Building Components using Swing and JavaFX

Prativa Nyaupane

Recap

- Frames
- Event handling and Listener Interfaces
- Handling Action Events

Today’s Objectives

- JavaFX vs Swing
- JavaFX Layouts
- JavaFX UI Controls

JavaFX

- JavaFX was created as a successor to Swing.
- JavaFX is Java’s next-generation client platform and GUI framework. JavaFX provides a powerful, streamlined, flexible framework that simpleifies the creation of moder, visually exciting GUIs.
- It provides a modern, hardware-accelerated graphics pipeline(graphical operations are offloaded to dedicated hardware components, such as GPUs to accelerate rendering - animations) for improved performance.
- JavaFX Supports animations and transitions to create visually engaging user interfaces.
- JavaFX applications can be deployed as standalone desktop applications or as applets within web browsers.

JavaFX History

- Original GUI- Abstract Window Toolkit(AWT)
- Swing was added to the platform in Java SE 1.2. Since then, Swing was the primary GUI technology. Swing is now in maintenance mode - Oracle has stopped development and will provide only bug fixes.
- Java's GUI, graphics and multimedia API of the future is JavaFX.
- Sun Microsystems(acquired by Oracle) announced JavaFX in 2007 as a competitor to Adobe Flash and Microsoft Silverlight.
- JavaFX is implemented as a set of Java libraries and could be used directly in Java apps.

Benefits of JavaFX over Swing

- **JavaFX and Swing vary on the organization of the framework and relationship of the main components** - JavaFX offers a more streamlined, easier-to-use, updated approach. JavaFX also greatly simplifies the rendering of objects because it handles repainting automatically.
- **Customization Flexibility-** Though Swing components could be customized, JavaFX gives us complete control over a JavaFX GUI's look and feel via CSS.
- JavaFX graphics rendering takes advantage of **hardware-accelarated capabilities**. The makes rendering graphics and animation perform well.

Benefits of JavaFX over Swing Contd..

- **Thread Safety-** JavaFX was designed for improved thread safety, which is important for today’s multi-core systems.
- **Supports transformation and animations-** JavaFX supports transformations for repositioning and reorienting JavaFX components and animations for changing the properties of JavaFX components over time.
- Hence, JavaFX **facilitates a more visually dynamic approach** to GUIs.

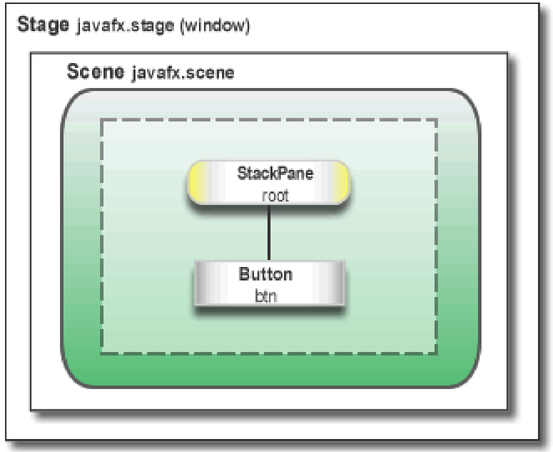
Cons of Swing: Requires manual coding to design UI. May not offer the most modern design elements and graphics effects. Performance can be slower than more modern GUI frameworks.

JavaFX App Window Structure

Stage - The window in which a JavaFX app's GUI is displayed. It is an instance of class Stage(package javafx.stage).

The central metaphor implemented by JavaFX is the stage. As in the case of an actual stage play, a stage contains a scene.

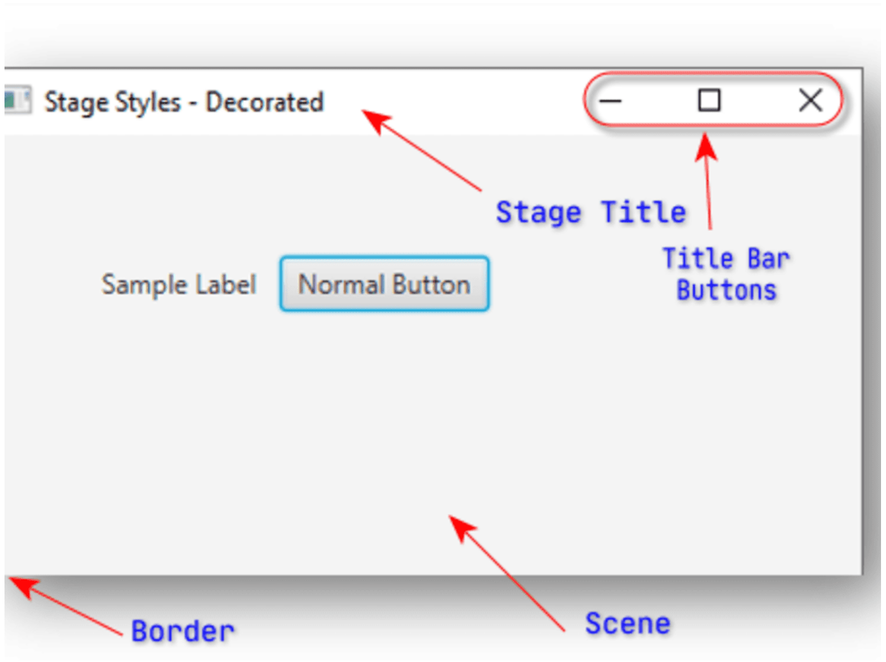
Thus, loosely speaking, a stage defines a space and a scene defines what goes in that space. It is a top-level container.



JavaFX App Window Structure Contd..

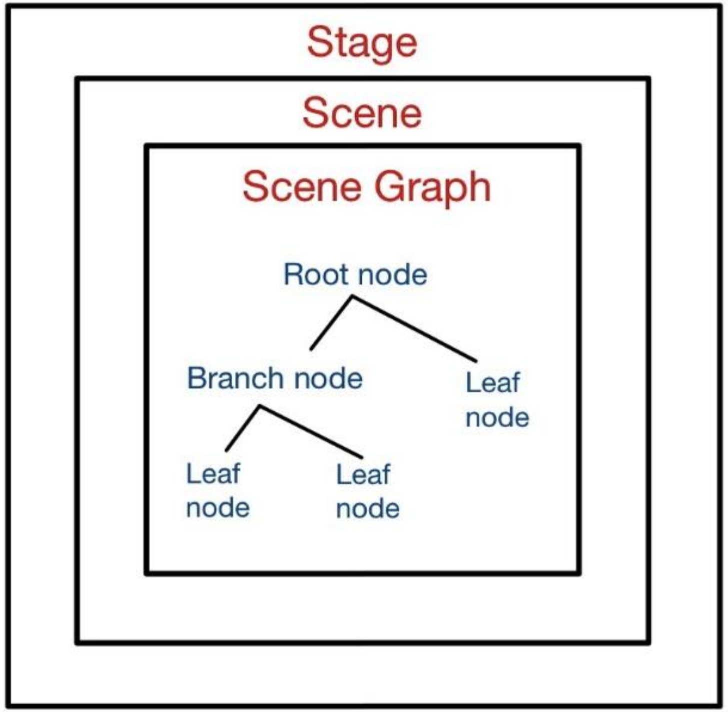
Scene -Scene is a container for the items that comprise the scene. The items consists of controls, such as button, check boxes, text and graphics.To create JavaFX application, we will have to add at least one Scene object to a Stage. Scene is instance of class Scene(p - javafx.scene)

```
Scene scene = new Scene(root, 300,
100);
stage.setScene(scene);
```



Node

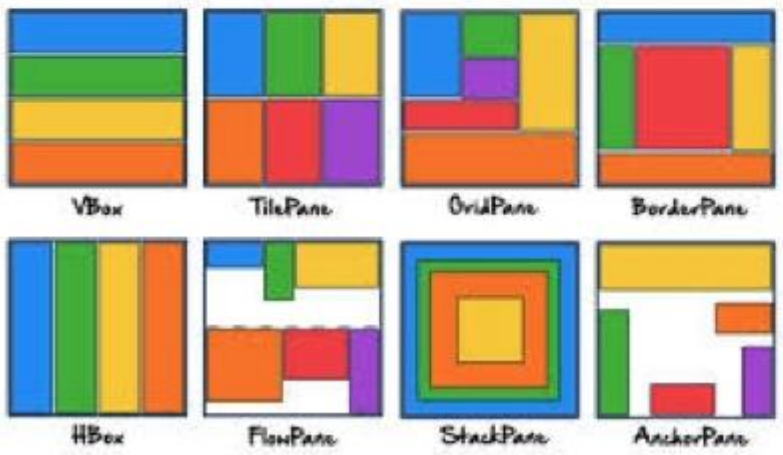
- The individual elements of a scene are called nodes. For eg, a button is a node. Nodes can also consist of groups of nodes.
- A node can have a child node. This is called a parent/branch node.
- First node is root node. Except root nodes, all nodes have parent nodes.
- Nodes without children are terminal nodes and are called leaves.
- The collection of nodes in a scene creates a scene graph, which comprises a tree.



Layout Containers

- JavaFX provides several layout panes that manage the process of placing elements in a sene. After constructing all the required nodes in a scene, we generally arrange them in the desired order. The container in which we arrange the components is called the **Layout** of the container.

  Predefined layouts in JavaFX:
  - HBox
  - VBox
  - Border Pane
  - Stack Pane
  - Anchor Pane
  - Title Pane
  - Grid Pane
  - Flow Panel

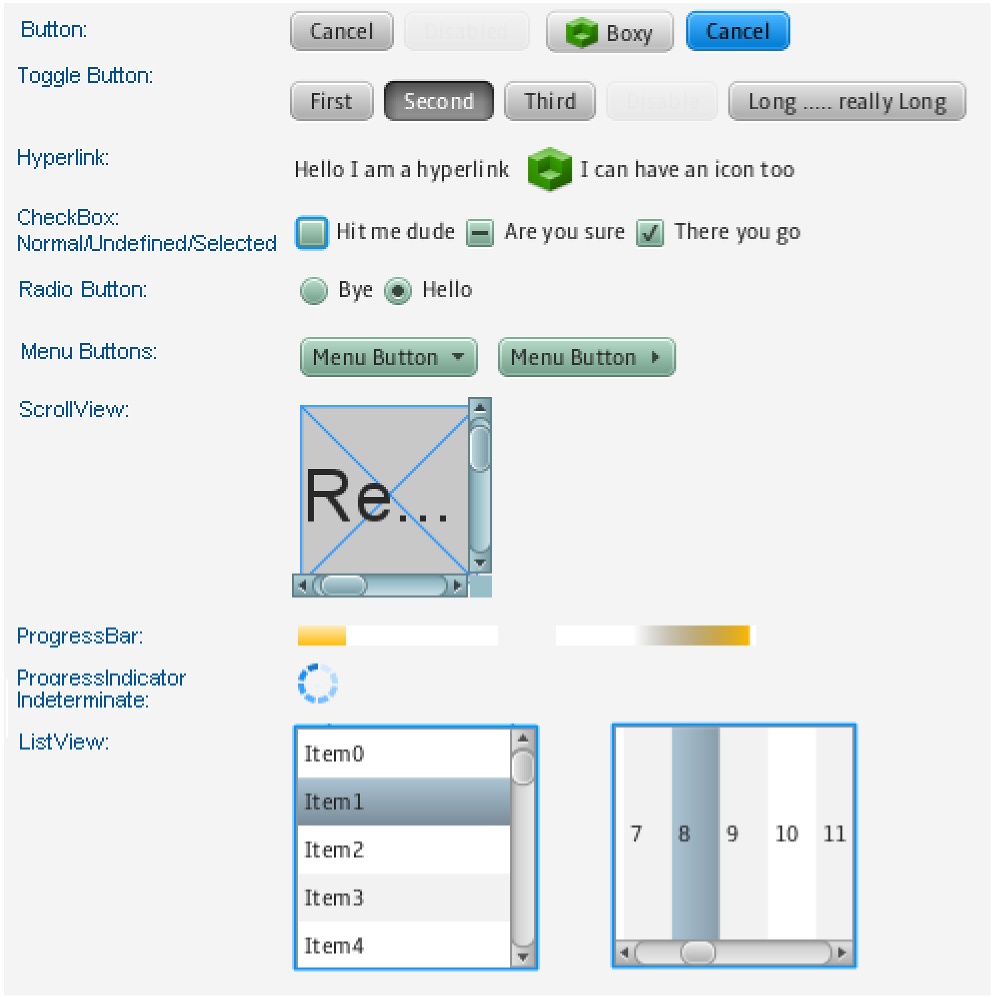


Controls

- Nodes arrange in a layout containers are combination of controls. Controls are GUI components, such as Labels that display text. When the user interacts with a control, such as clicking a Button, the control generates an event.
- Programs can respond to these events - known as event handling - to specify what should happen when each user interaction occurs.
- Event handler is a method that responds to a user interaction.

Controls

- Nodes arrange in a layout containers are combination of controls. Controls are GUI components, such as Labels that display text. When the user interacts with a control, such as clicking a Button, the control generates an event.
- Programs can respond to these events - known as event handling - to specify what should happen when each user interaction occurs.

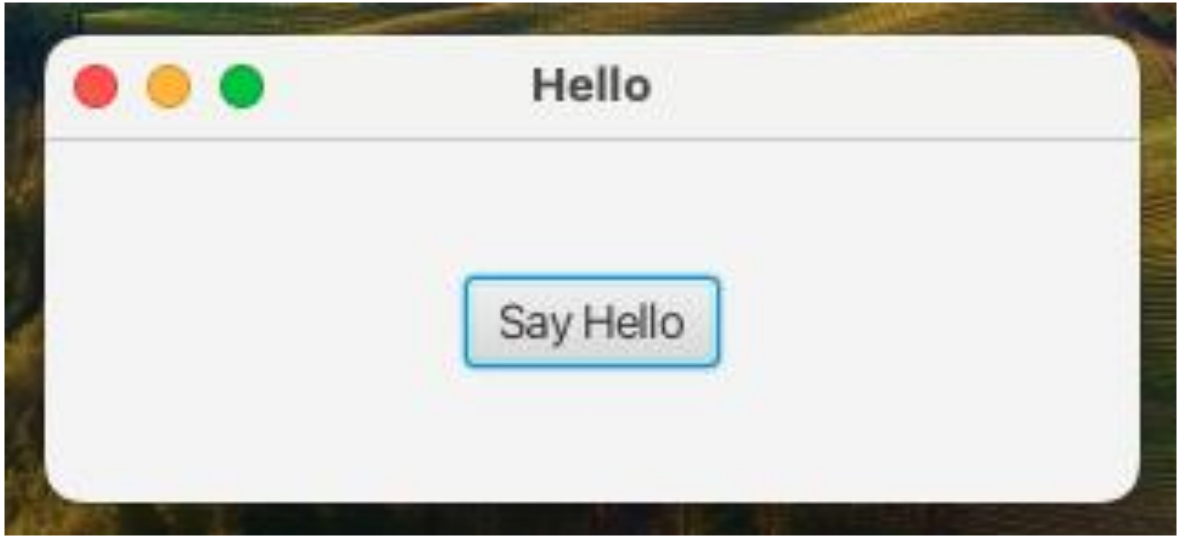


JavaFX By Example

```java
package com.example.javafxdemo;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
import java.io.IOException;
public class HelloApplication extends Application {

    @Override
    public void start(Stage stage) throws IOException {
        // Construct the "Button" and attach an "EventHandler"
        Button btnHello = new Button();
        btnHello.setText("Say Hello");
        // Construct a scene graph of nodes
        StackPane root = new StackPane();  // The root of scene graph is a layout node
        root.getChildren().add(btnHello);  // The root node adds Button as a child

        Scene scene = new Scene(root, 300, 100);  // Construct a scene given the root of scene graph
        stage.setScene(scene);    // The stage sets scene
        stage.setTitle("Hello");  // Set window's title
        stage.show();             // Set visible (show it)
    }

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

- In a JavaFX application, the Application class is typically extended to create the main class.
- Overriding the start method, which serves as the entry point for the JavaFX application.
- The launch method is a static method provided by the Application class in JavaFX,
- It is used to launch the JavaFX application.

Output:



How it works:

A JavaFX GUI Program extends from javafx.applicaation.Application to serve as the entry point for JavaFX applications. This approach follows the framework's design pattern, providing a structured way to manage the lifecycle of JavaFX applications. Life cycle includes:

1. init(): Called before the start method.
2. start(Stage primaryStage); The main entry point for the application, where we set up the user interface.

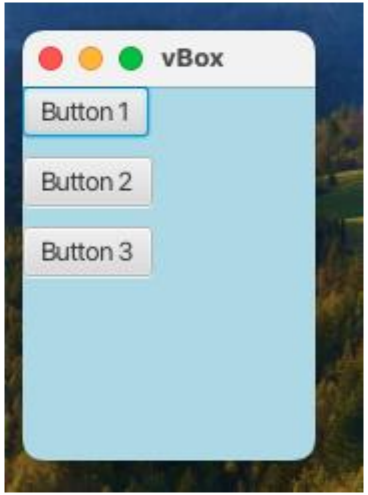
   ```
   public void start(Stage stage){//setup interface}
   ```

1. stop() : called when the application is about to stop.

launch() - When we call launch(args) from the main method, the JavaFX runtime initializes the JavaFX environment and prepares to start the application.

JavaFX Layouts

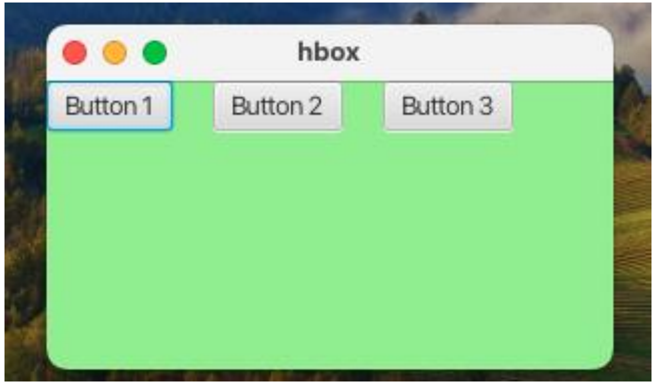
- VBox:
  - The VBox class arranges its content nodes vertically in a single column.



```
VBox vbox = new VBox(10); //
spacing between nodes
Button btn1 = new Button("Button
1");
Button btn2 = new Button("Button
2");
vbox.getChildren().addAll(btn1,
btn3);
```

HBox:

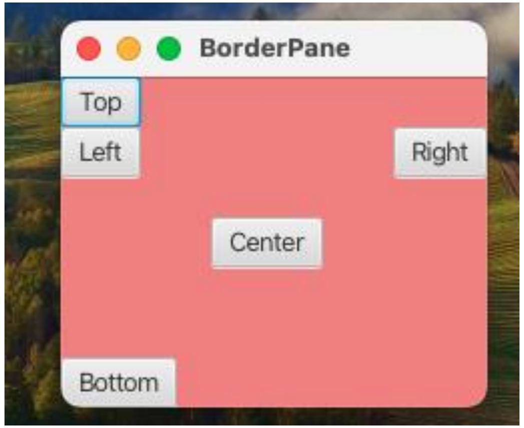
- The HBox class arranges its content nodes horizontally in a single row.



```
HBox hbox = new HBox(20); //
Button btn1 = new
Button("Button 1");
Button btn2 = new
Button("Button 2");
hbox.getChildren().addAll(btn1,
btn3);
```

BorderPane

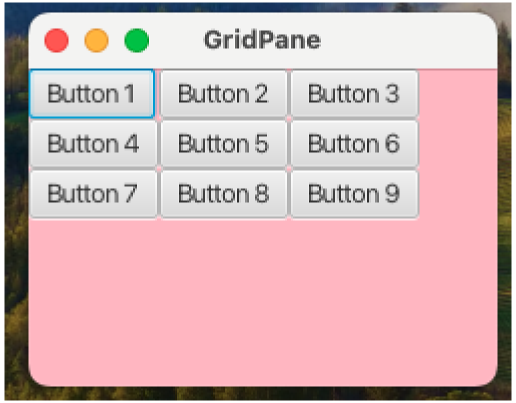
- The BorderPane class lays out its content nodes in the top, bottom, right, left or center region.



```
BorderPane borderPane = new
BorderPane();
Button top = new Button("Top");
Button left = new Button("Left");
Button center = new
Button("Center");
borderPane.setTop(top);
borderPane.setLeft(left);
borderPane.setCenter(center);
```

GridPane

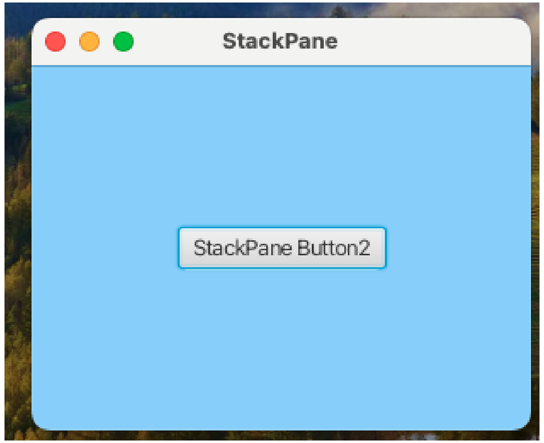
- The GridPane class enables the developer to create a flexible grid of rows and columns in which to layout content nodes.



```
GridPane gridPane = new
GridPane();
gridPane.setStyle("-fx-background-
color: lightpink;");
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
         Button btn = new
Button("Button " + (i * 3 + j +
1));
         gridPane.add(btn, j, i);
    }
}
```

StackPane

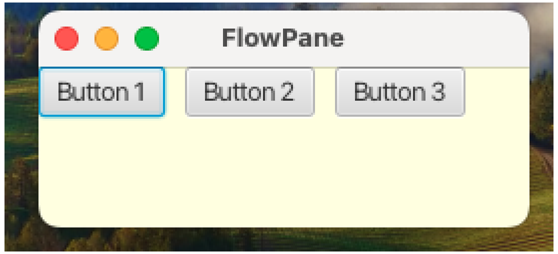
- The StackPane class places its content nodes in a back-to-front single stack.



```
StackPane stackPane = new
StackPane();
stackPane.setStyle("-fx-
background-color:
lightskyblue;");
Button btn = new
Button("StackPane Button");
Button btn2 = new
Button("StackPane Button2");
stackPane.getChildren().addAll(
btn,btn2);
```

FlowPane

- The FlowPane class arranges its content nodes in either a horizontal or vertical “flow”, wrapping at the specified width(for horizontal) or height(for vertical) boundaries.Components flow in a wrap-around manner, adjusting their position based on available space.



```
FlowPane flowPane = new FlowPane();
flowPane.setStyle("-fx-background-
color: lightyellow;");
flowPane.setHgap(10);
flowPane.setVgap(10);
for (int i = 0; i < 3; i++) {
   Button btn = new Button("Button "
+ (i + 1));
flowPane.getChildren().add(btn);
}
```

JavaFX UI Controls

- Button: Represents a clickable button.
- TextField: Allows users to input text.
- CheckBox and RadioButton: For binary and exclusive selection, respectively.
- ComboBox: A drop-down menu with a list of options.
- Slider: Allows users to select a value from a range.
- TableView: Displays tabular data in rows and columns.
- DatePicker: Enables users to pick a date from a calendar.
- WebView: Embeds a web browser in the application.

Disadvantages of JavaFX

- JavaFX may have a steep learning curve as it consists of rich feature set and modern architecture which makes it more challenging for beginners.
- Performance can be slower than Java Swing.
  - JavaFX have longer initialization and startup time because it loads a large number of libraries for its advance features.
- JavaFX has smaller community as compared to Swing, which will result into fewer online resources, tutorials and community support.
- JavaFX has limited ecosystem due to which there are less number of third party libraries. Lesser number of third party libraries will lead to limitations in its functionality and increase development effort for certain tasks.
- JavaFX consumes more memory as compared to Swing because the rich graphics, animations and media capabilities comes at the cost of higher memory usage.

Assignment - 2

1. Compare AWT with Swing. Write a GUI program using components to find sum and difference of two numbers. Use two text fields for giving input and a label for output. The program should display sum if user presses mouse and difference if user release mouse.
2. What is JavaFX? Compare it with swing. Explain FlowPane layout of JavaFX.
3. Why do we need event handling? Explain the use of action event with example. Why do we need adapter class in event handling?
4. Why do we need swing components ? Explain the uses of check boxes and radio buttons in GUI programming.
5. Compare JavaFX with swing. Explain HBox and BBox layouts of JavaFX.
6. Define event delegation model. What is the task of Layout manager? Describe about default layout manager.
7. Write a program to create a Frame that has two TextField components, one Label and a Button. When the user clicks on the button, calculate the sum of the values entered in the first and second TextField and display the result on the third Label.
8. Thare are the pros and cons of JavaFX. Explain any two JavaFX layout manager.

Thank You!