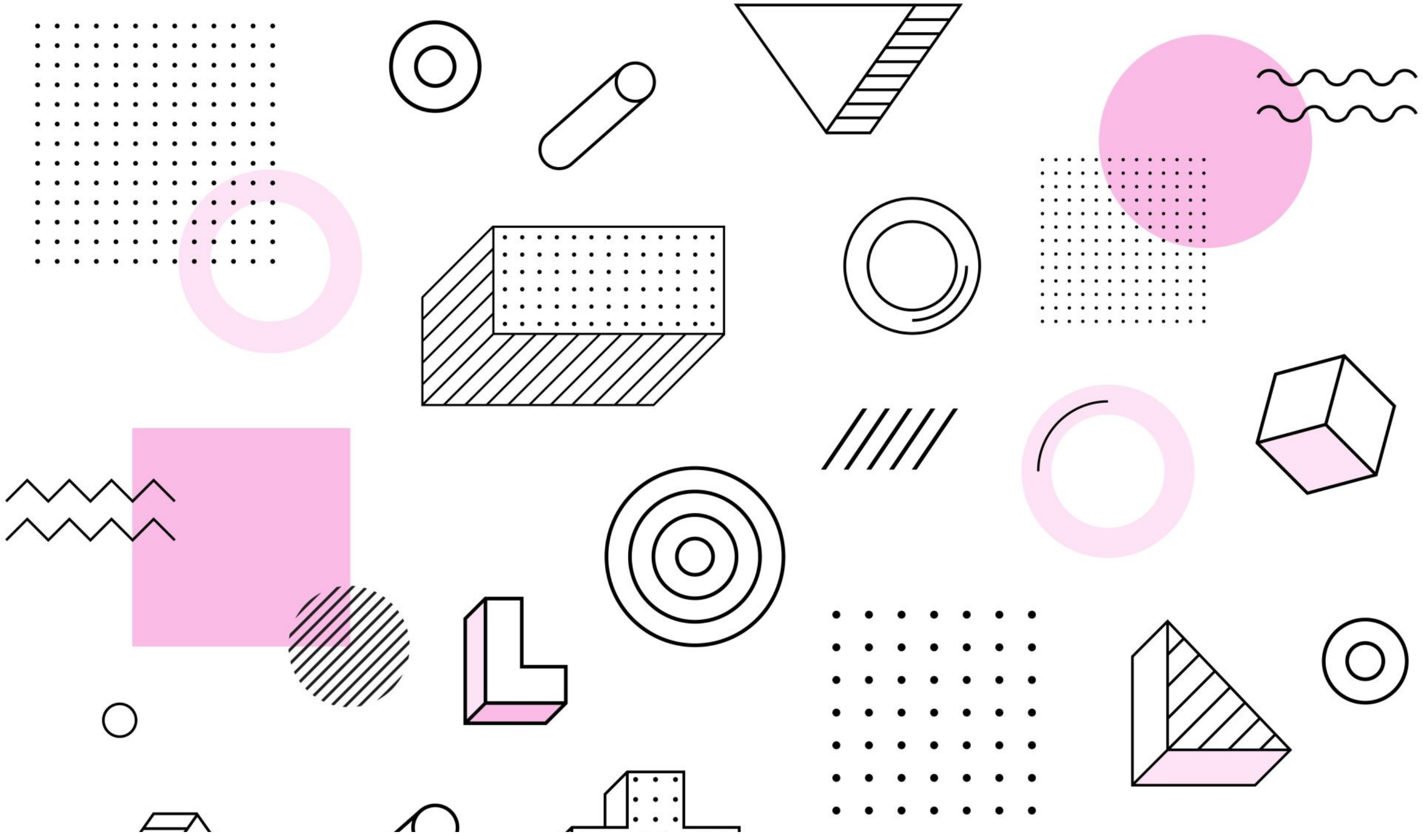


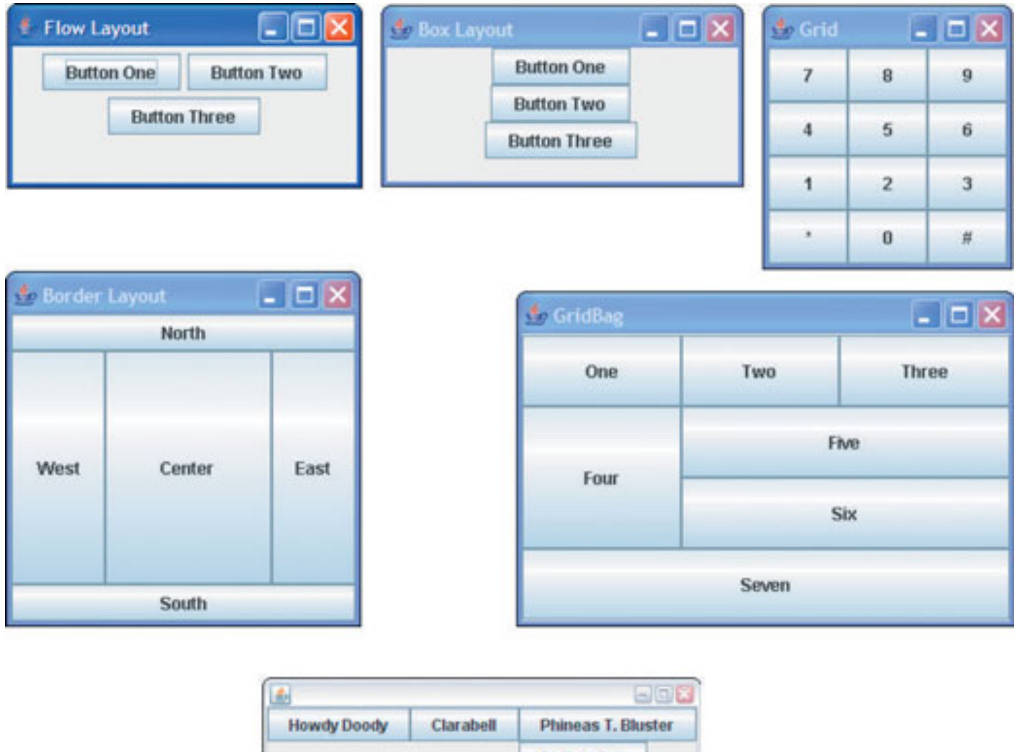
Graphicical User Interface IV

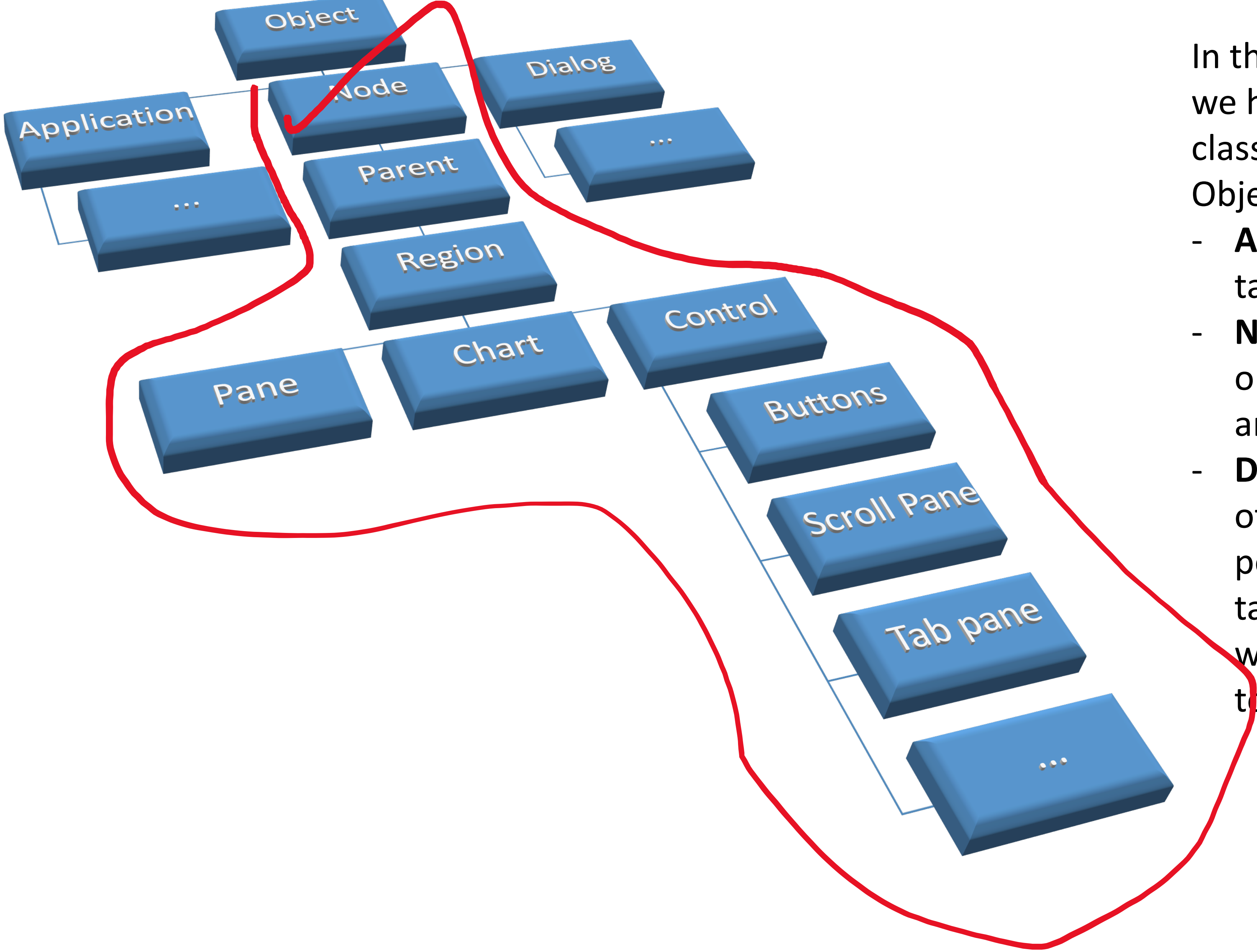
Group Layouts

Week 6 Presentation 2

In JavaFX Panes are used for Laying Out Containers

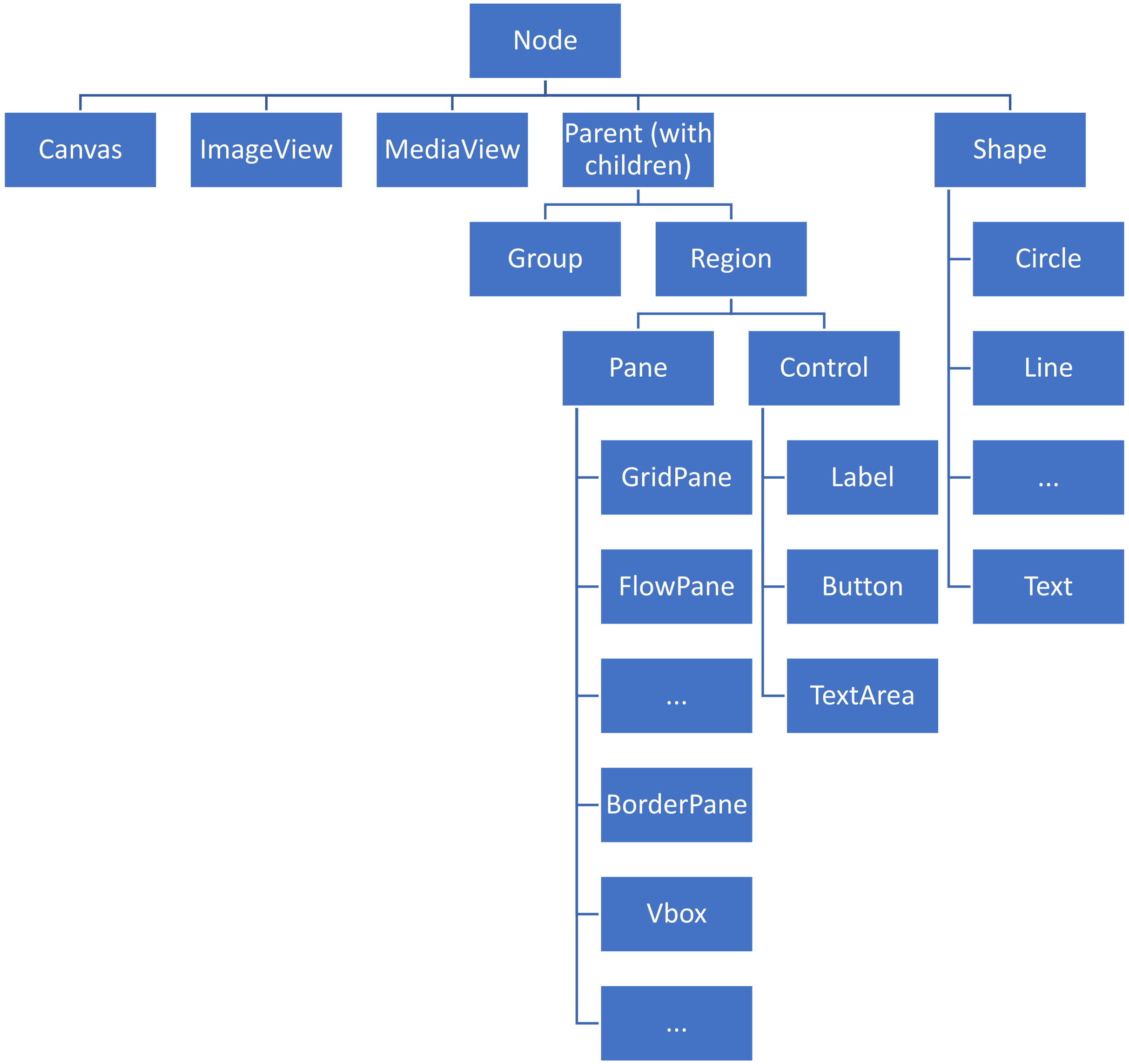
- The purpose of containers is to make creating a layout easier.
- A layout means how the various widgets are displayed on the screen in relation to each other.





In the JavaFx class hierarchy we have at the top, three classes that directly extend Object:

- **Application** (we have talked about it already),
- **Node** (basically anything on screen, visible or not) and
- **Dialog**. A Dialog is a kind of minimal application performing a specialized task (when you open a window to choose a file to open, it's a dialog).

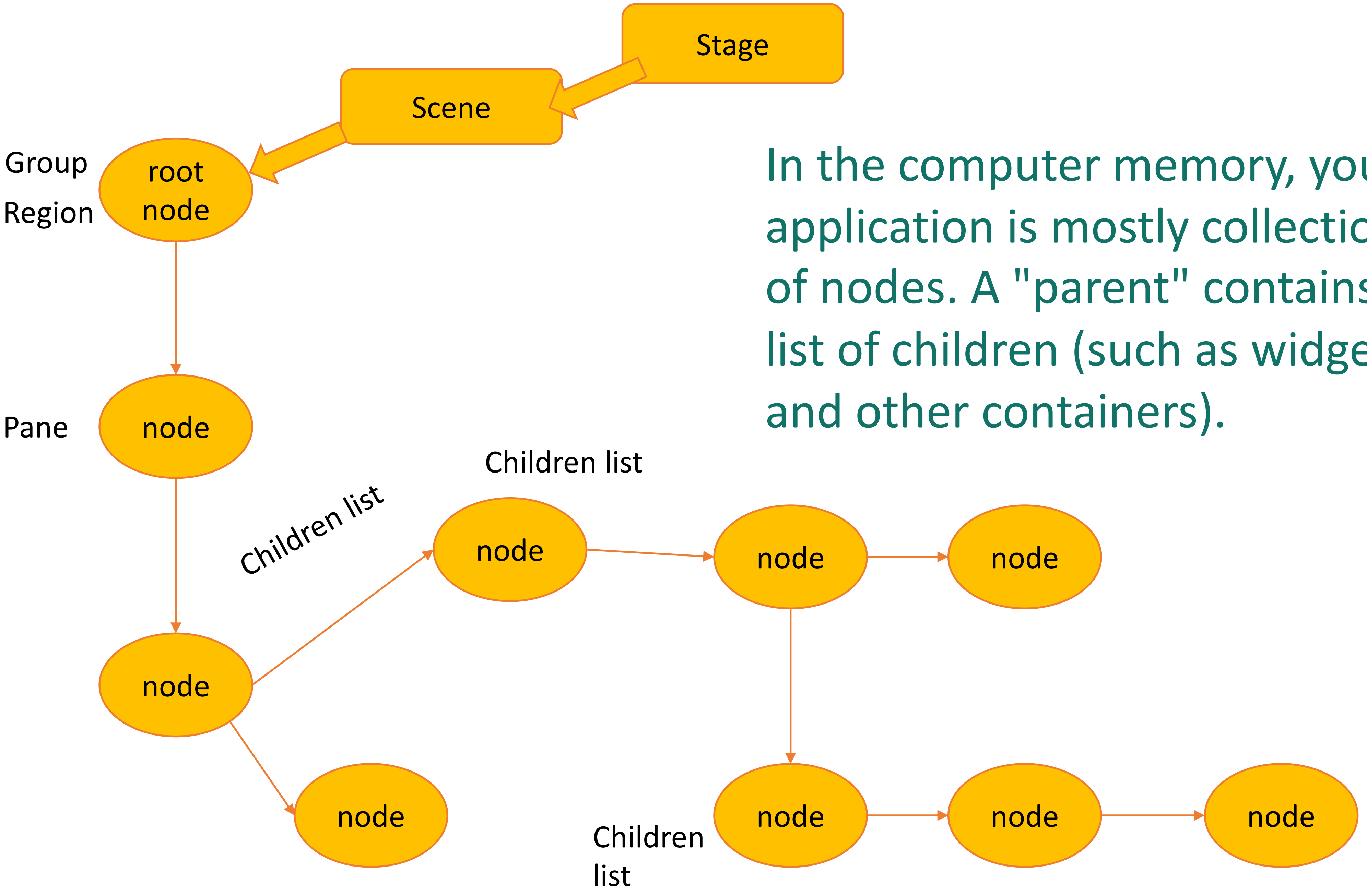


More classes that inherit from node…

JavaFX Package Hierarchy

- javafx.application
- javafx.scene
- javafx.scene.layout
- javafx.scene.control
- javafx.scene.input
- javafx.event
- javafx.geometry
- javafx.util

You also have a package hierarchy but beware that the package grouping isn't the same as the object hierarchy – grouping here is more by function than inherited methods or attributes.



In the computer memory, your application is mostly collections of nodes. A "parent" contains a list of children (such as widgets, and other containers).

Layout Panes and Groups

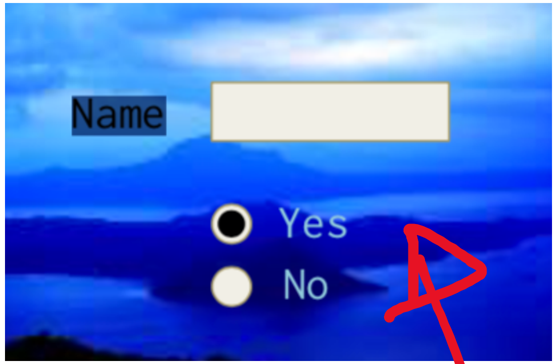
JavaFX provides many types of panes for automatically laying out nodes in a desired location and size.

Panes and groups are the containers for holding nodes. The `Group` class is often used to group nodes and to perform transformation and scale as a group. Panes and UI control objects are resizable, but group, shape, and text objects are not resizable. JavaFX provides many types of panes for organizing nodes in a container, as shown in Table .

Panes for Containing and Organizing Nodes

Class	*Description*
`Pane`	Base class for layout panes. It contains the `getChildren()` method for returning a list of nodes in the pane.
`StackPane`	Places the nodes on top of each other in the center of the pane.
`FlowPane`	Places the nodes row-by-row horizontally or column-by-column vertically.
`GridPane`	Places the nodes in the cells in a two-dimensional grid.
`BorderPane`	Places the nodes in the top, right, bottom, left, and center regions.
`HBox`	Places the nodes in a single row.
`VBox`	Places the nodes in a single column.

More advanced types of panes can also be added later



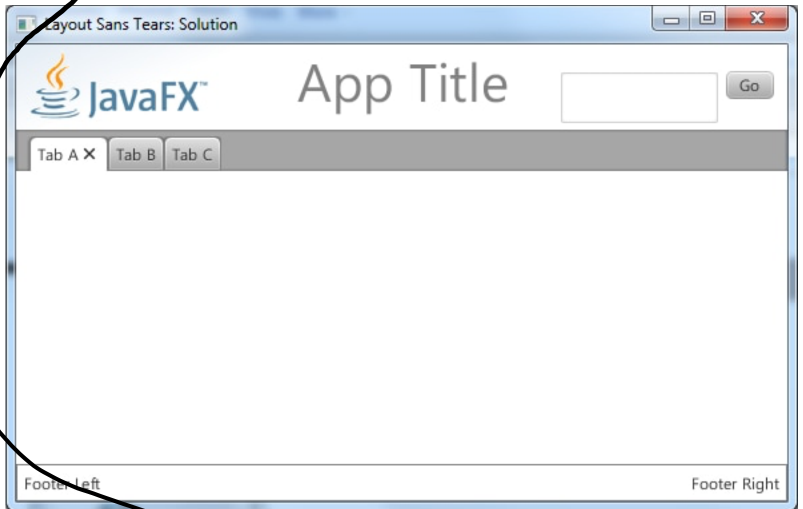
Most often your main Window will be one of these.

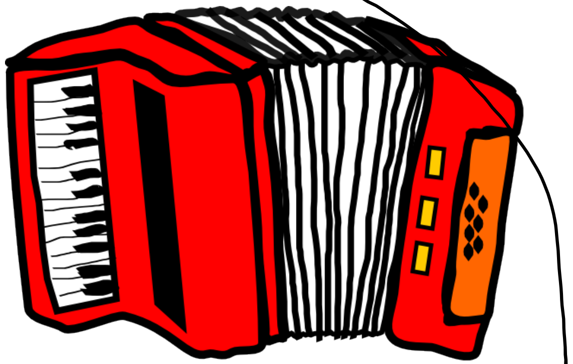
The StackPane allows to have elements on top of each other, which is mostly interesting for background images.

More Sophisticated Types of Panes Can be Added Afterwards

- AnchorPane
- ScrollPane
- SplitPane
- TabPane
- TitledPane (Accordion)

Controls



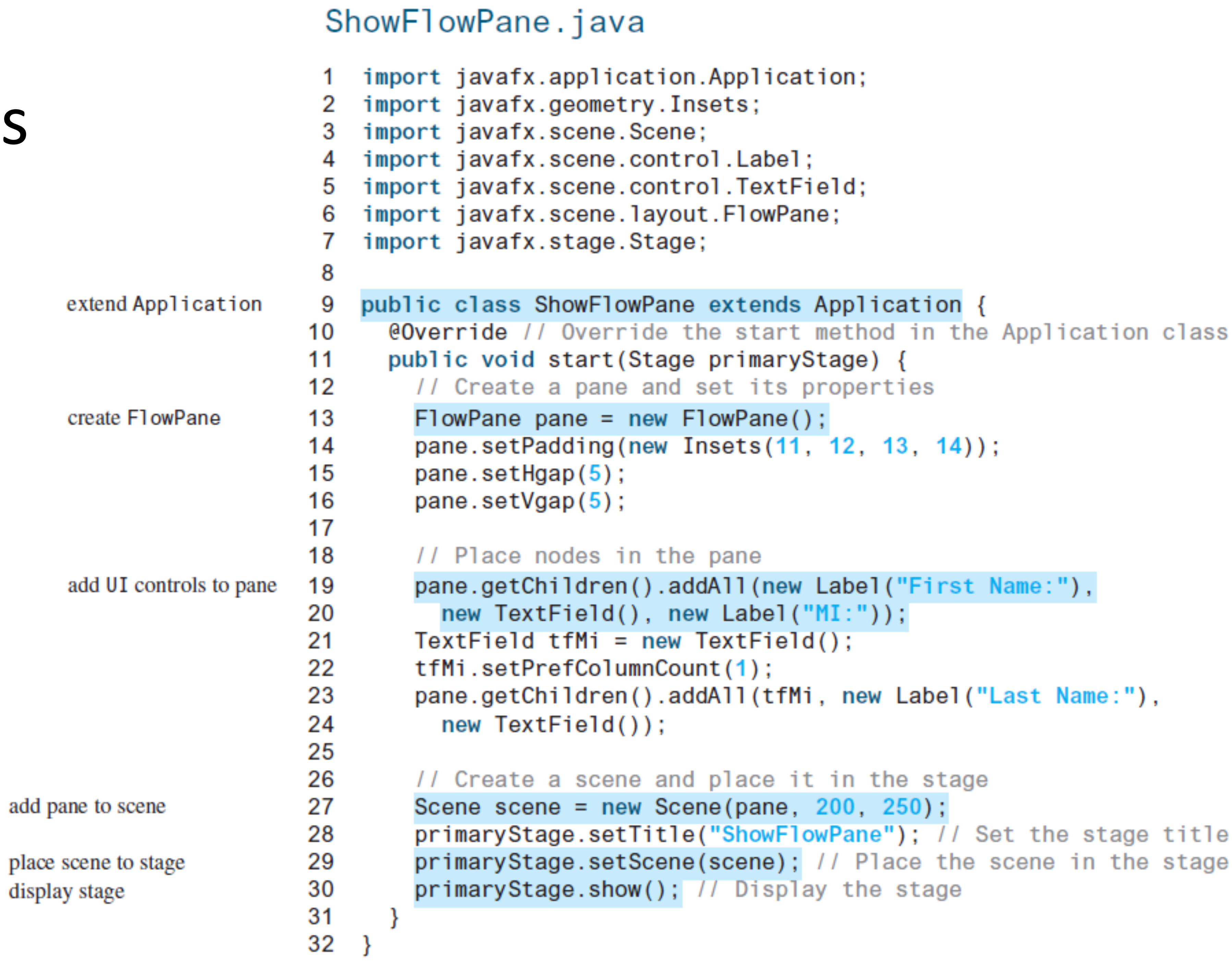


Typical Design

Here is a basic start method for a javaFX program

```
public static void start(Stage stage) {
        stage.setTitle("Window Title");
        Group root = new Group();
        Scene scene = new Scene(root);
        BorderPane pane = new BorderPane();
        root.getChildren().add(pane);

        // Add containers and widgets to pane

        stage.setScene(scene);
        stage.show();
```

Panes



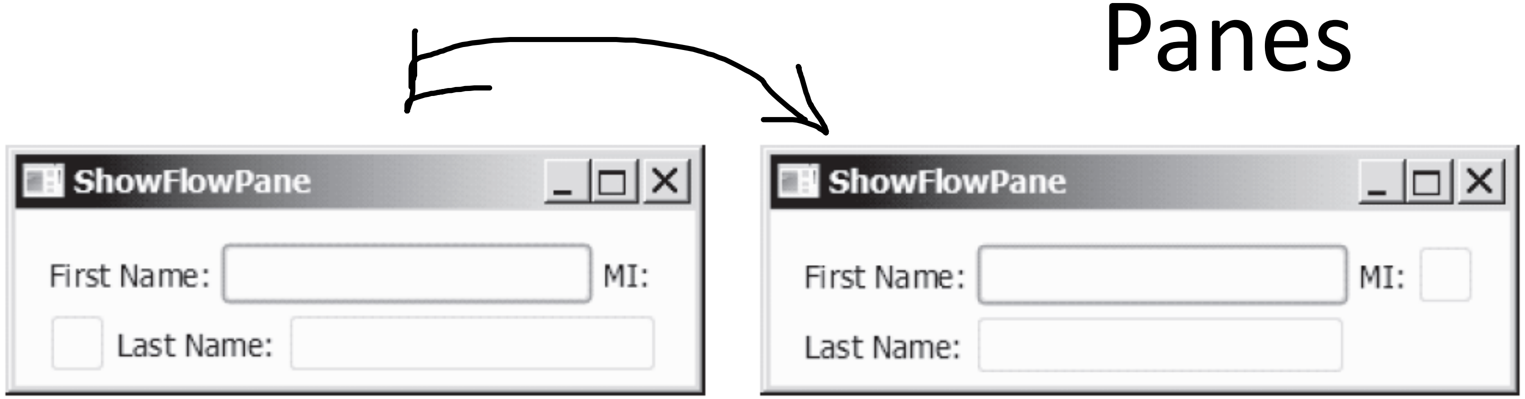
ShowFlowPane.java

```java
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class ShowFlowPane extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane and set its properties
    FlowPane pane = new FlowPane();
    pane.setPadding(new Insets(11, 12, 13, 14));
    pane.setHgap(5);
    pane.setVgap(5);

    // Place nodes in the pane
    pane.getChildren().addAll(new Label("First Name:"),
      new TextField(), new Label("MI:"));
    TextField tfMi = new TextField();
    tfMi.setPrefColumnCount(1);
    pane.getChildren().addAll(tfMi, new Label("Last Name:"),
      new TextField());

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 200, 250);
    primaryStage.setTitle("ShowFlowPane"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
}
```

Panes

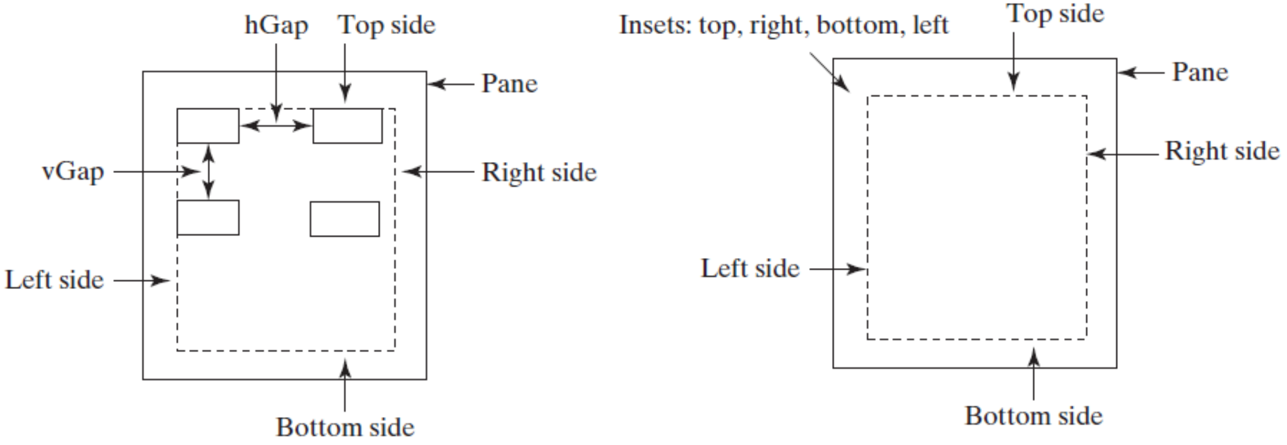
owFlowPane.java

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class ShowFlowPane extends Applicat
  @Override // Override the start method 
  public void start(Stage primaryStage) {
    // Create a pane and set its propertie
    FlowPane pane = new FlowPane();
    pane.setPadding(new Insets(11, 12, 13, 14));
    pane.setHgap(5);
    pane.setVgap(5);

    // Place nodes in the pane
    pane.getChildren().addAll(new Label("First Name:"),
      new TextField(), new Label("MI:"));
    TextField tfMi = new TextField();
    tfMi.setPrefColumnCount(1);
    pane.getChildren().addAll(tfMi, new Label("Last Name
      new TextField());

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 200, 250);
    primaryStage.setTitle("ShowFlowPane"); // Set the st
    primaryStage.setScene(scene); // Place the scene in
    primaryStage.show(); // Display the stage
  }
}
```



(a) (b)

The nodes fill in the rows in the FlowPane one after another.



You can specify hGap and vGap between the nodes in a FlowLPane.

Panes

ShowGridPane.java

```
import javafx.application.Application;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
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 ShowGridPane extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane and set its properties
    GridPane pane = new GridPane();
    pane.setAlignment(Pos.CENTER);
    pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
    pane.setHgap(5.5);
    pane.setVgap(5.5);

    // Place nodes in the pane
    pane.add(new Label("First Name:"), 0, 0);
    pane.add(new TextField(), 1, 0);
    pane.add(new Label("MI:"), 0, 1);
    pane.add(new TextField(), 1, 1);
    pane.add(new Label("Last Name:"), 0, 2);
    pane.add(new TextField(), 1, 2);
    Button btAdd = new Button("Add Name");
    pane.add(btAdd, 1, 3);
    GridPane.setHalignment(btAdd, HPos.RIGHT);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowGridPane"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
}
```

create a grid pane

set properties

add label

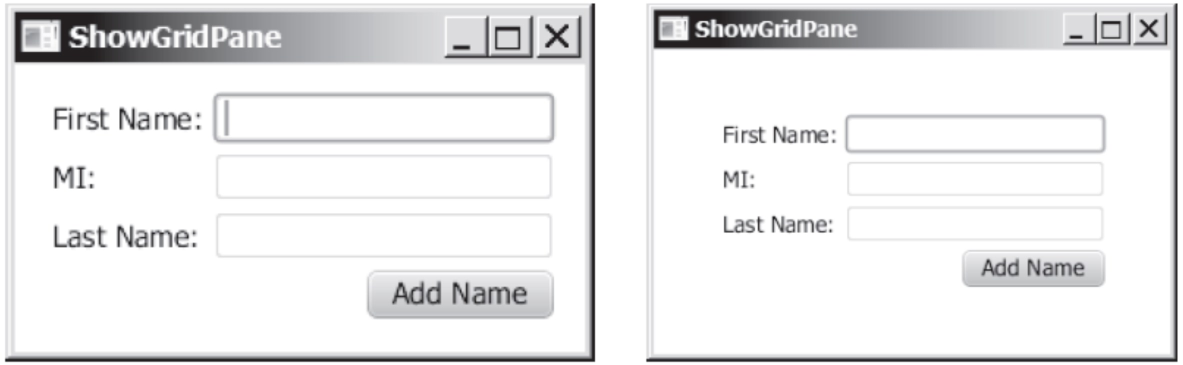
add text field

add button

align button right

create a scene

display stage



The GridPane places the nodes in a grid with a specified column and row indices.

Panes

ShowBorderPane.java

```java
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class ShowBorderPane extends Appli
  @Override // Override the start method
  public void start(Stage primaryStage) {
    // Create a border pane
    BorderPane pane = new BorderPane();

    // Place nodes in the pane
    pane.setTop(new CustomPane("Top"));
    pane.setRight(new CustomPane("Right"));
    pane.setBottom(new CustomPane("Bottom"));
    pane.setLeft(new CustomPane("Left"));
    pane.setCenter(new CustomPane("Center"));

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowBorderPane"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
}

// Define a custom pane to hold a label in the center of the pane
class CustomPane extends StackPane {
  public CustomPane(String title) {
    getChildren().add(new Label(title));
    setStyle("-fx-border-color: red");
    setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
  }
}
```

create a border pane

add to top

add to right

add to bottom

add to left

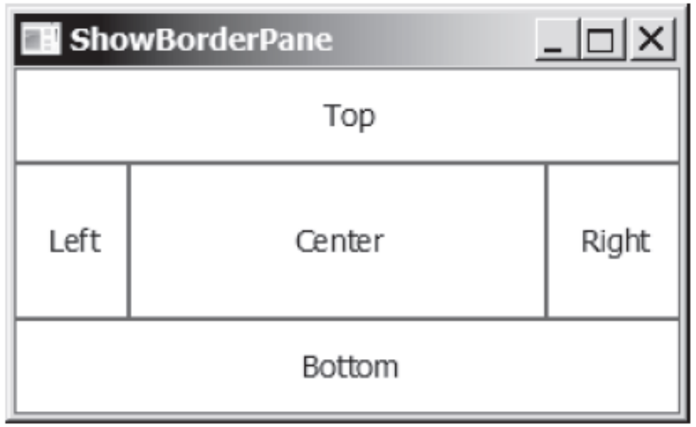
add to center

define a custom pane

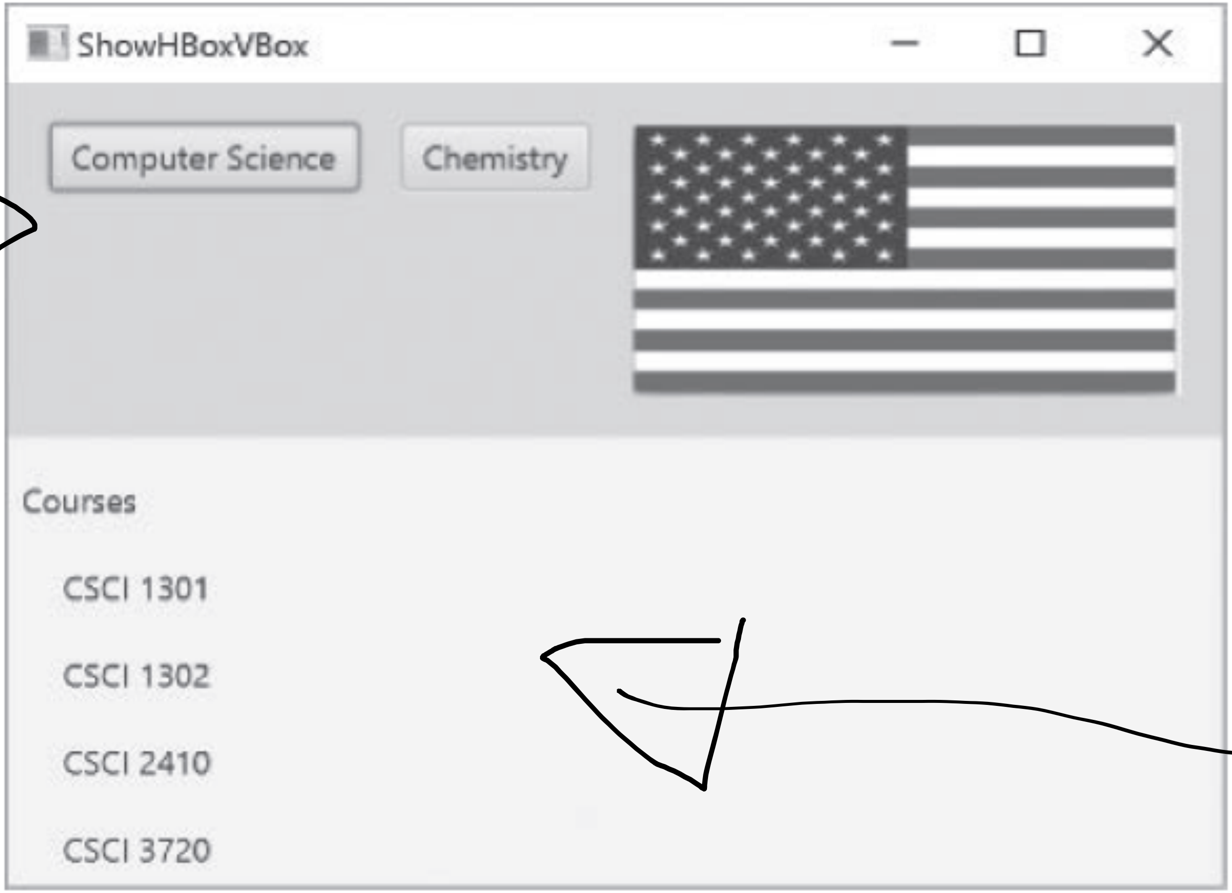
add a label to pane

set style

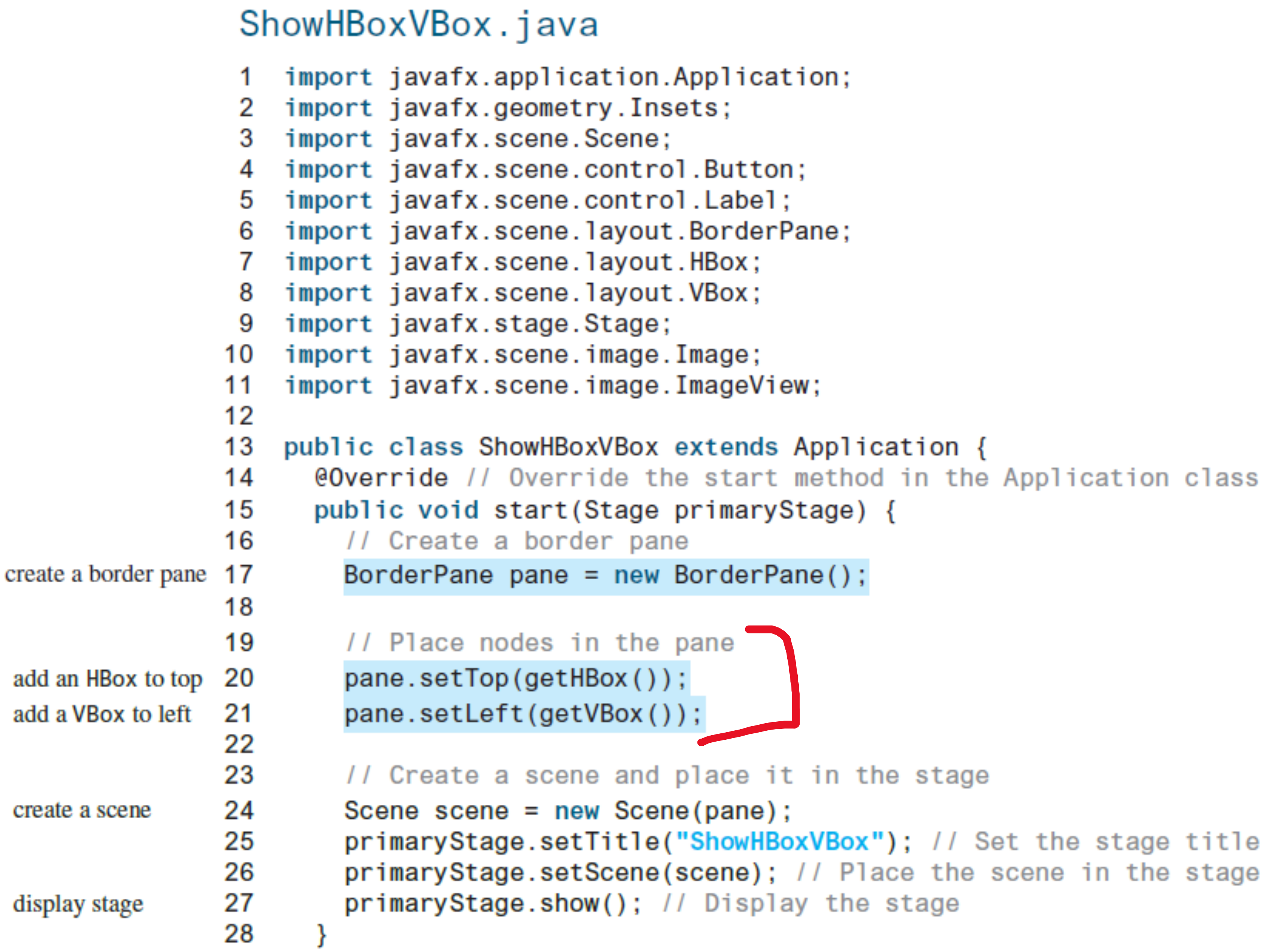
set padding



The BorderPane places the nodes in five regions of the pane.



The HBox places the nodes in one row, and the VBox places the nodes in one column.



ShowHBoxVBox.java

```java
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

public class ShowHBoxVBox extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a border pane
    BorderPane pane = new BorderPane();

    // Place nodes in the pane
    pane.setTop(getHBox());
    pane.setLeft(getVBox());

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowHBoxVBox"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
```

create a border pane

add an HBox to top

add a VBox to left

create a scene

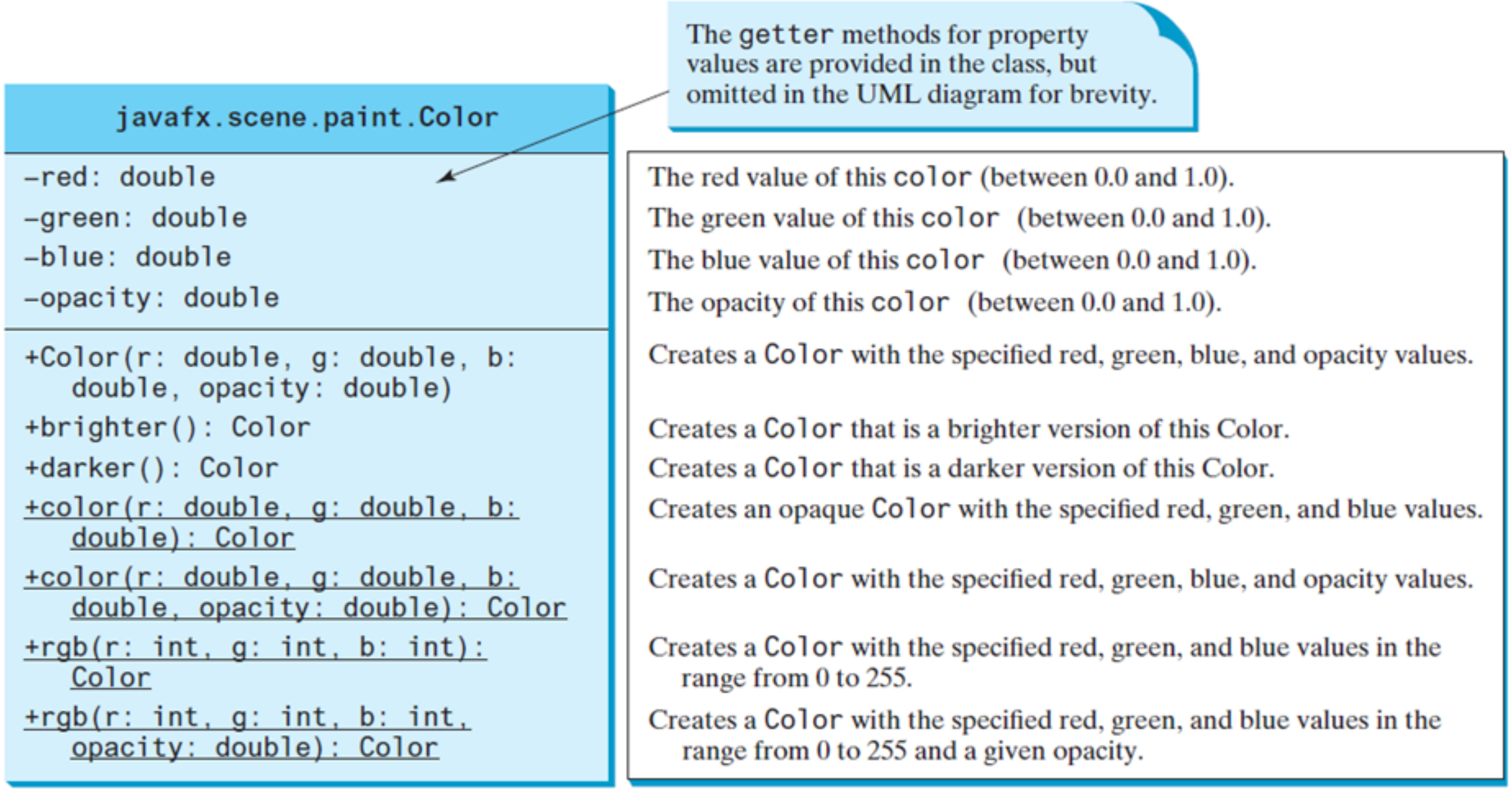
display stage

```
                     
getHBox                private HBox getHBox() {
                         HBox hBox = new HBox(15);
                         hBox.setPadding(new Insets(15, 15, 15, 15));
                         hBox.setStyle("-fx-background-color: gold");
add buttons to HBox      hBox.getChildren().add(new Button("Computer Science"));
                         hBox.getChildren().add(new Button("Chemistry"));
                         ImageView imageView = new ImageView(new Image("image/us.gif"));
                         hBox.getChildren().add(imageView);
return an HBox           return hBox;
                       }

getVBox                private VBox getVBox() {
                         VBox vBox = new VBox(15);
                         vBox.setPadding(new Insets(15, 5, 5, 5));
add a label              vBox.getChildren().add(new Label("Courses"));

                         Label[] courses = {new Label("CSCI 1301"), new Label("CSCI 1302"),
                             new Label("CSCI 2410"), new Label("CSCI 3720")};

                         for (Label course: courses) {
set margin                 VBox.setMargin(course, new Insets(0, 0, 0, 15));
add a label                vBox.getChildren().add(course);
                         }

return vBox              return vBox;
                       }
                     }
```

The Color Class

The Color class can be used to create colors.

JavaFX defines the abstract Paint class for painting a node. The javafx.scene.paint.Color is a concrete subclass of Paint .



The getter methods for property values are provided in the class, but omitted in the UML diagram for brevity.

```
javafx.scene.paint.Color
-red: double
-green: double
-blue: double
-opacity: double
+Color(r: double, g: double, b:
   double, opacity: double)
+brighter(): Color
+darker(): Color
+color(r: double, g: double, b:
   double): Color
+color(r: double, g: double, b:
   double, opacity: double): Color
+rgb(r: int, g: int, b: int):
   Color
+rgb(r: int, g: int, b: int,
   opacity: double): Color
```

The red value of this color (between 0.0 and 1.0).

The green value of this color (between 0.0 and 1.0).

The blue value of this color (between 0.0 and 1.0).

The opacity of this color (between 0.0 and 1.0).

Creates a Color with the specified red, green, blue, and opacity values.

Creates a Color that is a brighter version of this Color.

Creates a Color that is a darker version of this Color.

Creates an opaque Color with the specified red, green, and blue values.

Creates a Color with the specified red, green, blue, and opacity values.

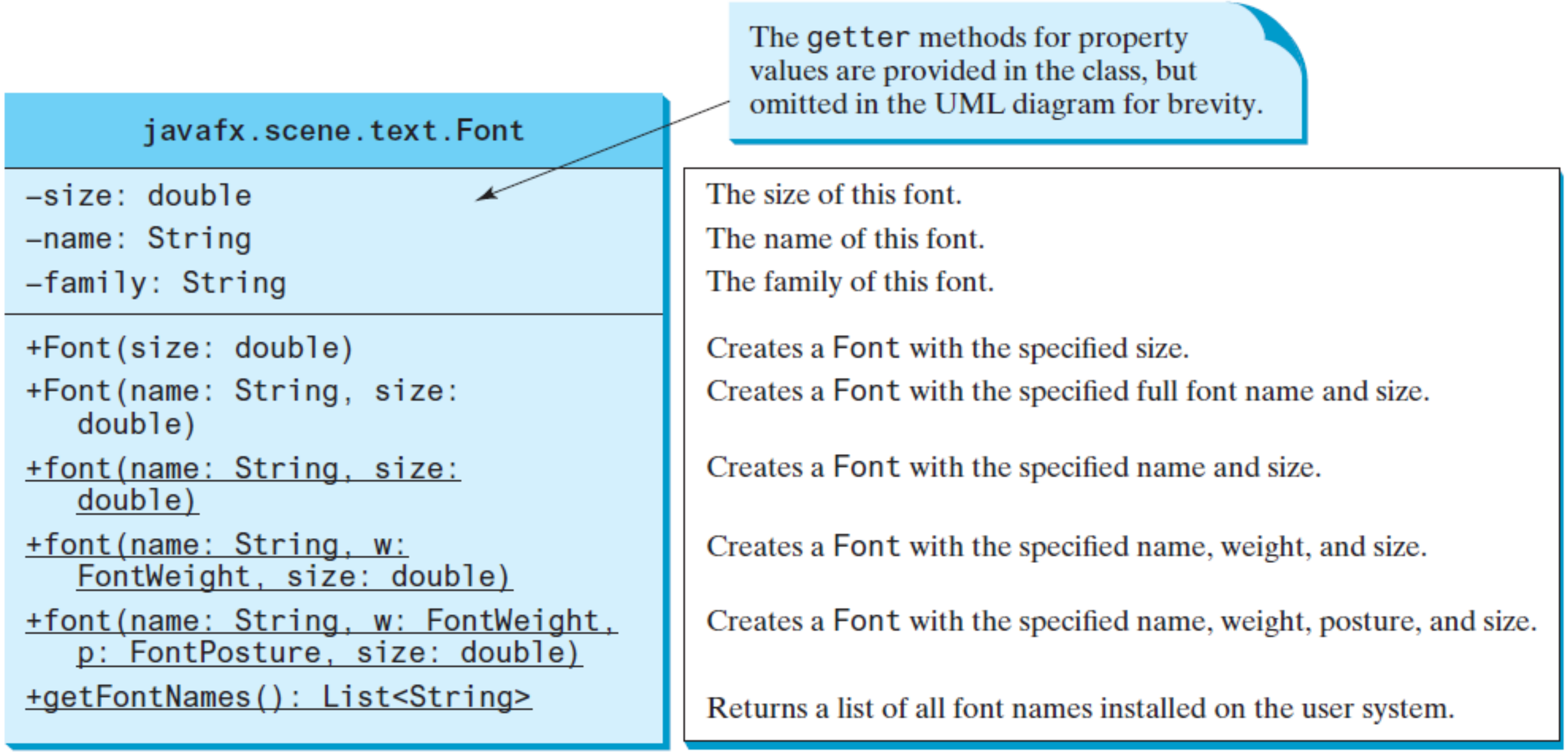
Creates a Color with the specified red, green, and blue values in the range from 0 to 255.

Creates a Color with the specified red, green, and blue values in the range from 0 to 255 and a given opacity.

The Font Class

A Font *describes font name, weight, and size.*

```
Font font1 = new Font("SansSerif", 16);
Font font2 = Font.font("Times New Roman", FontWeight.BOLD,
  FontPosture.ITALIC, 12);
```



The getter methods for property values are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.text.Font

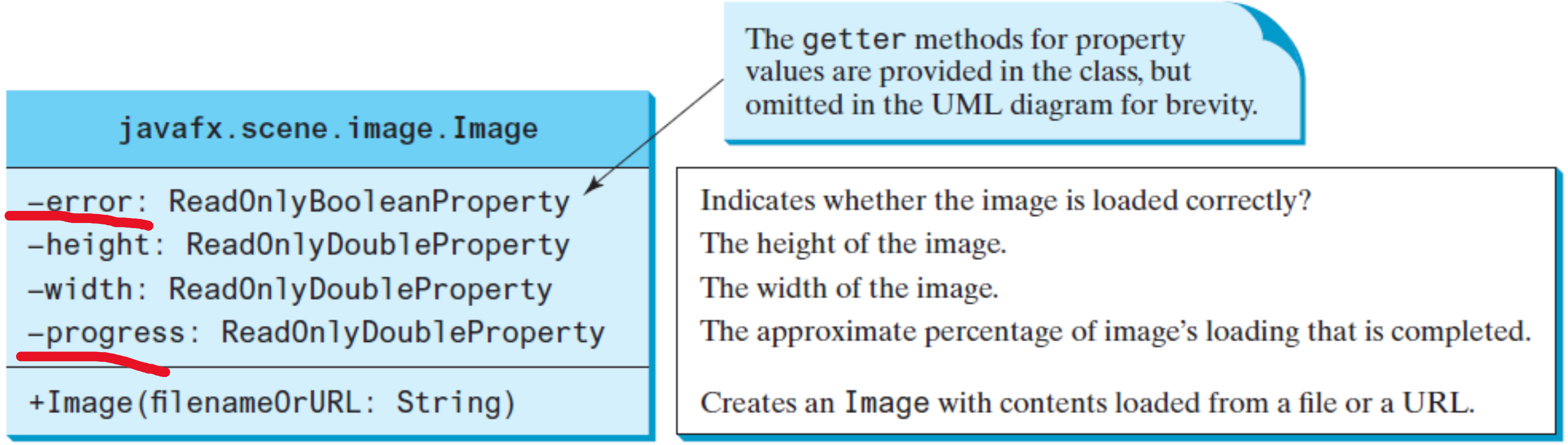
-size: double	The size of this font.
-name: String	The name of this font.
-family: String	The family of this font.
+Font(size: double)	Creates a Font with the specified size.
+Font(name: String, size: double)	Creates a Font with the specified full font name and size.
+font(name: String, size: double)	Creates a Font with the specified name and size.
+font(name: String, w: FontWeight, size: double)	Creates a Font with the specified name, weight, and size.
+font(name: String, w: FontWeight, p: FontPosture, size: double)	Creates a Font with the specified name, weight, posture, and size.
+getFontNames(): List<String>	Returns a list of all font names installed on the user system.

The **Image** and **ImageView** Classes

The Image *class represents a graphical image, and the* ImageView *class can be used to display an image.*

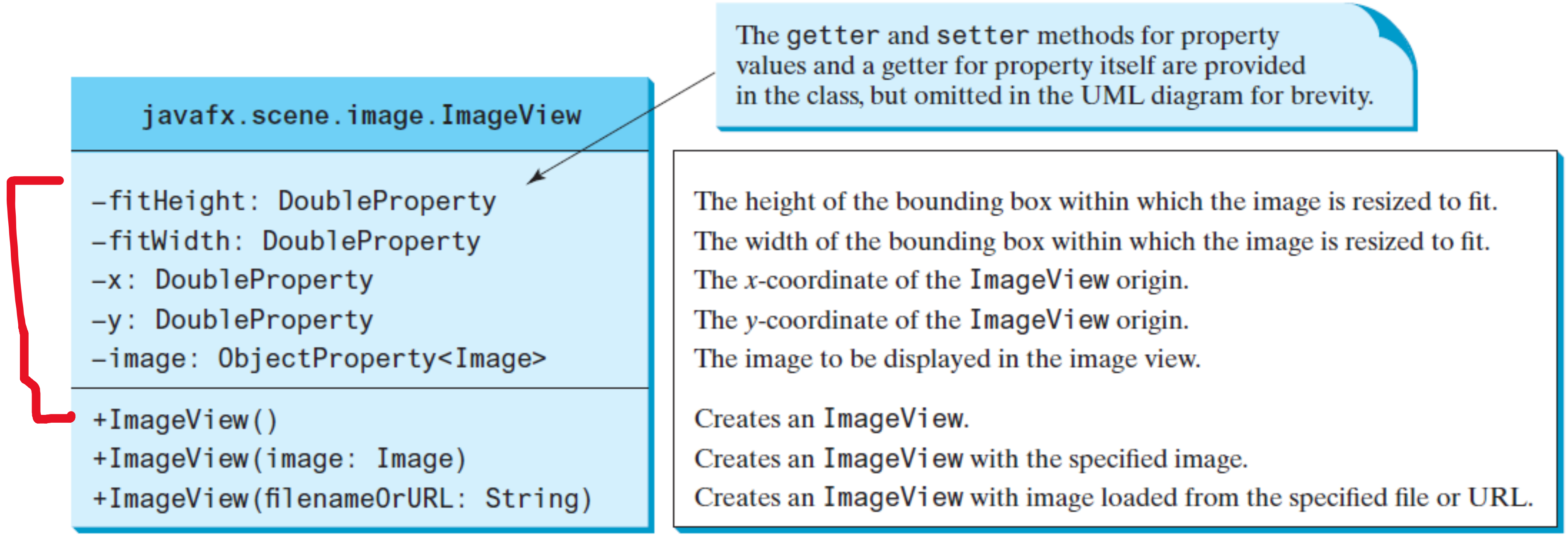
```
Image image = new Image("image/us.gif");
ImageView imageView = new ImageView(image);
```

```
ImageView imageView = new ImageView("image/us.gif");
```



The getter methods for property values are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.image.Image	
-error: ReadOnlyBooleanProperty	Indicates whether the image is loaded correctly?
-height: ReadOnlyDoubleProperty	The height of the image.
-width: ReadOnlyDoubleProperty	The width of the image.
-progress: ReadOnlyDoubleProperty	The approximate percentage of image's loading that is completed.
+Image(filenameOrURL: String)	Creates an Image with contents loaded from a file or a URL.



The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.image.ImageView	
-fitHeight: DoubleProperty	The height of the bounding box within which the image is resized to fit.
-fitWidth: DoubleProperty	The width of the bounding box within which the image is resized to fit.
-x: DoubleProperty	The *x*-coordinate of the ImageView origin.
-y: DoubleProperty	The *y*-coordinate of the ImageView origin.
-image: ObjectProperty<Image>	The image to be displayed in the image view.
+ImageView()	Creates an ImageView.
+ImageView(image: Image)	Creates an ImageView with the specified image.
+ImageView(filenameOrURL: String)	Creates an ImageView with image loaded from the specified file or URL.

ShowImage.java

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

public class ShowImage extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    // Create a pane to hold the image views
    Pane pane = new HBox(10);                                    create an HBox
    pane.setPadding(new Insets(5, 5, 5, 5));
    Image image = new Image("image/us.gif");                     create an image
    pane.getChildren().add(new ImageView(image));                add an image view to pane

    ImageView imageView2 = new ImageView(image);                 create an image view
    imageView2.setFitHeight(100);                                set image view properties
    imageView2.setFitWidth(100);
    pane.getChildren().add(imageView2);                          add an image to pane

    ImageView imageView3 = new ImageView(image);                 create an image view
    imageView3.setRotate(90);                                    rotate an image view
    pane.getChildren().add(imageView3);                          add an image to pane

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane);
    primaryStage.setTitle("ShowImage"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
}
```

Next

- More widgets and other components for controlling interactions…