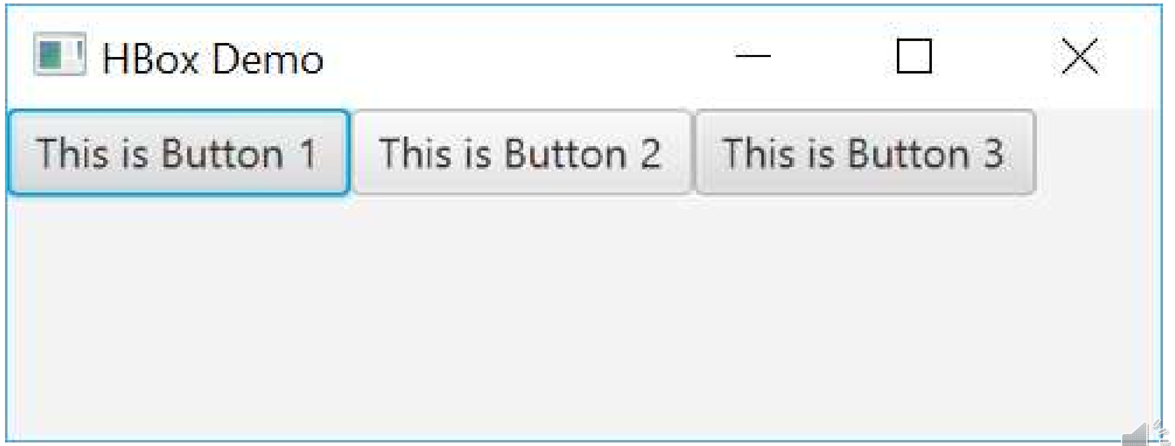
7.6 Graphics Supplement: Outline

- Layout Panes
  - Vbox
  - Hbox
  - StackPane
  - FlowPane
  - GridPane
  - BorderPane
- Combining Layouts
- The Classes **`TextArea`** and **`TextField`**
- Drawing Polygons

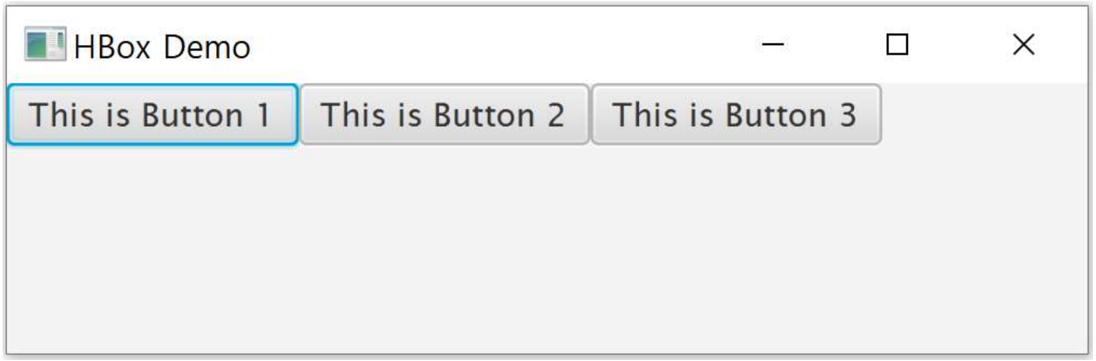
HBox Layout

- Layout panes control how components are laid out and displayed in a JavaFX application
  - VBox, which arranges components vertically
  - The HBox layout arranges components horizontally
- listing 7.15, class HBoxDemo

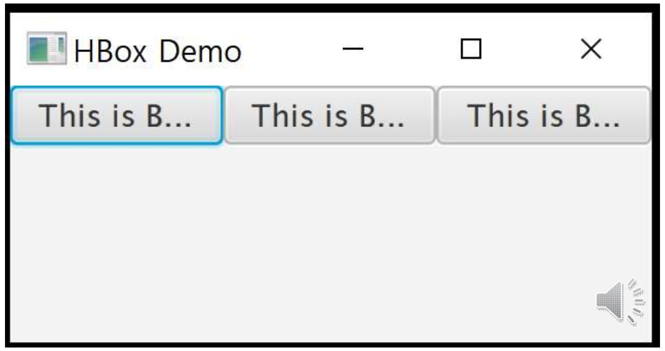


listing 7.15, class HBoxDemo

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

/**
Simple demonstration of adding buttons using the HBox layout.
These buttons do not do anything. That comes in a later version.
*/
public class HBoxDemo extends Application
{
   public static void main(String[] args)
   {
      launch(args);
   }
@Override
 public void start(Stage primaryStage) throws Exception
 {
           HBox root = new HBox();
           root.getChildren().add(new Button("This is Button 1"));
           root.getChildren().add(new Button("This is Button 2"));
           root.getChildren().add(new Button("This is Button 3"));

           Scene scene = new Scene(root, 400, 100);
           primaryStage.setTitle("HBox Demo");
           primaryStage.setScene(scene);
           primaryStage.show();
 }
}
```

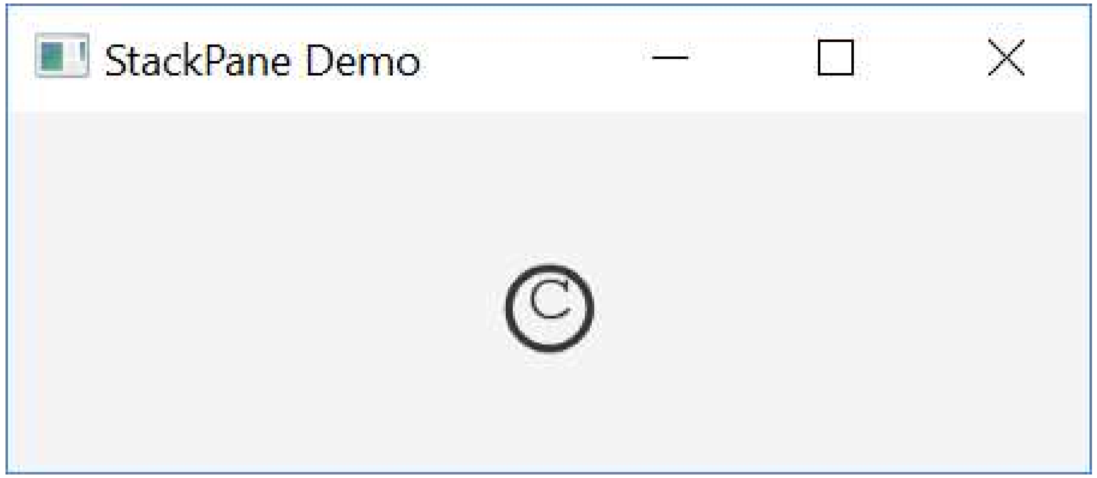


```
           Scene scene = new Scene(root, 250, 100);
```



StackPane Layout

- The StackPane layout
  - stacks components on top of one another
  - provides a way to overlay text onto a shape or image to create a more complex object
- listing 7.16, class StackPaneDemo



Overlay c
onto an O

listing 7.16, class StackPaneDemo

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.layout.StackPane;
import javafx.scene.control.Label;
import javafx.scene.text.Font;
/**
Simple demonstration of drawing two letters on top of each other
using the StackPane layout.
*/
public class StackPaneDemo extends Application
{
   public static void main(String[] args)
   {
      launch(args);
   }
```

```
 @Override
  public void start(Stage primaryStage) throws Exception
  {
          StackPane root = new StackPane();
          Label label1 = new Label("o");
          label1.setFont(Font.font("Courier New", 54));
          Label label2 = new Label("c");
          label2.setFont(Font.font("Courier New", 24));
          root.getChildren().add(label1);
          root.getChildren().add(label2);

          Scene scene = new Scene(root, 300, 100);
          primaryStage.setTitle("StackPane Demo");
          primaryStage.setScene(scene);
          primaryStage.show();
  }
}
```

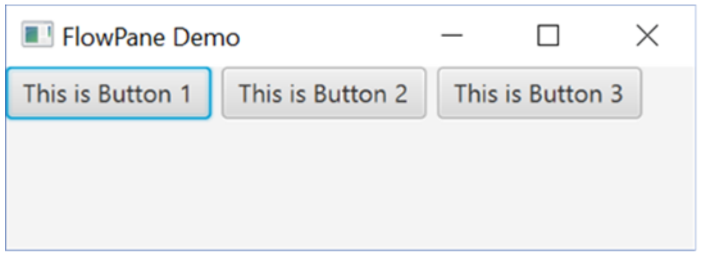
FlowPane Layout

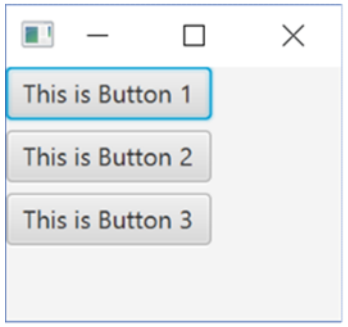
- The FlowPane layout
  - adds components left to right and wraps around to the left when the right side of the screen is reached
  - can control the amount of vertical and horizontal space between elements by using the methods **`setVgap`** and **`setHgap`**
- listing 7.17, class FlowPaneDemo

▪ listing 7.17, class FlowPaneDemo

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.layout.FlowPane;
import javafx.scene.control.Button;
/**
Simple demonstration of adding buttons to the FlowPane layout.
*/
public class FlowPaneDemo extends Application
{
   public static void main(String[] args)
   {
      launch(args);
   }
```

```java
@Override
  public void start(Stage primaryStage) throws Exception
  {
          FlowPane root = new FlowPane();

          // Set a gap of 5 pixels vertically and horizontally
          // between buttons
          root.setVgap(5);
          root.setHgap(5);

          root.getChildren().add(new Button("This is Button 1"));
          root.getChildren().add(new Button("This is Button 2"));
          root.getChildren().add(new Button("This is Button 3"));


          Scene scene = new Scene(root, 500, 200);
          primaryStage.setTitle("FlowPane Demo");
          primaryStage.setScene(scene);
          primaryStage.show();
  }
}
```





When the window is resized, the buttons flow to fit the width of the window

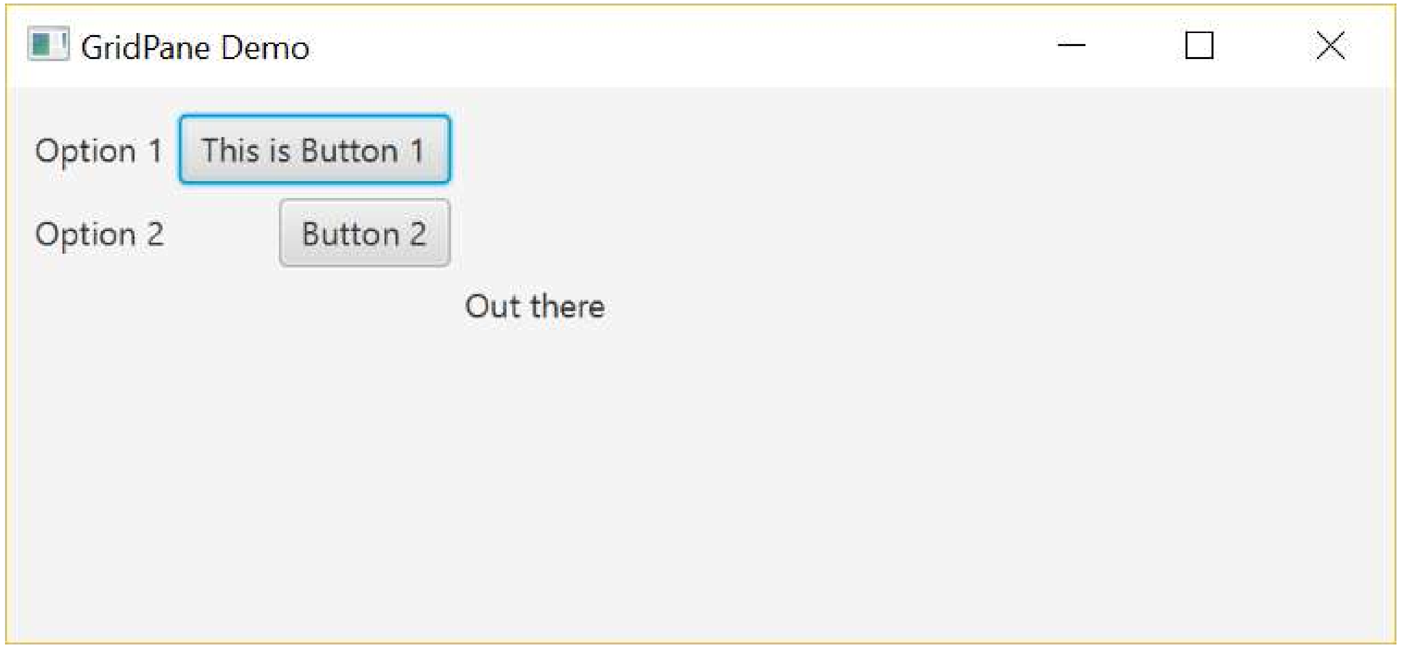
GridPane Layout

- The GridPane layout
  - arranges components into a grid of rows and columns that are accessible like a 2D array
  - The upper left cell is at 0,0;
    - “Out there” is at 2,2
    - This is Button 1 is at 1,0
- listing 7.18, class GridPaneDemo

listing 7.18, class GridPaneDemo

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.layout.GridPane;
import javafx.geometry.HPos;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.geometry.Insets;
/**
Simple demonstration of adding buttons and labels
to the GridPane layout.
*/
public class GridPaneDemo extends Application
{
  public static void main(String[] args)
  {
    launch(args);
  }

 @Override
  public void start(Stage primaryStage) throws Exception
  {
         GridPane root = new GridPane();
```

```
        // Set a gap of 5 pixels vertically and horizontally
        // between elements
        root.setVgap(5);
        root.setHgap(5);
        // Margins around the top, right, bottom, and left
        root.setPadding(new Insets(10,10,10,10));

        // Add three nodes, by default horizontally left-aligned
        root.add(new Label("Option 1"),0,0);
        root.add(new Button("This is Button 1"),1,0);
        root.add(new Label("Option 2"),0,1);

        // Add a button that is horizontally right-aligned
        Button btn2 = new Button("Button 2");
        GridPane.setHalignment(btn2, HPos.RIGHT);
        root.add(btn2,1,1);

        // Add a label to the bottom right of the buttons
        root.add(new Label("Out there"),2,2);

        Scene scene = new Scene(root, 500, 200);
        primaryStage.setTitle("GridPane Demo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

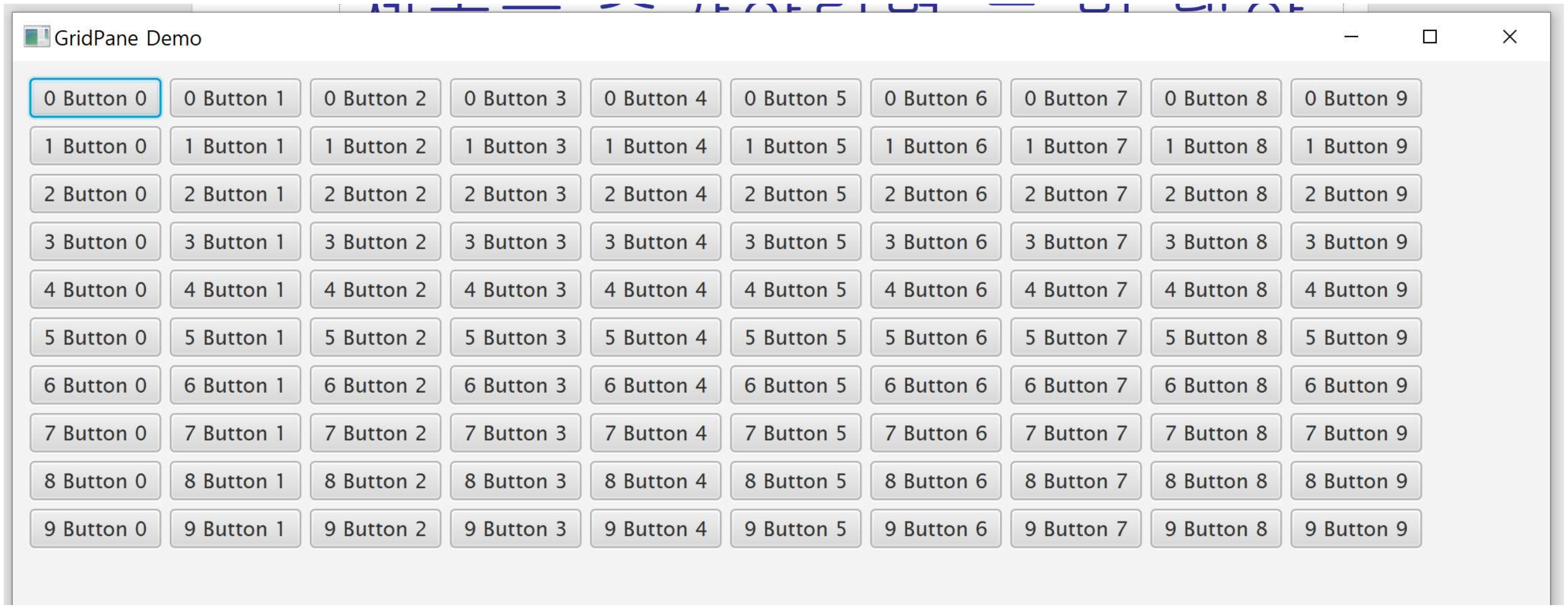

GridPane Demo
Option 1
This is Button 1
Option 2
Button 2
Out there

- public class Insets
  - extends Object
  - A set of inside offsets for the 4 side of a rectangular area
  - Insets(double top, double right, double bottom, double left)
    - Constructs a new Insets instance with four different offsets.

■class GridPaneDemo2

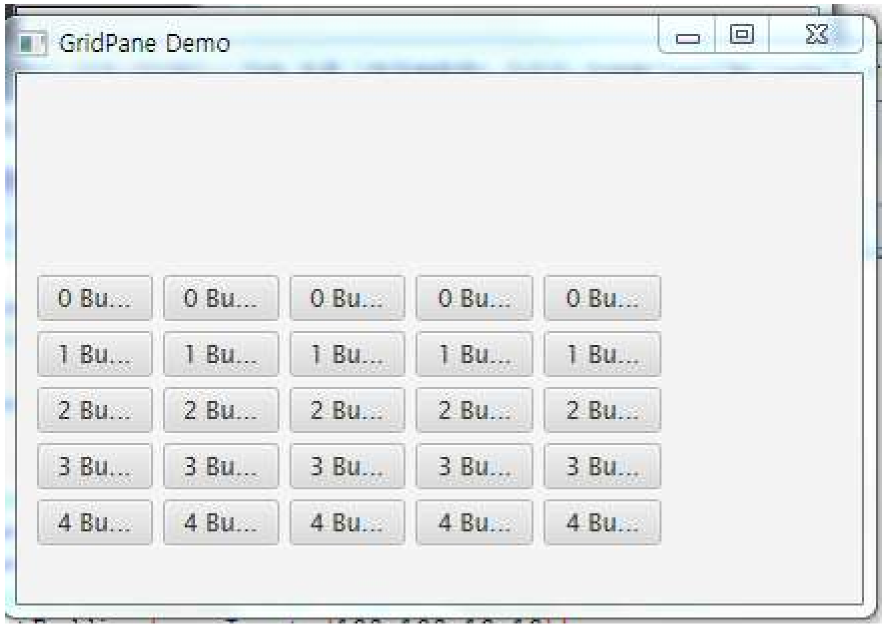
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.layout.GridPane;
import javafx.geometry.HPos;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.geometry.Insets;
/**
Simple demonstration of adding buttons and labels
to the GridPane layout.
*/
public class GridPaneDemo2 extends Application
{
  public static void main(String[] args)
  {
    launch(args);
  }
```

```
 @Override
  public void start(Stage primaryStage) throws Exception
  {
          GridPane root = new GridPane();

          // Set a gap of 5 pixels vertically and horizontally
          // between elements
          root.setVgap(5);
          root.setHgap(5);
          // Margins around the top, right, bottom, and left
          root.setPadding(new Insets(10,10,10,10));

          // Add three nodes, by default horizontally left-aligned
          for (int i = 0; i<10; i++) {
                  for (int j = 0; j<10; j++) {
                  root.add(new Button(i+" Button "+j),j,i);
                  }
          }

          Scene scene = new Scene(root, 900, 500);
          primaryStage.setTitle("GridPane Demo");
          primaryStage.setScene(scene);
          primaryStage.show();
  }
}
```


GridPane Demo

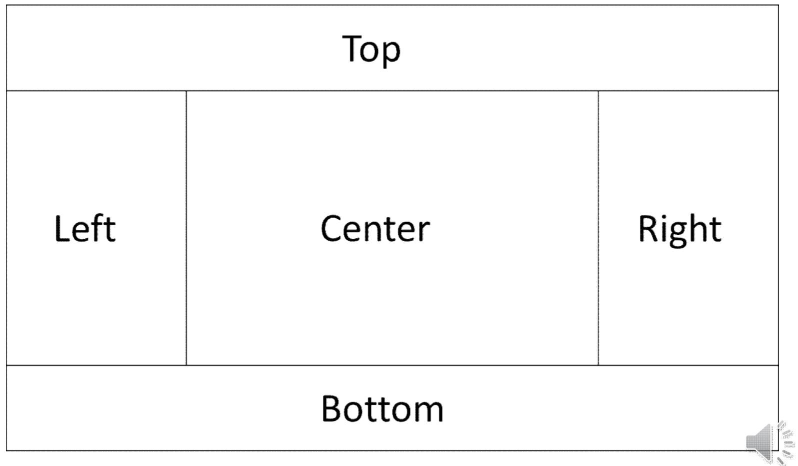
0 Button 0	0 Button 1	0 Button 2	0 Button 3	0 Button 4	0 Button 5	0 Button 6	0 Button 7	0 Button 8	0 Button 9
1 Button 0	1 Button 1	1 Button 2	1 Button 3	1 Button 4	1 Button 5	1 Button 6	1 Button 7	1 Button 8	1 Button 9
2 Button 0	2 Button 1	2 Button 2	2 Button 3	2 Button 4	2 Button 5	2 Button 6	2 Button 7	2 Button 8	2 Button 9
3 Button 0	3 Button 1	3 Button 2	3 Button 3	3 Button 4	3 Button 5	3 Button 6	3 Button 7	3 Button 8	3 Button 9
4 Button 0	4 Button 1	4 Button 2	4 Button 3	4 Button 4	4 Button 5	4 Button 6	4 Button 7	4 Button 8	4 Button 9
5 Button 0	5 Button 1	5 Button 2	5 Button 3	5 Button 4	5 Button 5	5 Button 6	5 Button 7	5 Button 8	5 Button 9
6 Button 0	6 Button 1	6 Button 2	6 Button 3	6 Button 4	6 Button 5	6 Button 6	6 Button 7	6 Button 8	6 Button 9
7 Button 0	7 Button 1	7 Button 2	7 Button 3	7 Button 4	7 Button 5	7 Button 6	7 Button 7	7 Button 8	7 Button 9
8 Button 0	8 Button 1	8 Button 2	8 Button 3	8 Button 4	8 Button 5	8 Button 6	8 Button 7	8 Button 8	8 Button 9
9 Button 0	9 Button 1	9 Button 2	9 Button 3	9 Button 4	9 Button 5	9 Button 6	9 Button 7	9 Button 8	9 Button 9

```
 @Override
  public void start(Stage primaryStage) throws Exception
  {
           GridPane root = new GridPane();

           // Set a gap of 5 pixels vertically and horizontally
           // between elements
           root.setVgap(5);
           root.setHgap(5);
           // Margins around the top, right, bottom, and left
           root.setPadding(new Insets(100,100,10,10));

           // Add three nodes, by default horizontally left-aligned
           for (int i = 0; i<5; i++) {
                   for (int j = 0; j<5; j++) {
           root.add(new Button(i+" Button "+j),j,i);
   }
   }


           Scene scene = new Scene(root, 400, 200);
           primaryStage.setTitle("GridPane Demo");
           primaryStage.setScene(scene);
           primaryStage.show();
  }
}
```


GridPane Demo
0 Bu... 0 Bu... 0 Bu... 0 Bu... 0 Bu...
1 Bu... 1 Bu... 1 Bu... 1 Bu... 1 Bu...
2 Bu... 2 Bu... 2 Bu... 2 Bu... 2 Bu...
3 Bu... 3 Bu... 3 Bu... 3 Bu... 3 Bu...
4 Bu... 4 Bu... 4 Bu... 4 Bu... 4 Bu...

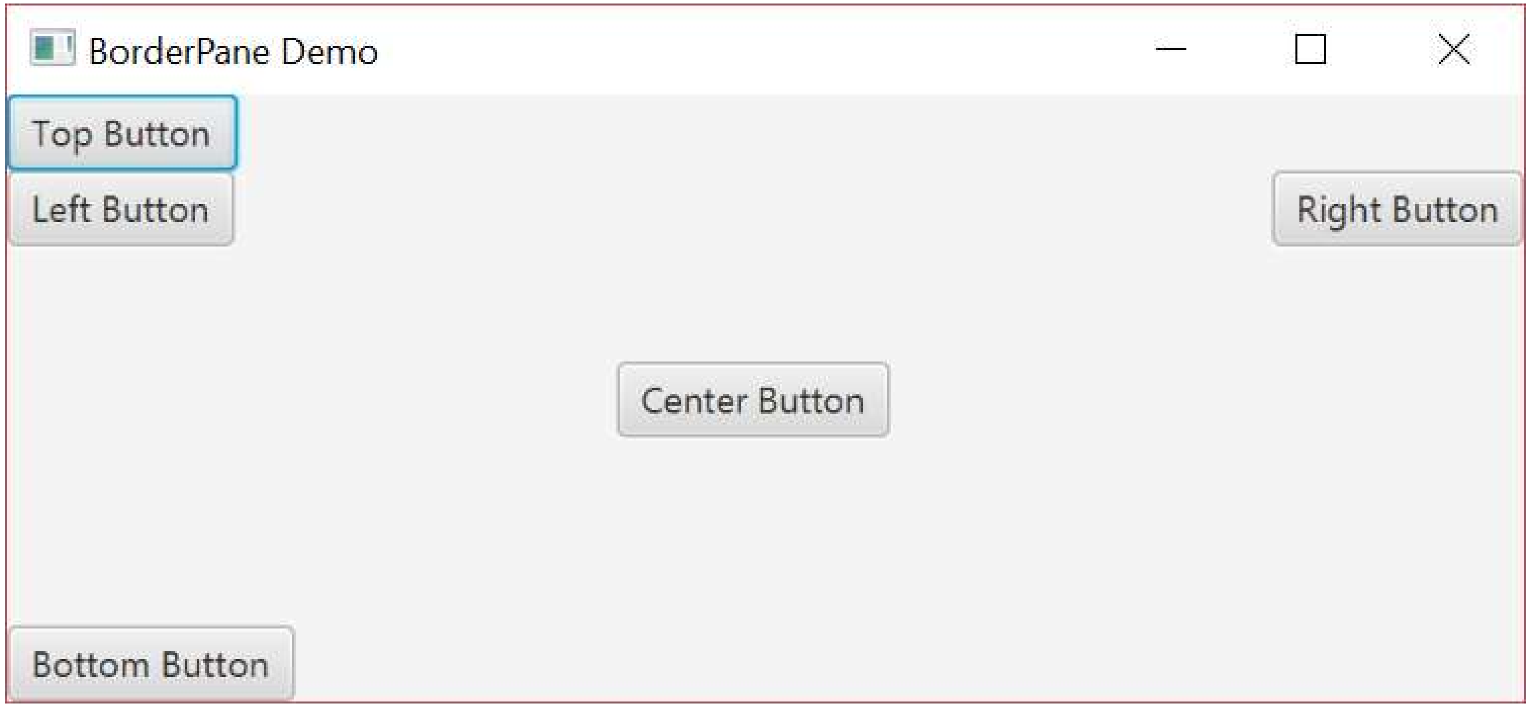
BorderPane Layout

- The BorderPane layout
  - places items into the five regions shown below.
  - Unused regions take up no space.



BorderPane Layout

- BorderPaneDemo Output



listing 7.19, class BorderPaneDemo

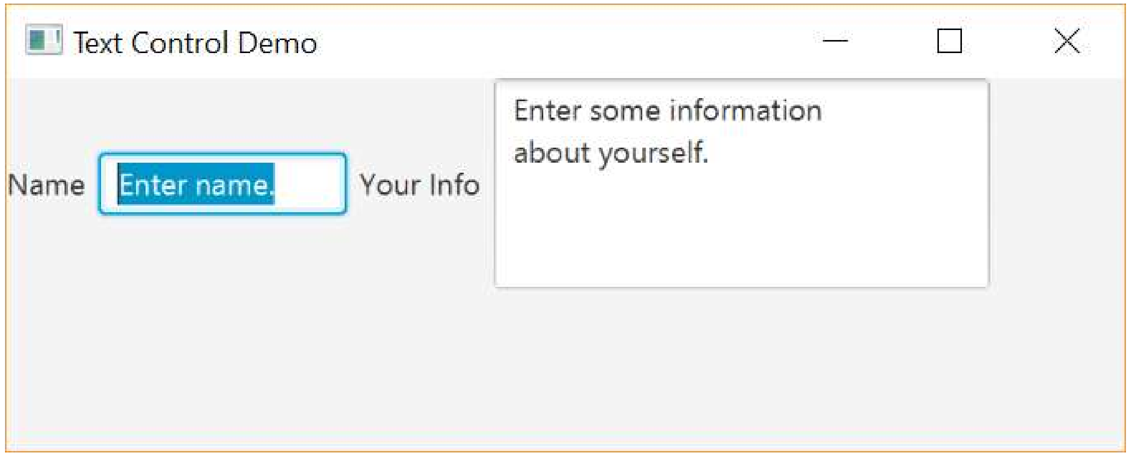
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.layout.BorderPane;
import javafx.scene.control.Button;
/**
Simple demonstration of adding buttons to the BorderPane layout.
*/
public class BorderPaneDemo extends Application
{
   public static void main(String[] args)
   {
      launch(args);
   }
@Override
  public void start(Stage primaryStage) throws Exception
  {
          BorderPane root = new BorderPane();

          root.setTop(new Button("Top Button"));
          root.setLeft(new Button("Left Button"));
          root.setCenter(new Button("Center Button"));
          root.setRight(new Button("Right Button"));
          root.setBottom(new Button("Bottom Button"));

          Scene scene = new Scene(root, 500, 200);
          primaryStage.setTitle("BorderPane Demo");
          primaryStage.setScene(scene);
          primaryStage.show();
  }
}
```

Text Areas, Text Fields

- class **`TextArea`**
  - Displayed as a place for user to enter m ultiple lines of text
- class **`TextField`**
  - Displayed as a place for user to enter a single line of text
- Can place into layouts
- listing 7.20, class TextControlDemo

listing 7.20, class TextControlDemo

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.layout.FlowPane;
import javafx.scene.control.TextField;
import javafx.scene.control.TextArea;
import javafx.scene.control.Label;
/**
Demonstration of TextField and TextArea controls.
*/
public class TextControlDemo extends Application
{
   public static void main(String[] args)
   {
      launch(args);
   }
@Override
  public void start(Stage primaryStage) throws Exception
  {
            FlowPane root = new FlowPane();
            root.setVgap(5);
            root.setHgap(5);

            // Label and textfield for name
            root.getChildren().add(new Label("Name"));
            TextField txtName = new TextField("Enter name.");
            txtName.setPrefWidth(100);
            root.getChildren().add(txtName);

            // Label and textarea for info
            root.getChildren().add(new Label("Your Info"));
            TextArea txtInfo = new TextArea(
           "Enter some information\nabout yourself.");
            txtInfo.setPrefWidth(200);
            txtInfo.setPrefRowCount(4);
            txtInfo.setPrefColumnCount(40);
            root.getChildren().add(txtInfo);

            Scene scene = new Scene(root, 450, 150);
            primaryStage.setTitle("Text Control Demo");
            primaryStage.setScene(scene);
            primaryStage.show();
  }
}
```



Text Control Demo

Name Enter name. Your Info

Enter some information
about yourself.

Combining Layout

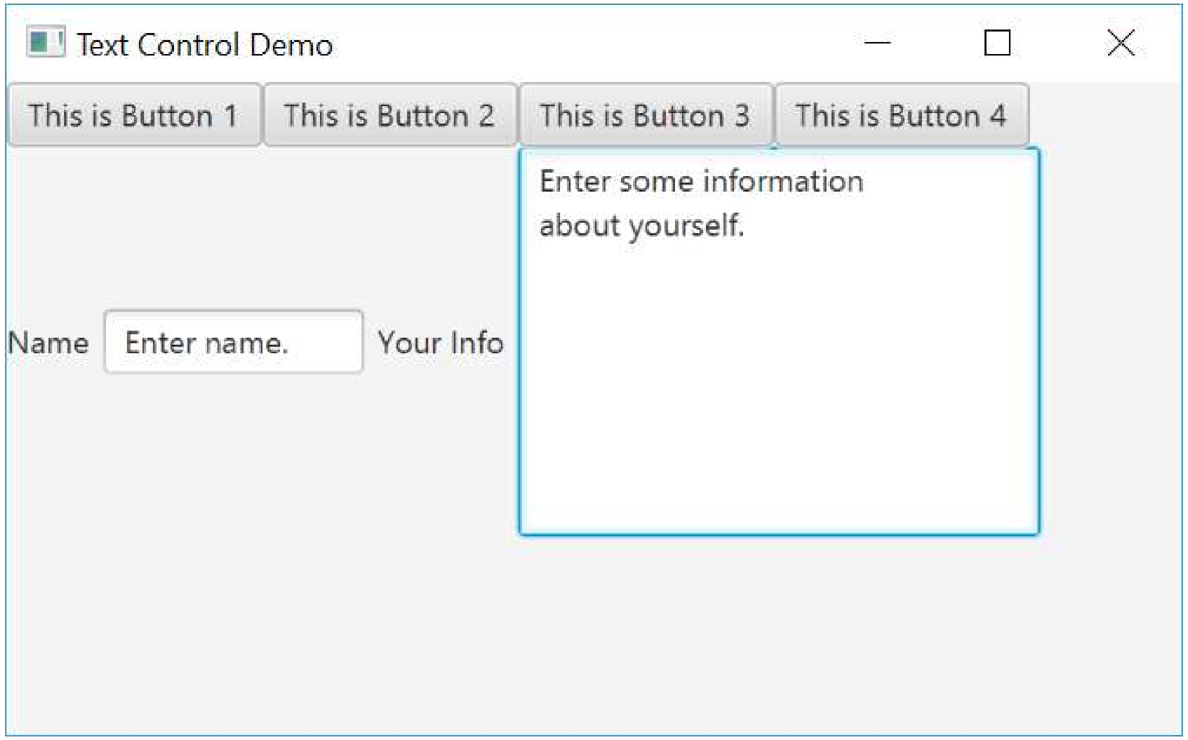
- Layouts can be added as a component inside another layout, giving you the flexibility to create sophisticated interfaces
- listing 7.21, class CombinedLayoutDe mo,
  - embeds an HBox and FlowPane into a B orderPane

listing 7.21, class CombinedLayoutDemo

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.stage.Stage;
import javafx.scene.layout.FlowPane;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.control.TextField;
import javafx.scene.control.TextArea;
import javafx.scene.control.Label;
import javafx.scene.control.Button;
/**
Embedding an HBox and FlowPane into a BorderPane.
*/
public class CombinedLayout extends Application
{
   public static void main(String[] args)
   {
      launch(args);
   }

   @Override
   public void start(Stage primaryStage) throws Exception
   {
          BorderPane root = new BorderPane();
```

```
// Create a FlowPane with a TextField and TextArea
FlowPane centerPane = new FlowPane();
centerPane.setVgap(5);
centerPane.setHgap(5);

// Label and textfield for name
centerPane.getChildren().add(new Label("Name"));
TextField txtName = new TextField("Enter name.");
txtName.setPrefWidth(100);
centerPane.getChildren().add(txtName);

// Label and textarea for info
centerPane.getChildren().add(new Label("Your Info"));
TextArea txtInfo = new TextArea(
    "Enter some information\nabout yourself.");

txtInfo.setPrefWidth(200);
txtInfo.setPrefRowCount(8);
txtInfo.setPrefColumnCount(40);

centerPane.getChildren().add(txtInfo);
```

```
        // Create an HBox with four buttons
        HBox topPane = new HBox();
        topPane.getChildren().add(new Button("This is Button 1"));
        topPane.getChildren().add(new Button("This is Button 2"));
        topPane.getChildren().add(new Button("This is Button 3"));
        topPane.getChildren().add(new Button("This is Button 4"));

        // Add the FlowPane to the center
        root.setCenter(centerPane);
        // Add the HBox to the top
        root.setTop(topPane);

        Scene scene = new Scene(root, 450, 250);
        primaryStage.setTitle("Text Control Demo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
}
```

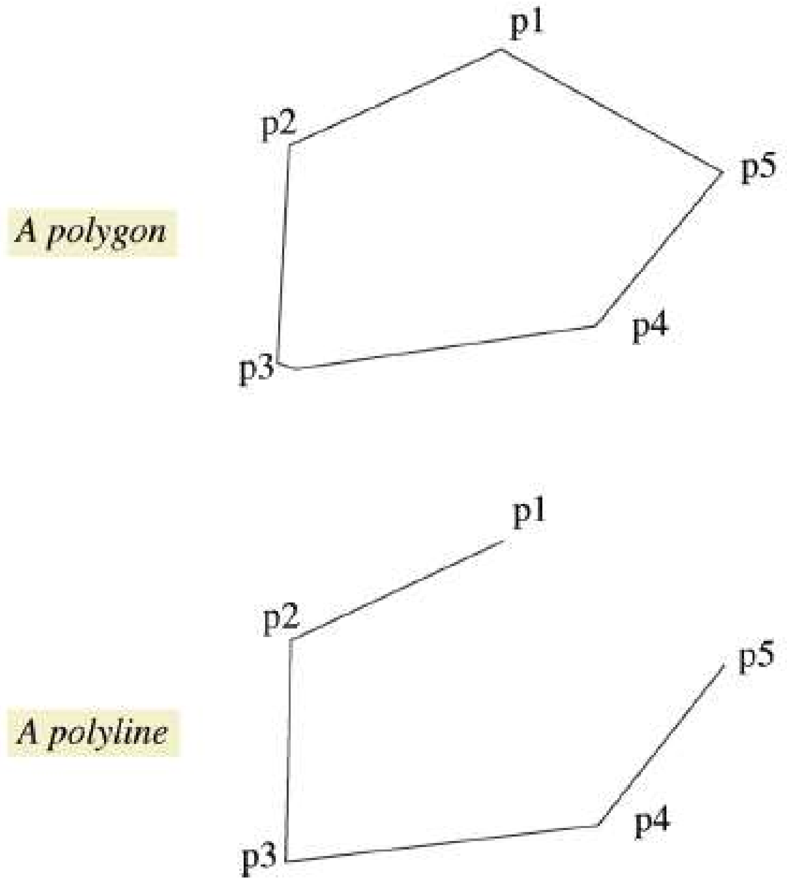

Text Control Demo
This is Button 1
This is Button 2
This is Button 3
This is Button 4
Name
Enter name.
Your Info
Enter some information
about yourself.

Drawing Polygons

- Class GraphicsContext has method **`strok eRect`**
  - But only 4 sides and only 90 degree corners
- Method **`strokePolygon`** can draw polygo n of any number of sides
  - Three arguments
    - Array of **`double`** for x values of coordinates
    - Array of **`double`** for y values of coordinates
    - Number of points (vertices)
- A *polyline* like a polygon, not closed

Drawing Polygons

- Figure 7.10
  A polygon and
  a polyline



▪listing 7.22 `class PolygonDemo`

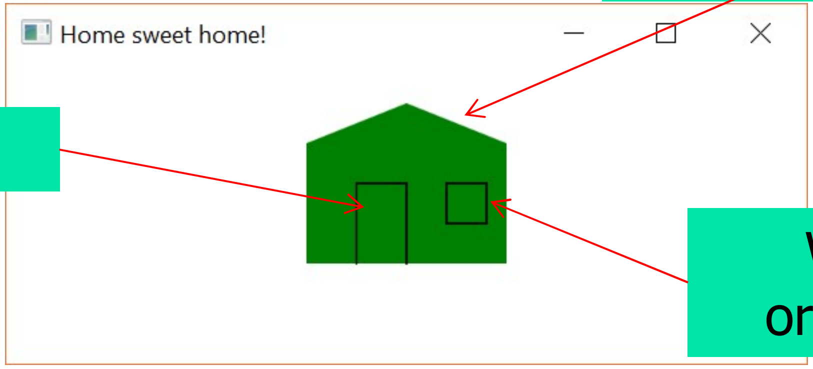
```
import javafx.application.Application;
import javafx.scene.canvas.Canvas;
import javafx.scene.Scene;
import javafx.scene.Group;
import javafx.stage.Stage;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;

public class PolygonDemo extends Application
{
   private double[] xHouse = {150, 150, 200, 250, 250};
   private double[] yHouse = {100, 40, 20, 40, 100};
   private double[] xDoor = {175, 175, 200, 200};
   private double[] yDoor = {100, 60, 60, 100};
   private double[] xWindow = {220, 220, 240, 240};
   private double[] yWindow = {60, 80, 80, 60};

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

```
 @Override
  public void start(Stage primaryStage) throws Exception
  {
          Group root = new Group();
          Scene scene = new Scene(root);

     Canvas canvas = new Canvas(400, 150);
     GraphicsContext gc = canvas.getGraphicsContext2D();

     gc.setFill(Color.GREEN);
     gc.fillPolygon(xHouse, yHouse, xHouse.length);
     gc.setFill(Color.BLACK);
     gc.strokePolyline(xDoor, yDoor, xDoor.length);
     gc.strokePolygon(xWindow, yWindow, xWindow.length);

     root.getChildren().add(canvas);
     primaryStage.setTitle("Home sweet home!");
     primaryStage.setScene(scene);
     primaryStage.show();
  }
}
```

Drawing Polygons

- listing 7.22 **`class PolygonDemo`**

House is a filled polygon



Door is a polyline

Window is an ordinary polygon

Sample screen output

끝