**14.Write a JavaFX program to implement all types of layouts/roots.**

**Theory:**
Layouts are the top level container classes that define the UI styles for scene graph objects. Layout can be seen as the parent node to all the other nodes. JavaFX provides various layout panes that support different styles of layouts.

In JavaFX, Layout defines the way in which the components are to be seen on the stage. It basically organizes the scene-graph nodes. We have several built-in layout panes in JavaFX that are HBox, VBox, StackPane, FlowBox, AnchorPane, etc. Each Built-in layout is represented by a separate class which needs to be instantiated in order to implement that particular layout pane.

*Steps to create layout*

In order to create the layouts, we need to follow the following steps.

1.Instantiate the respective layout class, for example, **HBox root = new HBox();**

2.Setting the properties for the layout, for example, **root.setSpacing(20);**

3.Adding nodes to the layout object, for example, **root.getChildren().addAll(<NodeObjects>);**

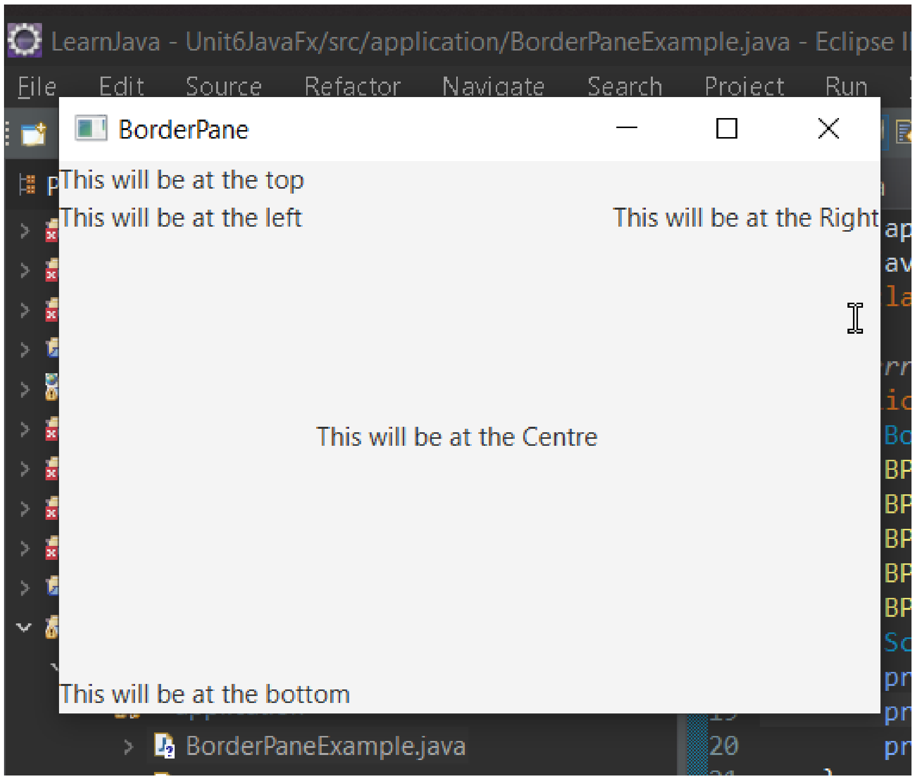
Types of JavaFX Layouts:
1. BorderPane
2. FlowPane
3. Hbox
4. Vbox
5. GridPane

*1. 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.

**Source Code:**
```
package application;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.*;
import javafx.stage.Stage;
public class BorderPaneExample extends Application {

   @Override
   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) {
        launch(args);
    }

}
```

**Output:**



LearnJava - Unit6JavaFx/src/application/BorderPaneExample.java - Eclipse I
File Edit Source Refactor Navigate Search Project Run
BorderPane
BorderPaneExample.java

```
This will be at the top
This will be at the left                              This will be at the Right



                  This will be at the Centre



This will be at the bottom
```

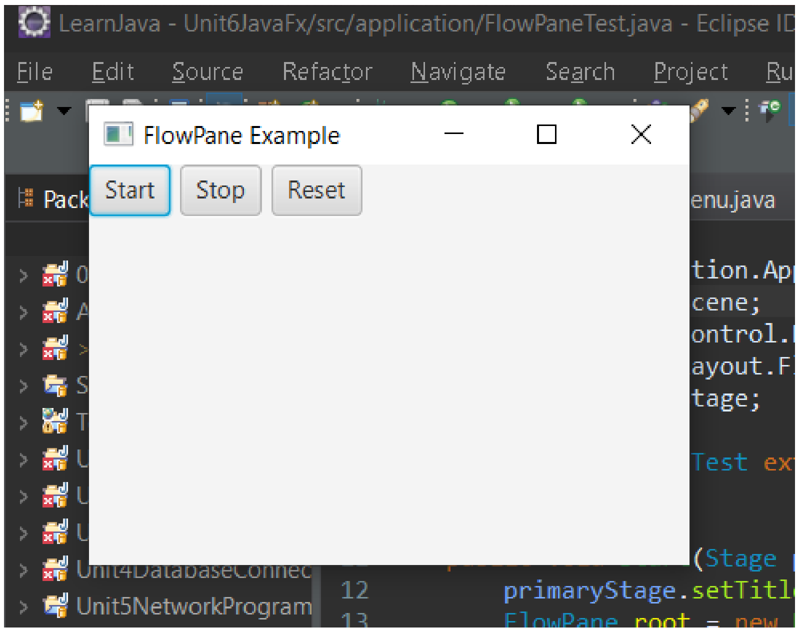
*2. JavaFX FlowPane*

FlowPane layout pane organizes the nodes in a flow that are wrapped at the flowpane's boundary. The horizontal flowpane arranges the nodes in a row and wrap them according to the flowpane's width. The vertical flowpane arranges the nodes in a column and wrap them according to the flowpane's height. FlowPane layout is represented by javafx.scene.layout.FlowPane class. We just need to instantiate this class to create the flowpane layout.

**Source code:**

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

public class FlowPaneTest extends Application {

   @Override
   public void start(Stage primaryStage) {
      primaryStage.setTitle("FlowPane Example");
      FlowPane root = new FlowPane();
      root.setVgap(6);
      root.setHgap(5);
      root.setPrefWrapLength(250);
      root.getChildren().add(new Button("Start"));
      root.getChildren().add(new Button("Stop"));
      root.getChildren().add(new Button("Reset"));
      Scene scene = new Scene(root,300,200);

      primaryStage.setScene(scene);
      primaryStage.show();
   }

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

**Output:**


LearnJava - Unit6JavaFx/src/application/FlowPaneTest.java - Eclipse ID
File Edit Source Refactor Navigate Search Project Ru
FlowPane Example
Start
Stop
Reset
Unit5NetworkProgram

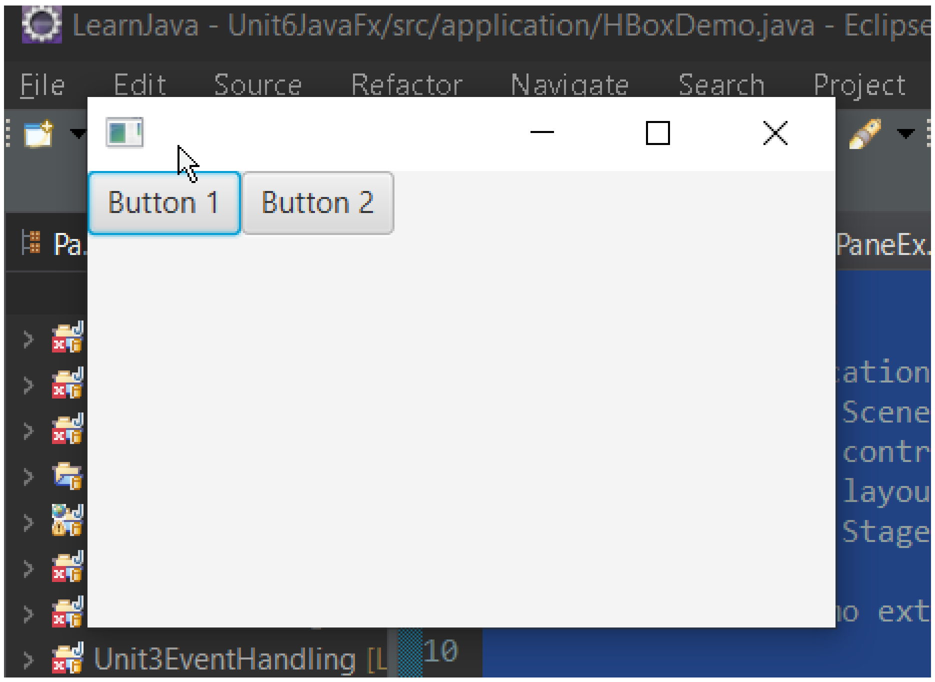
*3. JavaFX  Hbox*

HBox layout pane arranges the nodes in a single row. It is represented by javafx.scene.layout.HBox class. We just need to instantiate HBox class in order to create HBox layout.

**Source Code:**

```
package application;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.HBox;
import javafx.stage.Stage;
public class HBoxDemo extends Application {
        @Override
        public void start(Stage primaryStage) throws Exception {
                Button btn1 = new Button("Button 1");
                Button btn2 = new Button("Button 2");
                HBox root = new HBox();
                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);
        }
}
```

***Output:***


LearnJava - Unit6JavaFx/src/application/HBoxDemo.java - Eclipse
File Edit Source Refactor Navigate Search Project
Button 1
Button 2
Unit3EventHandling

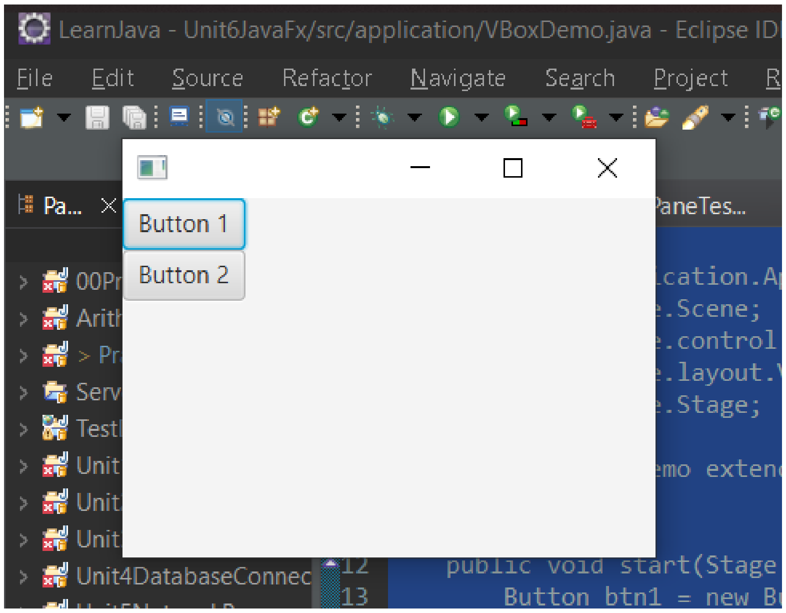
*4. JavaFX  Vbox*

Instead of arranging the nodes in horizontal row, Vbox Layout Pane arranges the nodes in a single vertical column. It is represented by javafx.scene.layout.VBox class which provides all the methods to deal with the styling and the distance among the nodes. This class needs to be instantiated in order to implement VBox layout in our application.

**Source Code:**

```
package application;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
public class VBoxDemo extends Application {
        @Override
        public void start(Stage primaryStage) throws Exception {
                Button btn1 = new Button("Button 1");
                Button btn2 = new Button("Button 2");
                VBox root = new VBox();
                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);
        }
}
```

**Output:**



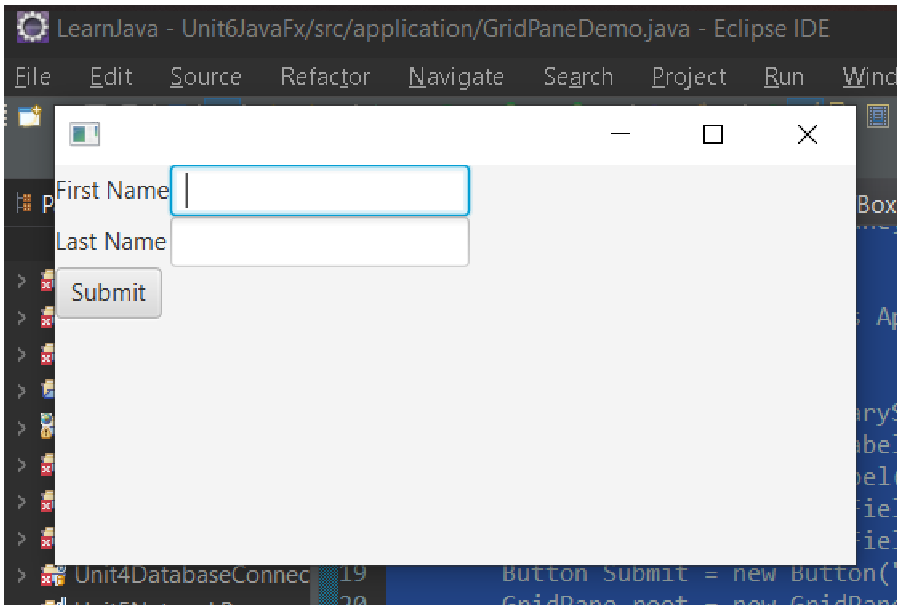
*5. JavaFX  GridPane*

GridPane Layout pane allows us to add the multiple nodes in multiple rows and columns. It is seen as a flexible grid of rows and columns where nodes can be placed in any cell of the grid. It is represented by javafx.scence.layout.GridPane class. We just need to instantiate this class to implement GridPane.

**Source Code:**

```
package application;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
public class GridPaneDemo extends Application {
        @Override
        public void start(Stage primaryStage) throws Exception {
                Label first_name = new Label("First Name");
                Label last_name = new Label("Last Name");
                TextField tf1 = new TextField();
                TextField tf2 = new TextField();
                Button Submit = new Button("Submit");
                GridPane root = new GridPane();
                Scene scene = new Scene(root, 400, 200);
                root.addRow(0, first_name, tf1);
                root.addRow(1, last_name, tf2);
                root.addRow(2, Submit);
                primaryStage.setScene(scene);
                primaryStage.show();
        }
        public static void main(String[] args) {
                launch(args);
        }
}
```

**Output:**


LearnJava - Unit6JavaFx/src/application/GridPaneDemo.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Wind
First Name
Last Name
Submit
Unit4DatabaseConnec
Button Submit = new Button(