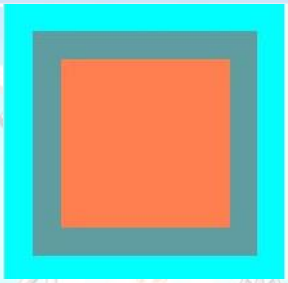
JAVAFX LAYOUT PANE

Stack Pane

Stack Pane lays out its children in a back-to-front stack.

The z-order of the children is defined by the order of the children list (accessible by calling ge tChildren): the 0th child being the bottom and last child on top of the stack.

The stack pane attempts to resize each child to fill its own content area. In the case if a child cannot be resized to fill the area of the Stack Pane (either because it was not resizable or its max size prevented it)then it will be aligned within the area using the alignment Property of the stack pane, which defaults to Pos.CENTER.



Example

```
// Create a StackPane
StackPane pane = new StackPane();

// Create three squares
Rectangle rectBottom = new
Rectangle(250, 250);
rectBottom.setFill(Color.AQUA);
Rectangle rectMiddle = new
Rectangle(200, 200);
rectMiddle.setFill(Color.CADETB
LUE); Rectangle rectUpper = new
Rectangle(150, 150);
rectUpper.setFill(Coor.CORAL);

 // Place them on top of each other
 pane.getChildren().addAll(rectBottom,
 rectMiddle, rectUpper);
```

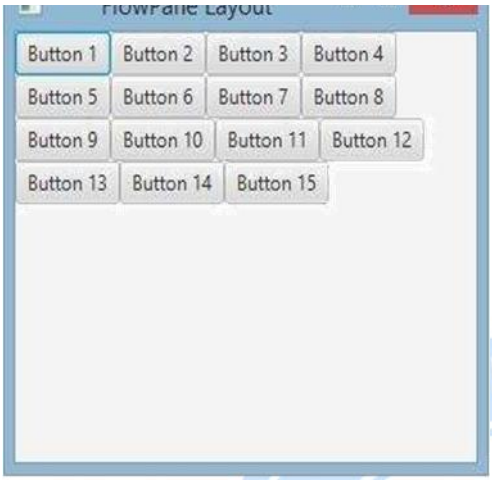
FLOWPANE :

FlowPane lays out nodes in rows or columns based on the available horizontal or vertical spaceavailable. It wraps nodes to the next line when the horizontal space is less than the total of all the nodes' widths; it wraps nodes to the next column when the vertical space is less than the total of all the nodes' heights. This example illustrates the default horizontal layout:

```
import javafx.application.Application;import
javafx.fxml.FXMLLoader;
 import javafx.scene.Parent;
import javafx.scene.Scene;
import
javafx.scene.control.B
utton; import
javafx.scene.layout.Fl
owPane;import
javafx.stage.Stage;

public class Main extends Application {@Override
    public void start(Stage primaryStage) throwsException{ FlowPane root =
        new FlowPane();
        for (int i=1; i<=15; i++) {
            Button b1=new Button("Button"+String.valueOf(i));
            root.getChildren().add(b1); //for
            adding button toroot


        }
        Scene scene = new Scene(root, 300, 250);
        primaryStage.setTitle("FlowPane Layout");
        primaryStage.setScene(scene); primaryStage.show();
    }
    public static void main(String[] args) {launch(args);
    }
}
```



JAVAFX GRIDPANE :

A JavaFX GridPane is a layout component which lays out its child components in a grid. The size of the cells in the grid depends on the components displayed in the GridPane, but there are some rules. All cells in the same row will have the same height, and all cells in the same columnwill have the same width. Different rows can have different heights and different columns can have different widths.

Creating a GridPane

You create a JavaFX GridPane via its constructor. Here is a JavaFX GridPane instantiationexample:

```
GridPane gridPane = new GridPane();
```

Adding Children to a GridPane

You can add children to a JavaFX GridPane in several ways. The easiest way is to use the add() of the GridPane. Here is an example of adding 6 buttons to a GridPane:

```
Button button1 = new Button("Button 1");
Button button2 = new Button("Button 2");
 Button button3 = new Button("Button 3");
 Button button4 = new Button("Button 4");
 Button button5 = new Button("Button 5");
 Button button6 = new Button("Button 6");
GridPane gridPane = new GridPane();
 gridPane.add(button1, 0, 0, 1, 1);
gridPane.add(button2, 1, 0, 1, 1);
gridPane.add(button3, 2, 0, 1, 1);
gridPane.add(button4, 0, 1, 1, 1);
gridPane.add(button5, 1, 1, 1, 1);
```

```
gridPane.add(button6, 2, 1, 1, 1);
```

The first parameter of the add() method is the component (node) to add to the GridPane.

The second and third parameter of the add() method is the column index and row index of thecell in which the component should be displayed. Column and row indexes start from 0.

The fourth and fifth parameter of the add() method are the oclumn span and row span of the component, meaning how many rows and columns the component should extend to. Columnspan and row span works similarly to colspan and rowspan in an **HTML table**.

Adding a GridPane to the Scene Graph

To make a JavaFX GridPane visible you must add it to the JavaFX scene graph. To do so youmust add the GridPane instance to a Scene object, or add the GridPane to a layout componentwhich is added to a Scene object.

Here is an example of adding a JavaFX GridPane to the scene graph:

```
Package com.jenkov.javafx.layouts;

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


public class GridPaneExperiments


   extends Application  {@Override
   public void start(Stage
      primaryStage) throws Exception
      {
      primaryStage.setTitle("GridPane
      Experiment");

      Button button1 = new Button("Button 1");
      Button button2 = new Button("Button 2");
      Button button3 = new Button("Button 3");
      Button button4 = new Button("Button 4");
      Button button5 = new Button("Button 5");
      Button button6 = new Button("Button 6");
```

```
        GridPane gridPane = new GridPane();gridPane.add(button1, 0, 0, 1, 1);
        gridPane.add(button2, 1, 0, 1, 1);
        gridPane.add(button3, 2, 0, 1, 1);
        gridPane.add(button4, 0, 1, 1, 1);
        gridPane.add(button5, 1, 1, 1, 1);
        gridPane.add(button6, 2, 1, 1, 1);

        Scene scene = new Scene(gridPane, 240, 100);
        primaryStage.setScene(scene); primaryStage.show();
    }

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

The application resulting from this application looks like the following screen shots.