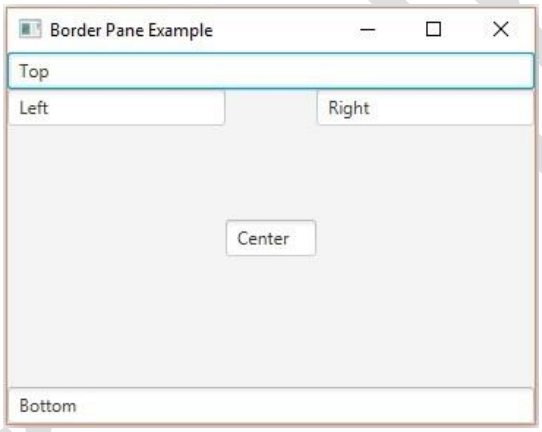
BorderPane

If we use the BorderPane, the nodes are arranged in the Top, Left, Right, Bottom and Center positions.
The class named **BorderPane** of the package **javafx.scene.layout**represents the BorderPane.
This class contains five properties, which include −
bottom − This property is of **Node** type and it represents the node placed at the bottom of the BorderPane. You can set value to this property using the setter method **setBottom()**.
center − This property is of **Node** type and it represents the node placed at the center of the BorderPane. You can set value to this property using the setter method **setCenter()**.
left − This property is of **Node** type and it represents the node placed at the left of the BorderPane. You can set value to this property using the setter method **setLeft()**.
right − This property is of **Node** type and it represents the node placed at the right of the BorderPane. You can set value to this property using the setter method **setRight()**.
top − This property is of **Node** type and it represents the node placed at the top of the BorderPane. You can set value to this property using the setter method **setTop()**.
In addition to these, this class also provides the following method −
setAlignment() − This method is used to set the alignment of the nodes belonging to this pane. This method accepts a node and a priority value.
Example: The following program is an example of the **BorderPane** layout. In this, we are inserting a five text fields in the Top, Bottom, Right, Left and Center positions.

```
package javafxapplication8;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.scene.control.TextField;

public class JavaFXApplication8 extends Application {

    @Override
    public void start(Stage stage) {
      //Instantiating the BorderPane class
      BorderPane bPane = new BorderPane();

      //Setting the top, bottom, center, right and left nodes to the pane
      bPane.setTop(new TextField("Top"));
      bPane.setBottom(new TextField("Bottom"));
      bPane.setLeft(new TextField("Left"));
      bPane.setRight(new TextField("Right"));
      bPane.setCenter(new TextField("Center"));

      //Creating a scene object
      Scene scene = new Scene(bPane);
```

```
        //Setting title to the Stage
        stage.setTitle("BorderPane Example");

        //Adding scene to the stage
        stage.setScene(scene);

        //Displaying the contents of the stage
        stage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }

}
```

On executing, the above program generates a JavaFX window as shown below.



StackPane

If we use the Stack Pane, the nodes are arranged on top of another, just like in stack. The node added first is placed at the bottom of the stack and the next node is placed on top of it.

The class named **StackPane** of thepackage **javafx.scene.layout** represents the StackPane. This class contains a single property named alignment. This property represents the alignment of the nodes within the stack pane. In addition to these, this class also provides a method named **setMargin()**. This method is used to set margin for the node within the stack pane.

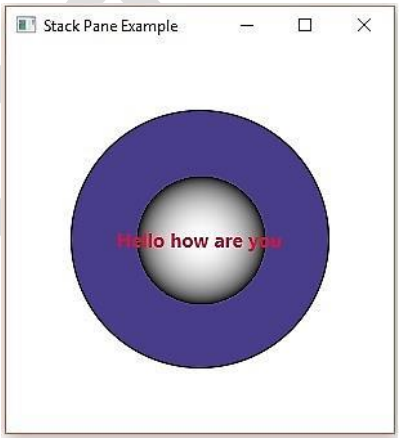
Example: The following program is an example of the **StackPane** layout. In this, we are inserting two Circle, and a Text in the same order.

```
package javafxapplication12;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
```

```
import javafx.scene.shape.Circle;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.Text;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;

public class JavaFXApplication12 extends Application {

    @Override
    public void start(Stage stage) {
      //Drawing a Circle
      Circle circle = new Circle(300, 135, 100);
      circle.setFill(Color.DARKSLATEBLUE);
      circle.setStroke(Color.BLACK);

      //Drawing a Circle
      Circle ncircle = new Circle(300, 135, 50);
      ncircle.setFill(Color.DARKGRAY);
      ncircle.setStroke(Color.BLACK);


      //Creating a text
      Text text = new Text("Hello how are you");

      //Setting the font of the text
      text.setFont(Font.font(null, FontWeight.BOLD, 15));

      //Setting the color of the text
      text.setFill(Color.CRIMSON);

      //setting the position of the text
      text.setX(20);
      text.setY(50);

      //Creating a Stackpane
      StackPane stackPane = new StackPane();

      //Setting the margin for the circle
      stackPane.setMargin(ncircle, new Insets(50, 50, 50, 50));

      //Retrieving the observable list of the Stack Pane
      ObservableList list = stackPane.getChildren();

      //Adding all the nodes to the pane
```

```
        list.addAll(circle, ncircle, text);

        //Creating a scene object
        Scene scene = new Scene(stackPane);

        //Setting title to the Stage
        stage.setTitle("Stack Pane Example");

        //Adding scene to the stage
        stage.setScene(scene);

        //Displaying the contents of the stage
        stage.show();
    }
    public static void main(String args[]){
        launch(args);
    }
}
```

On executing, the above program generates a JavaFX window as shown in figure.



GridPane

If we use Grid Pane in our application, all the nodes that are added to it are arranged in a way that they form a grid of rows and columns. This layout comes handy while creating forms using JavaFX.

The class named **GridPane** of the package **javafx.scene.layout** represents the GridPane. This class provides eleven properties, which are −

alignment − This property represents the alignment of the pane and you can set value of this property using the **setAlignment()** method.

hgap − This property is of the type double and it represents the horizontal gap between columns.

vgap − This property is of the type double and it represents the vertical gap between rows.

gridLinesVisible − This property is of Boolean type. On true, the lines of the pane are set to be visible.

Following are the cell positions in the grid pane of JavaFX −

(0, 0)	(1, 0)	(2, 0)
(2, 1)	(1, 1)	(0, 1)
(2, 2)	(1, 2)	(0, 2)

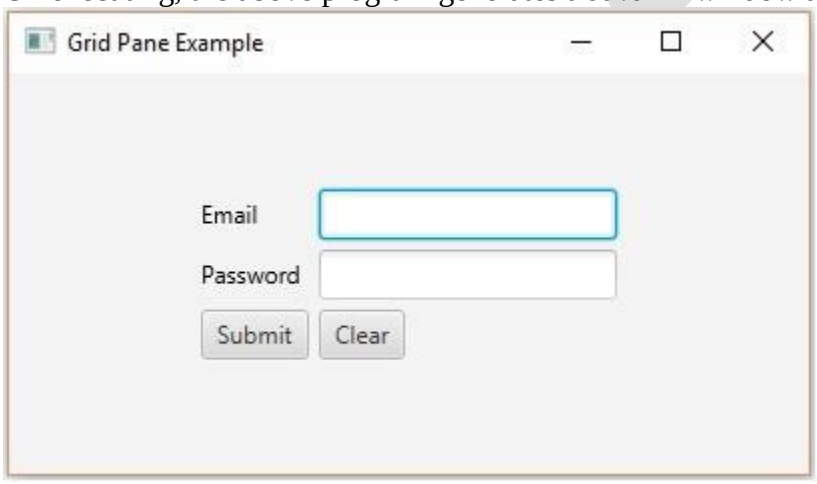
Example : The following program is an example of the grid pane layout. In this, we are creating a form using a Grid Pane.

```
package javafxapplication13;

import javafx.application.Application;
```

```
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.GridPane;
import javafx.scene.text.Text;
import javafx.scene.control.TextField;
import javafx.geometry.Insets;
import javafx.geometry.Pos;


public class JavaFXApplication13 extends Application {

    @Override
    public void start(Stage stage) {
      //creating label email
      Text text1 = new Text("Email");

      //creating label password
      Text text2 = new Text("Password");

      //Creating Text Filed for email
      TextField textField1 = new TextField();

      //Creating Text Filed for password
      TextField textField2 = new TextField();

      //Creating Buttons
      Button button1 = new Button("Submit");
      Button button2 = new Button("Clear");

      //Creating a Grid Pane
      GridPane gridPane = new GridPane();

      //Setting size for the pane
      gridPane.setMinSize(400, 200);

      //Setting the padding
      gridPane.setPadding(new Insets(10, 10, 10, 10));

      //Setting the vertical and horizontal gaps between the columns
      gridPane.setVgap(5);
      gridPane.setHgap(5);

      //Setting the Grid alignment
      gridPane.setAlignment(Pos.CENTER);
```

```
        //Arranging all the nodes in the grid
        gridPane.add(text1, 0, 0);
        gridPane.add(textField1, 1, 0);
        gridPane.add(text2, 0, 1);
        gridPane.add(textField2, 1, 1);
        gridPane.add(button1, 0, 2);
        gridPane.add(button2, 1, 2);

        //Creating a scene object
        Scene scene = new Scene(gridPane);

        //Setting title to the Stage
        stage.setTitle("Grid Pane Example");

        //Adding scene to the stage
        stage.setScene(scene);

        //Displaying the contents of the stage
        stage.show();
    }
    public static void main(String args[]){
        launch(args);
    }
}
```


On executing, the above program generates a JavaFX window as shown below.



FlowPane

If we use flow pane in our application, all the nodes are wrapped in a flow. A horizontal flow pane wraps the elements of the pane at its height, while a vertical flow pane wraps the elements at its width.

The class named **FlowPane** of the package **javafx.scene.layout** represents the Flow Pane. This class contains 7 properties, which includes –

alignment – This property represents the alignment of the contents of the Flow pane. You can set this property using the setter method **setAllignment()**.

columnHalignment – This property represents the horizontal alignments of nodes in a vertical flow pane.

rowValignment – This property represents the vertical alignment of nodes in a horizontal flow pane.

Hgap – This property is of double type and it represents the horizontal gap between the rows/columns of a flow pane.

Orientation – This property represents the orientation of a flow pane.

Vgap – This property is of double type and it represents the vertical gap between the rows/columns of a flow pane.

Example: The following program is an example of the **FlowPane** layout. In this, we are inserting four button in the horizontal flow pane.

```
package javafxapplication17;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.FlowPane;
import javafx.collections.ObservableList;
import javafx.geometry.Insets;

public class JavaFXApplication17 extends Application {

    @Override
    public void start(Stage stage) {
      //Creating button1
      Button button1 = new Button("Button1");

      //Creating button2
      Button button2 = new Button("Button2");

      //Creating button3
      Button button3 = new Button("Button3");

      //Creating button4
      Button button4 = new Button("Button4");

      //Creating a Flow Pane
      FlowPane flowPane = new FlowPane();

      //Setting the horizontal gap between the nodes
      flowPane.setHgap(25);
```

```
        //Setting the margin of the pane
        flowPane.setMargin(button1, new Insets(20, 0, 20, 20));

        //Retrieving the observable list of the flow Pane
        ObservableList list = flowPane.getChildren();

        //Adding all the nodes to the flow pane
        list.addAll(button1, button2, button3, button4);

        //Creating a scene object
        Scene scene = new Scene(flowPane);

        //Setting title to the Stage
        stage.setTitle("Flow Pane Example");

        //Adding scene to the stage
        stage.setScene(scene);

        //Displaying the contents of the stage
        stage.show();
    }
    public static void main(String args[]){
        launch(args);
    }
}
```

On executing, the above program generates a JavaFX window as shown in figure.