

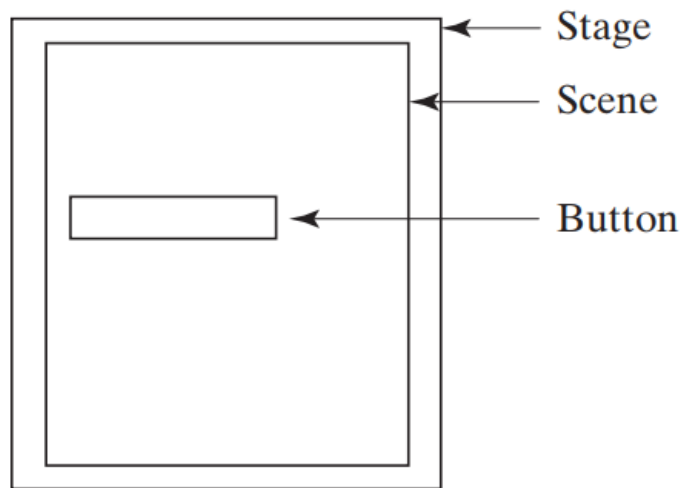
# Introduction to GUI

**IQRA UNIVERSITY**

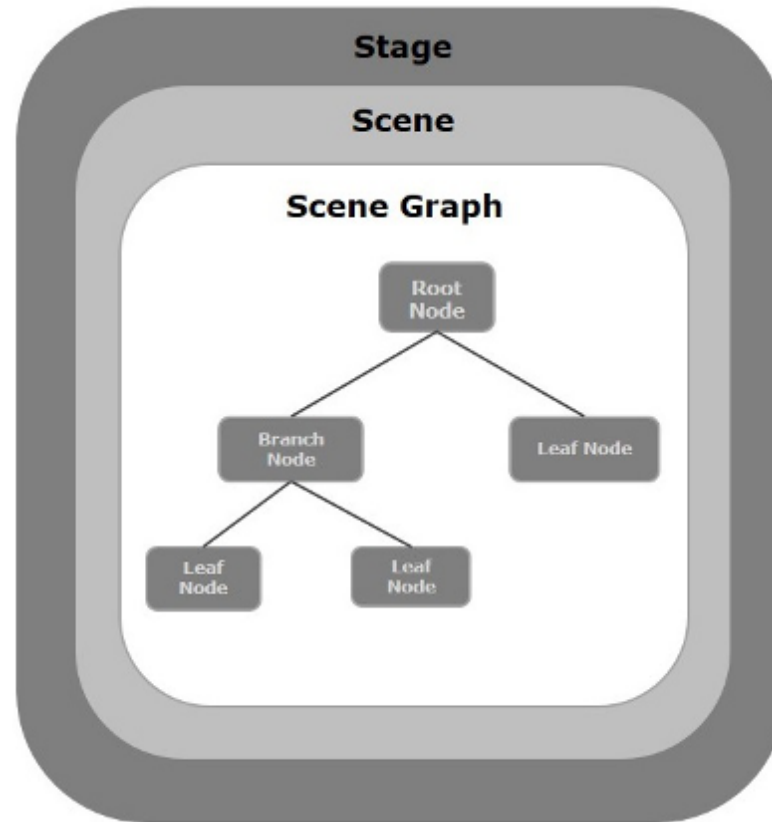
# JavaFX

- JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms.
- The applications developed using JavaFX can run on various devices such as Desktop Computers, Mobile Phones, TVs, Tablets, etc.
- To develop GUI Applications using Java programming language, the programmers rely on libraries such as Advanced Windowing Tool kit and Swings.
- After the advent of JavaFX, Java programmers can now develop GUI applications effectively with rich content.

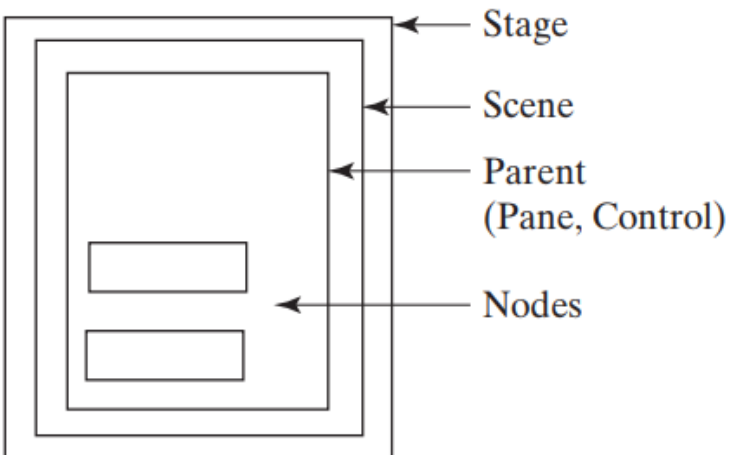
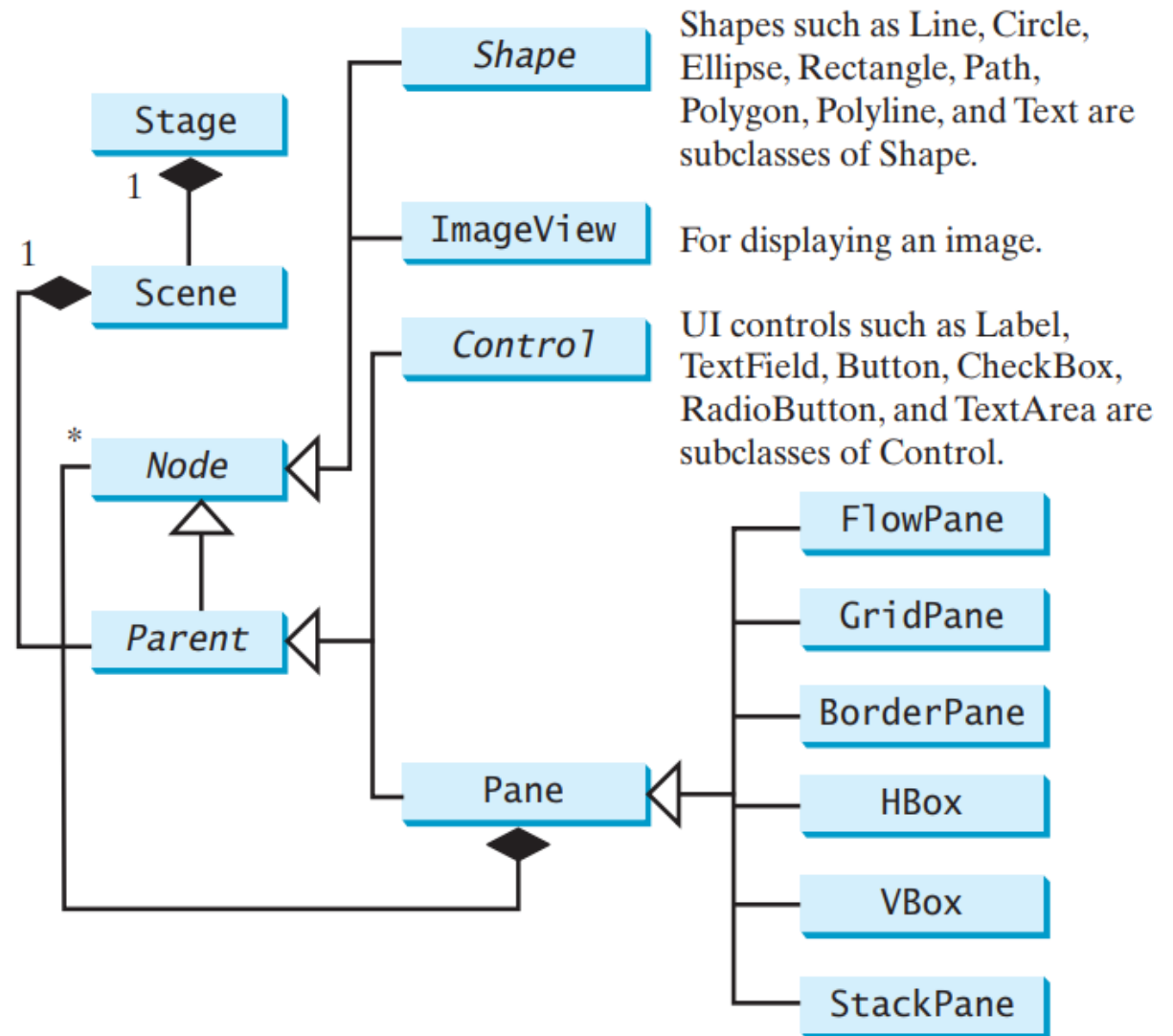
# Relationship I



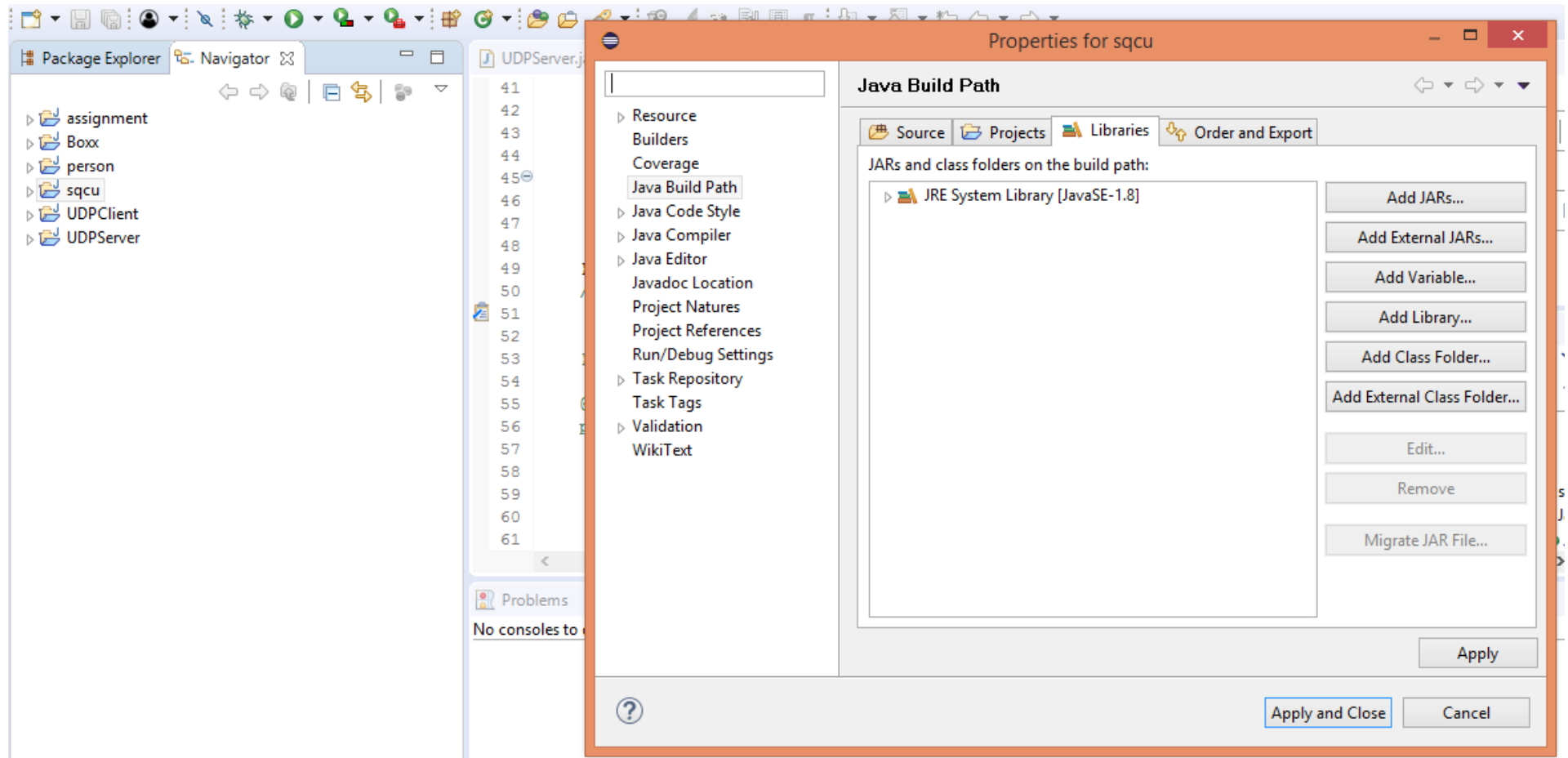
# Application Structure



# Relationship II



1- Check JRE System library should be 1.8  
(by clicking on any java project-> properties->java build  
path



## 2- Add Access Rule

The screenshot shows the Eclipse IDE interface. On the left is the 'Project Explorer' with a tree view containing: Resource, Builders, Coverage, Java Build Path (selected), Java Code Style, Java Compiler, Java Editor, Javadoc Location, Project Natures, Project References, Run/Debug Settings, Task Repository, Task Tags, Validation, and WikiText.

The main area displays the 'Java Build Path' dialog box, with the 'Libraries' tab selected. It shows 'JRE System Library [JavaSE-1.8]' and 'Access rules: 1 rule defined, added to all library ch'. Below this, it lists 'External annotations: (None)', 'Native library location: (None)', and 'resources.jar - C:\eclipse\jre\lib'.

Two 'Add Access Rule' dialog boxes are overlaid on the main window. The one in the foreground has 'Resolution' set to 'Forbidden' and 'Rule Pattern' is empty. The one in the background has 'Resolution' set to 'Accessible' and 'Rule Pattern' set to 'javafx/\*\*'. Both dialog boxes include a help icon (?) and 'OK' and 'Cancel' buttons. The text inside the dialog boxes reads: 'Enter a pattern for the rule.', 'Allowed wildcards are '\*', '?' and '\*'. Pattern segments are separated by '/'. '\*' matches any number of segments. Examples are: 'java/util/\*', '\*\*/internal/\*', 'org/e\*/\*'.

At the bottom right, a snippet of the 'Access rules' list is visible, showing 'Accessible: javafx/\*\*' with a checkmark.

# Basic Structure of a JavaFX Program

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;

public class MyJavaFX extends Application {
    @Override // Override the start method in the Application class
    public void start(Stage primaryStage) {
        // Create a scene and place a button in the scene
        Button btOK = new Button("OK");
        Scene scene = new Scene(btOK, 200, 250);
        primaryStage.setTitle("MyJavaFX"); // Set the stage title
        primaryStage.setScene(scene); // Place the scene in the stage
        primaryStage.show(); // Display the stage
    }

    /**
     * The main method is only needed for the IDE with limited
     * JavaFX support. Not needed for running from the command line.
     */
    public static void main(String[] args) {
        Application.launch(args);
    }
}
```





# Layouts

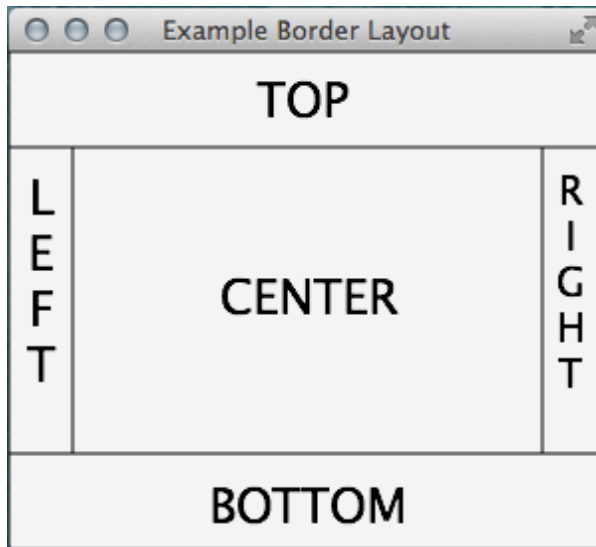
1. BorderPane
2. StackPane
3. GridPane
4. FlowPane
5. HBox
6. VBox
7. TilePane
8. AnchorPane

## JavaFX BorderPane

BorderPane arranges the nodes at the left, right, centre, top and bottom of the screen. It is represented by `javafx.scene.layout.BorderPane` class. This class provides various methods like `setRight()`, `setLeft()`, `setCenter()`, `setBottom()` and `setTop()` which are used to set the position for the specified nodes. We need to instantiate BorderPane class to create the BorderPane layout.

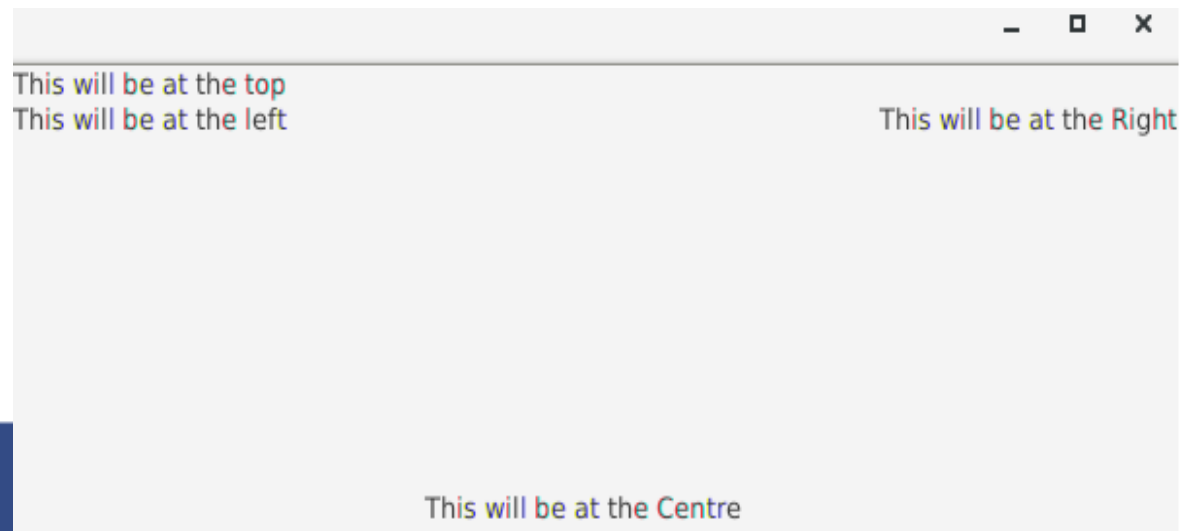
### Constructors

1. `BorderPane()` : create the empty layout
2. `BorderPane(Node Center)` : create the layout with the center node
3. `BorderPane(Node Center, Node top, Node right, Node bottom, Node left)` : create the layout with all the nodes



# Example of BorderPane

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.*;
import javafx.stage.Stage;
public class Label_Test extends Application {
    public void start(Stage primaryStage) throws Exception {
        BorderPane BPane = new BorderPane();
        BPane.setTop(new Label("This will be at the top"));
        BPane.setLeft(new Label("This will be at the left"));
        BPane.setRight(new Label("This will be at the Right"));
        BPane.setCenter(new Label("This will be at the Centre"));
        BPane.setBottom(new Label("This will be at the bottom"));
        Scene scene = new Scene(BPane,600,400);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main(String[] args) {
        Application.launch(args);
    }
}
```

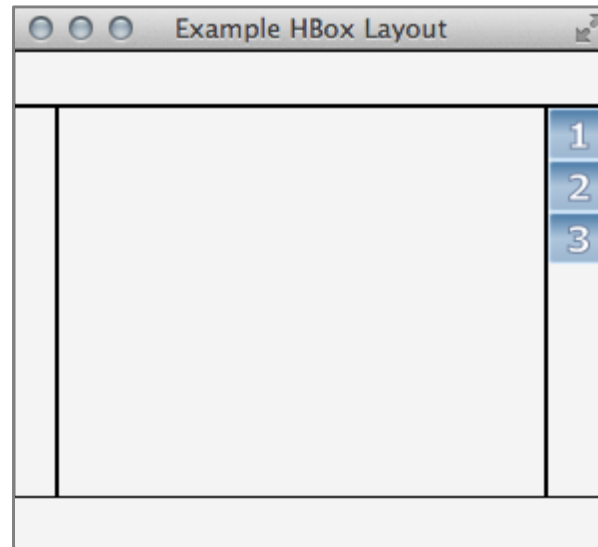


## JavaFX StackPane

The StackPane layout pane places all the nodes into a single stack where every new node gets placed on the top of the previous node. It is represented by **`javafx.scene.layout.StackPane`** class. We just need to instantiate this class to implement StackPane layout into our application.

### Constructors

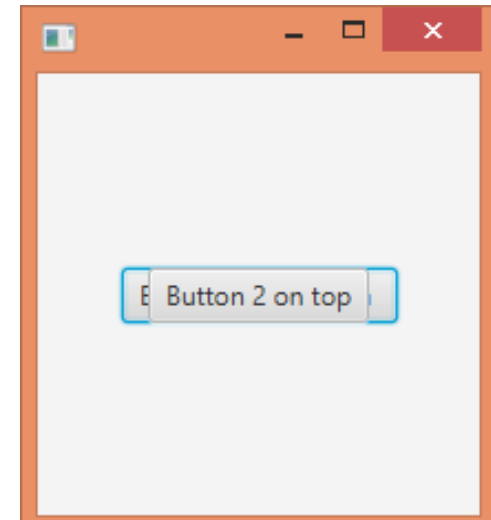
1. `StackPane()`
2. `StackPane(Node/Children)`



StackPane

# Example of StackPane

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class Label_Test extends Application {
    public void start(Stage primaryStage) throws Exception {
        Button btn1 = new Button("Button 1 on bottom ");
        Button btn2 = new Button("Button 2 on top");
        StackPane root = new StackPane();
        Scene scene = new Scene(root,200,200);
        root.getChildren().addAll(btn1,btn2);
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```



```

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.*;
import javafx.scene.control.Label;
import javafx.stage.Stage;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
public class test extends Application{
    private Button btnHello;
    @Override
    public void start(Stage primarystage) throws Exception {
        // Construct the "Button" and attach an "EventHandler"
        btnHello = new Button();
        btnHello.setText("Say Hello");
        // Using JDK 8 Lambda Expression to construct an EventHandler<ActionEvent>
        btnHello.setOnAction(evt -> System.out.println("Hello World!"));
        // 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
        primarystage.setScene(scene); // The stage sets scene
        primarystage.setTitle("Hello"); // Set window's title
        primarystage.show(); // Set visible (show it)
    }
    public static void main(String[] args) {
        Application.launch(args);
    }
}

```