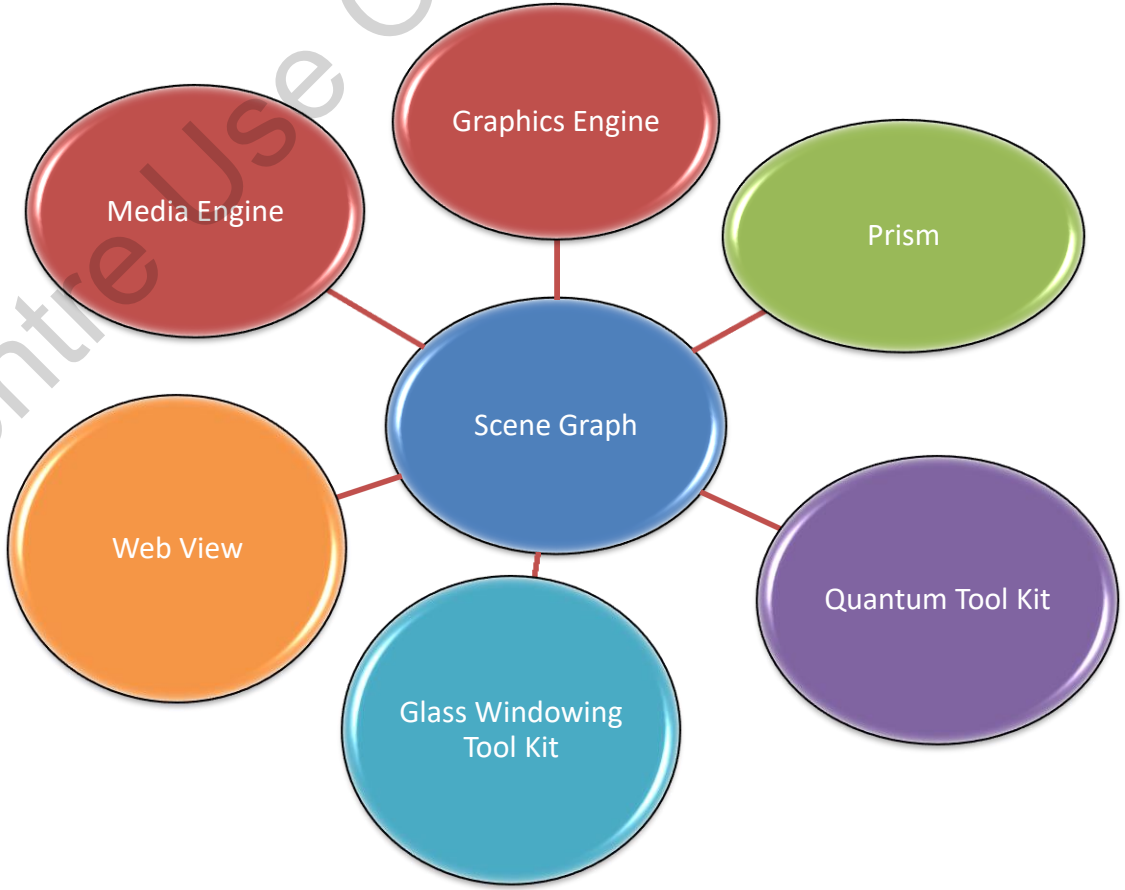
# Components of JavaFX Architecture 1-2



# Components of JavaFX Architecture 2-2

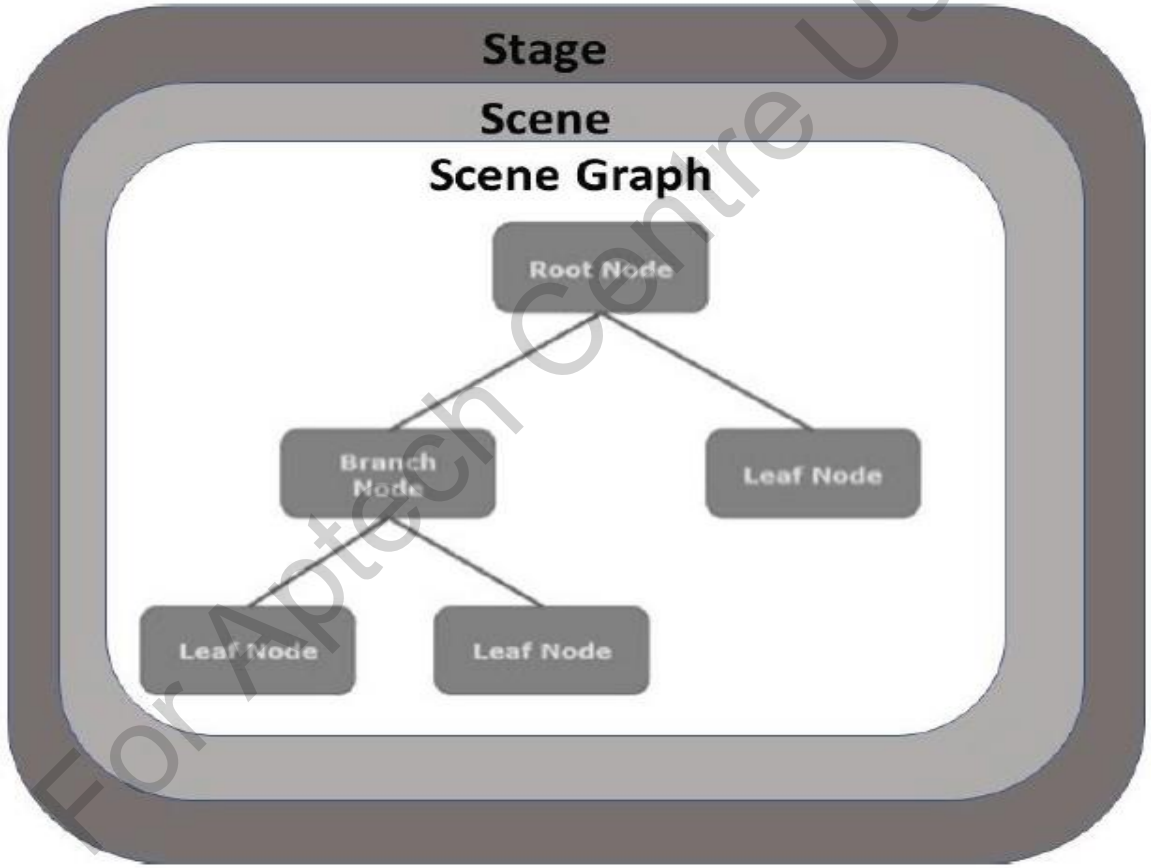
**Scene Graph**



# JavaFX Application Structure 1-3

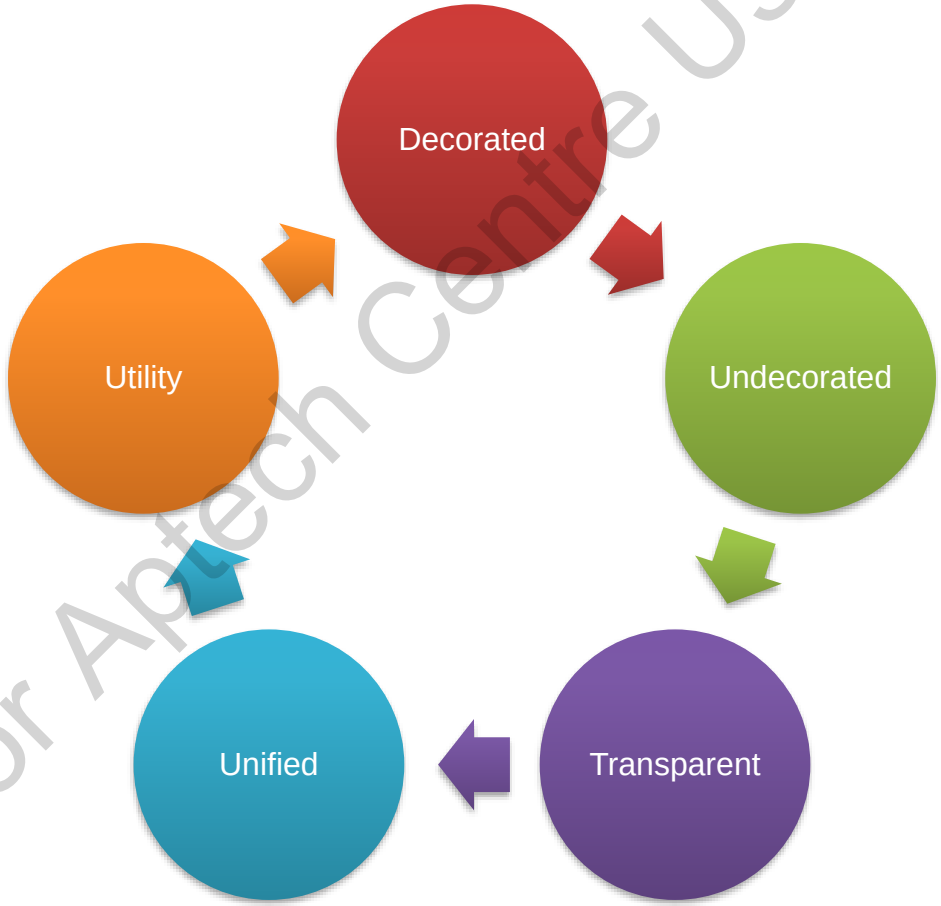
JavaFX application has majorly three components Stage, Scene, and Nodes



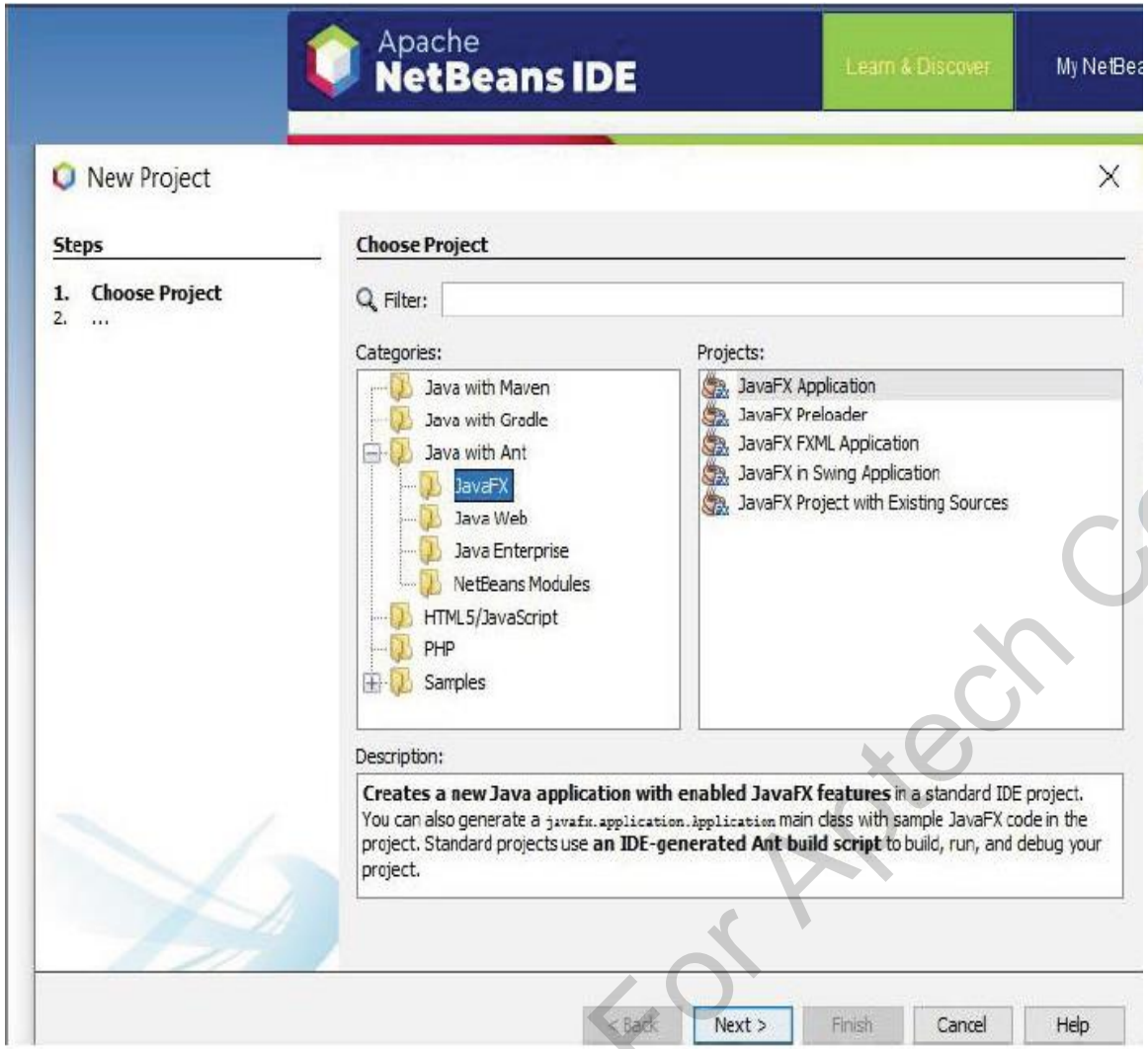
# JavaFX Application Structure 2-3

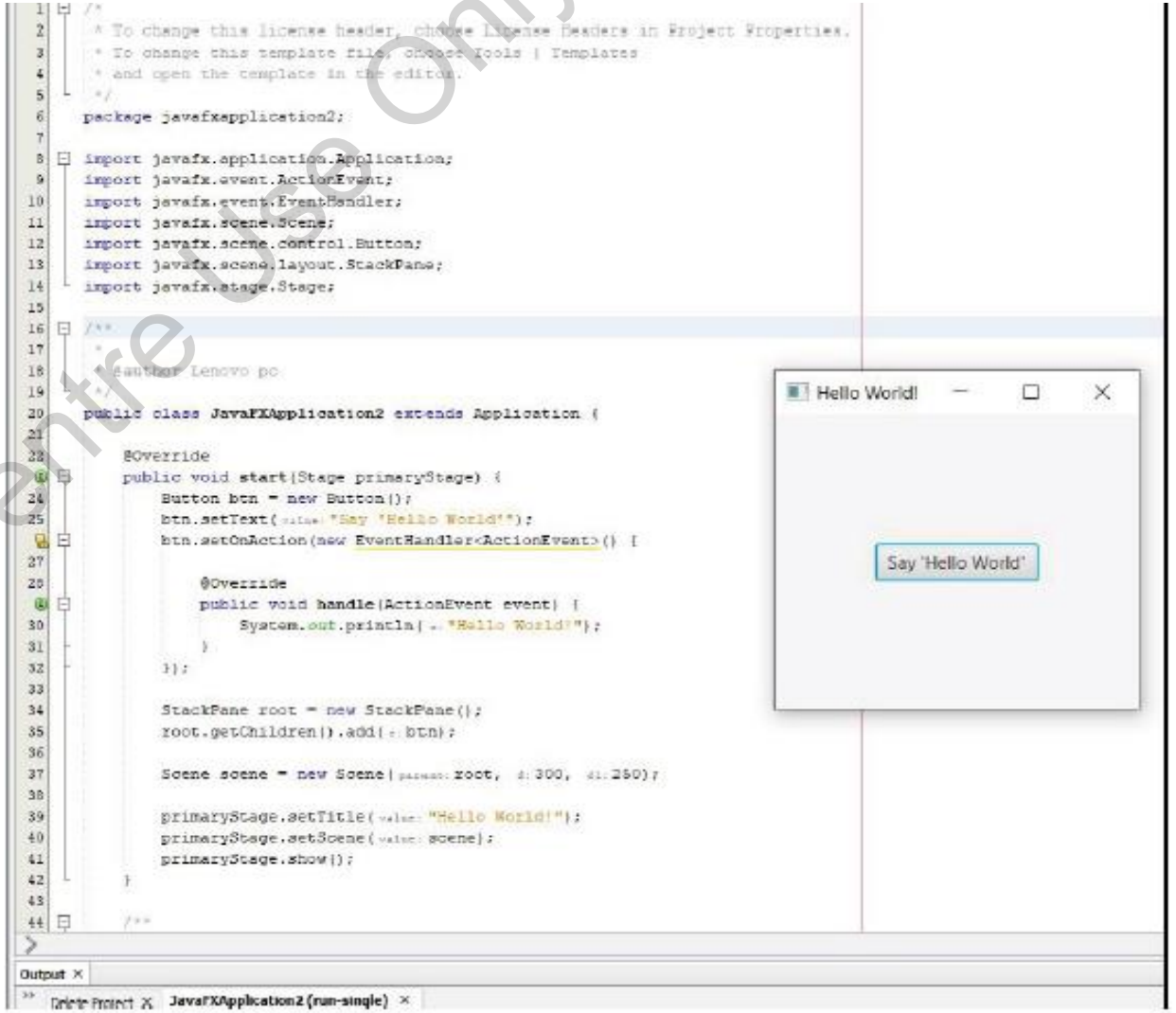
## Stage

Stage has two parameters determining its position namely, Width and Height.



# JavaFX Application Structure 3-3



# JavaFX 2D Shapes 1-3

JavaFX provides the flexibility to create 2D shapes with customized specifications.

All classes that implement 2D shapes are part of javafx.scene.shape package.

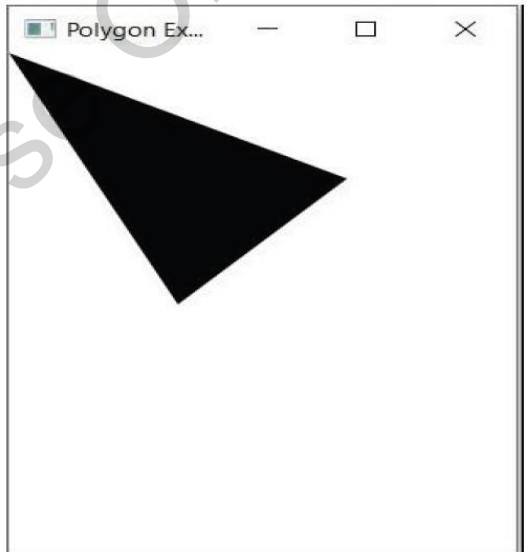
Methods inside these classes deal with coordinates of 2D shape creation.

# JavaFX 2D Shapes 2-3

| Shape | Description |
|---|---|
| Line | It is a geometrical shape, which connects two points on 2D coordinate system. To create a line in JavaFX application, `javafx.scene.shape.Line` class requires to be instantiated. |
| Rectangle | It is a geometrical shape with two pairs of two equal sides and four right angles at their joint. To create a rectangle in JavaFX application, `javafx.scene.shape.Rectangle` class requires to be instantiated. |
| Ellipse | Ellipse can be defined as a curve with two focal points. To create it in JavaFX application, `javafx.scene.shape.Ellipse` class requires to be instantiated. |
| Arc | Arc can be defined as part of circumference of the circle of ellipse. In JavaFX application, `javafx.scene.shape.Arc` class requires to be instantiated to create Arcs. |
| Circle | Circle is the special type of Ellipse having both the focal points at the same location. In JavaFX application, a circle can be created by instantiating `javafx.scene.shape.Circle` class. |
| Polygon | It is a geometrical shape that can be created by joining the multiple Co-planner line segments. In JavaFX application, `javafx.scene.shape.Polygon` class requires to be instantiated in order to create a polygon. |
| Cubic Curve | It is a curve of degree three in the XY plane. In JavaFX application, `javafx.scene.shape.CubicCurve` class requires to be instantiated in order to create Cubic Curves. |
| Quad Curve | It is a curve of degree two in the XY plane. In JavaFX application, `javafx.scene.shape.QuadCurve` class requires to be instantiated in order to create a QuadCurve. |

# JavaFX 2D Shapes 3-3

```
package javafxapplication2; import
javafx.application.Application; import
javafx.event.ActionEvent; import
javafx.event.EventHandler; import
javafx.scene.Group; import javafx.scene.Scene;
import javafx.scene.control.Button; import
javafx.scene.layout.StackPane; import
javafx.stage.Stage; import
javafx.scene.shape.Polygon; public class
JavaFXApplication2 extends Application {
    @Override   public void start(Stage
   primarystage) {

      Group root = new Group();
      primarystage.setTitle("Polygon Example");
       Polygon polygon = new Polygon();
     polygon.getPoints().addAll(new Double[]{
     0.0, 0.0,
      100.0, 200.0,
     200.0, 100.0 });
      root.getChildren().add(polygon);
     Scene scene = new Scene(root,300,400);
     primarystage.setScene(scene);
     primarystage.show();
    }   public static void main(String[]
   args) {  launch(args);
    }
}
```




Similarly, JavaFX applications can use Text, Effects, Transformation, Animation, 3D Shapes, Layouts, and so on to create interactive GUI applications.

# Summary

- Swing is a part of Java Foundation Classes (JFC) that is used to create desktop applications.
- JButton class is used to create a labeled button.
- JColorChooser class is used to create a color chooser dialog box.
- ActionListeners perform event handling based on actions done on GUI.
- Layout Managers are useful to arrange elements on GUI at specific position or flow.
- JavaFX is a library designed to help developers to create Rich Internet and Desktop GUI applications.
- Stage is a window that contains all objects of a JavaFX application.