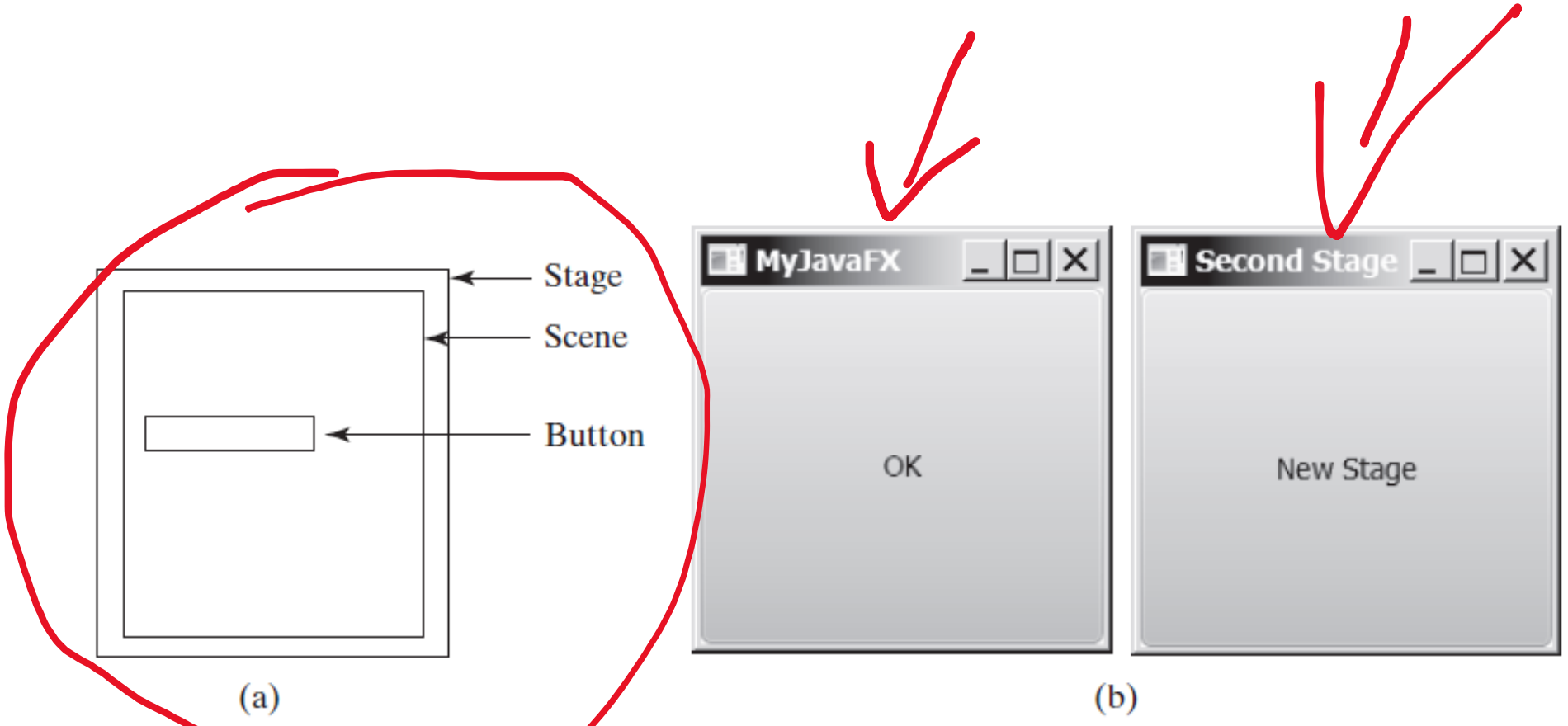
Graphical User Interfaces III

Multiple Stages,Coordinate Containers, Actions and Events

Week 6 Presentation 1

Overview

- Last week, in GUI I and II, concepts that are important for designing GUIs and using JavaFX were introduced
- This presentation extends these concepts to demonstrate the use of “stages” and “containers” and other elements of JavaFX more generally
- The use of different types tools to support developing graphical user interfaces is touched upon at the end (interface design tools – e.g. scene builder)



(a) Stage is a window for displaying a scene that contains nodes.
(b) Multiple stages can be displayed in a JavaFX program.

MultipleStageDemo.java

primary stage in start

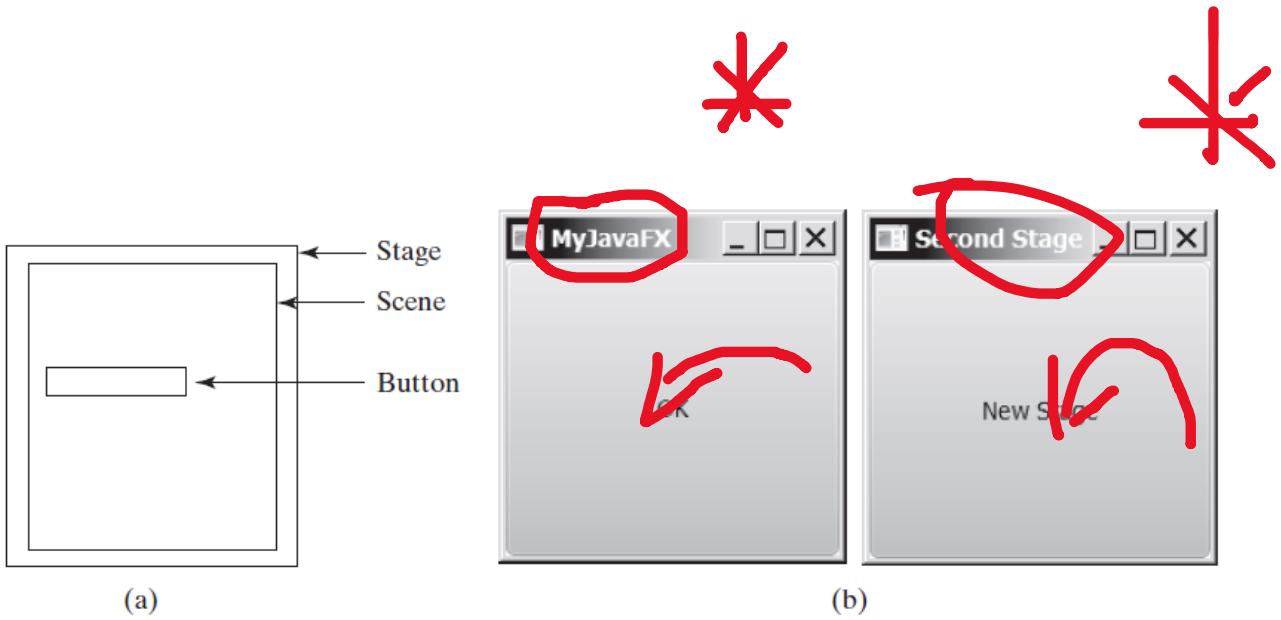
display primary stage

create second stage

display second stage

main method omitted

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.stage.Stage;
5
6  public class MultipleStageDemo extends Application {
7    @Override // Override the start method in the Application class
8    public void start(Stage primaryStage) {
9      // Create a scene and place a button in the scene
10     Scene scene = new Scene(new Button("OK"), 200, 250);
11     primaryStage.setTitle("MyJavaFX"); // Set the stage title
12     primaryStage.setScene(scene); // Place the scene in the stage
13     primaryStage.show(); // Display the stage
14
15     Stage stage = new Stage(); // Create a new stage
16     stage.setTitle("Second Stage"); // Set the stage title
17     // Set a scene with a button in the stage
18     stage.setScene(new Scene(new Button("New Stage"), 200, 250));
19     stage.show(); // Display the stage
20   }
21 }
```



(a) Stage is a window for displaying a scene that contains nodes.
(b) Multiple stages can be displayed in a JavaFX program.

MultipleStageDemo.java

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.stage.Stage;
5
6  public class MultipleStageDemo extends Application {
7    @Override // Override the start method in the Application class
8    public void start(Stage primaryStage) {
9      // Create a scene and place a button in the scene
10     Scene scene = new Scene(new Button("OK"), 200, 250);
11     primaryStage.setTitle("MyJavaFX"); // Set the stage title
12     primaryStage.setScene(scene); // Place the scene in the stage
13     primaryStage.show(); // Display the stage
14
15     Stage stage = new Stage(); // Create a new stage
16     stage.setTitle("Second Stage"); // Set the stage title
17     // Set a scene with a button in the stage
18     stage.setScene(new Scene(new Button("New Stage"), 200, 250));
19     stage.show(); // Display the stage
20   }
21 }
```

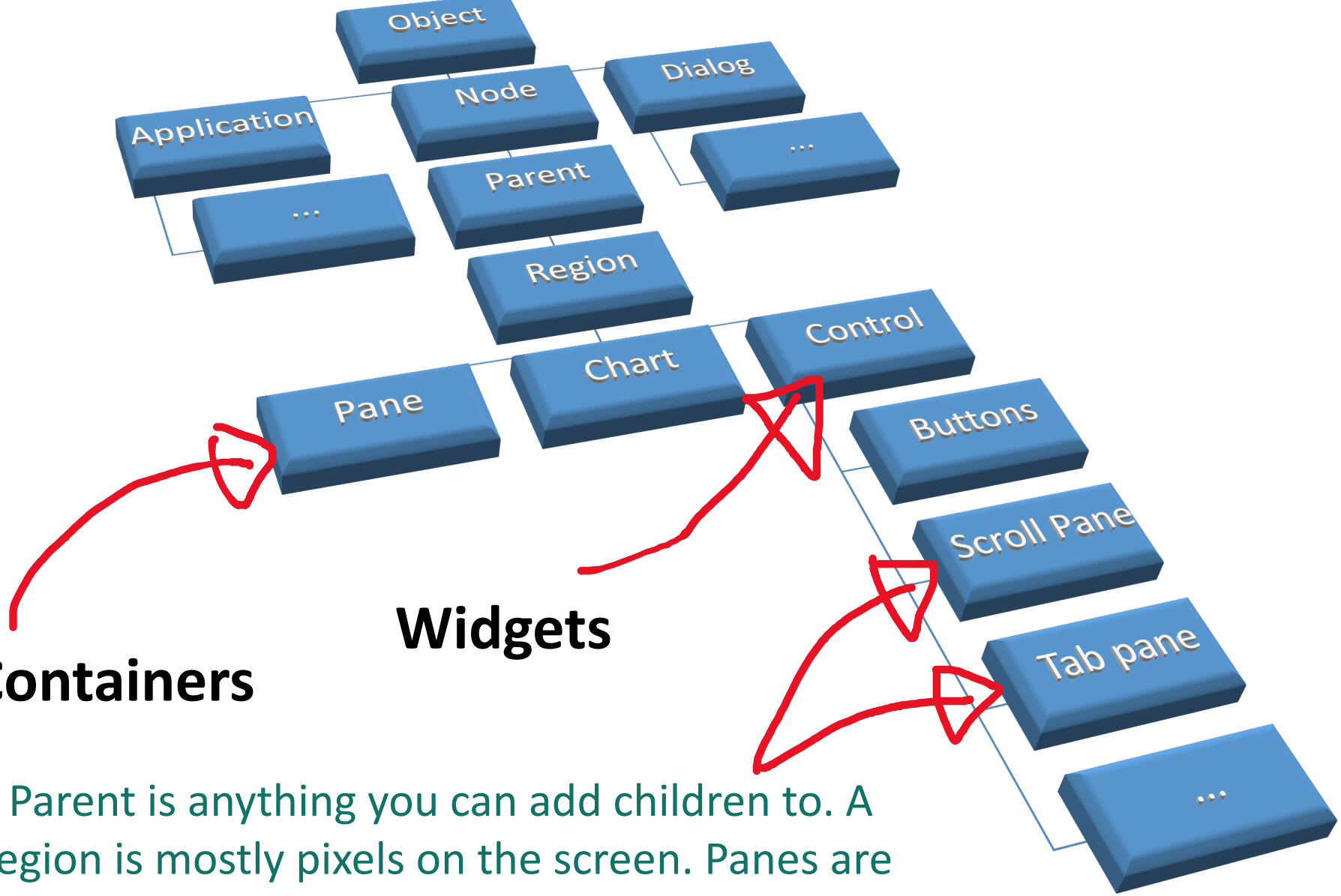
primary stage in start

display primary stage

create second stage

display second stage

main method omitted



Containers

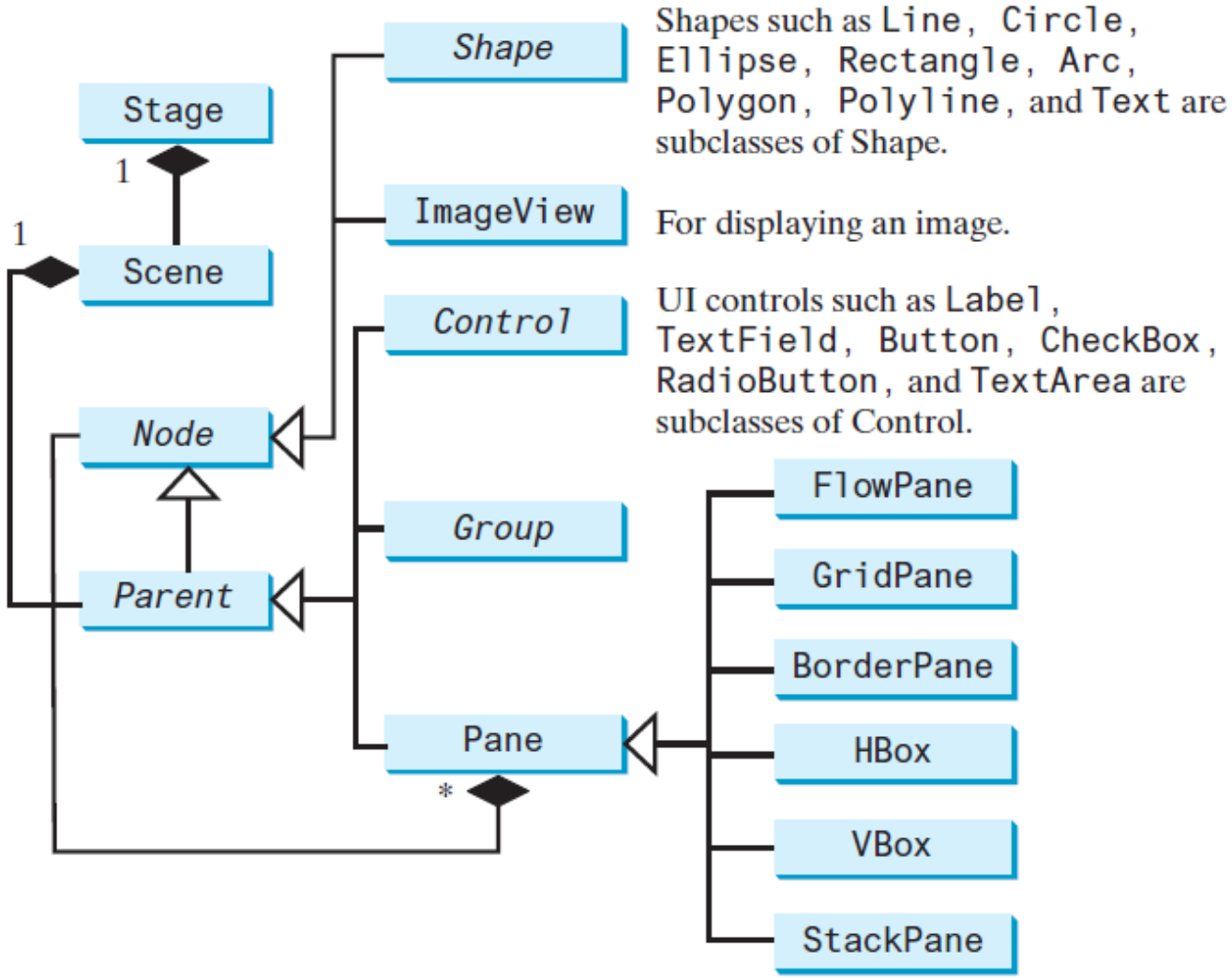
Widgets

A Parent is anything you can add children to. A Region is mostly pixels on the screen. Panes are containers, but beware that some widgets that you might regard as containers are in fact "Controls".

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).

Panes, Groups, UI controls, and shapes are subtypes of **Node**.

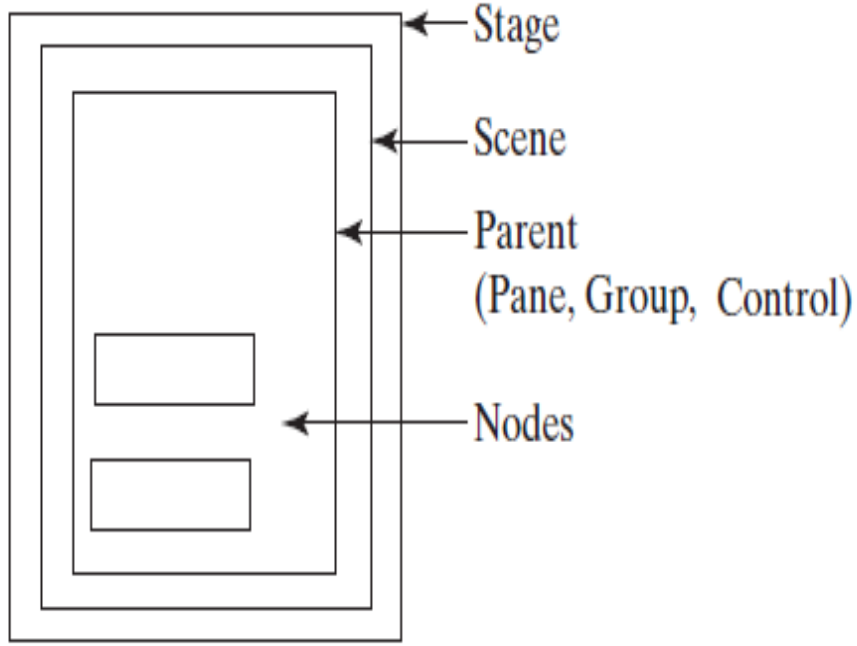


Shapes such as `Line`, `Circle`, `Ellipse`, `Rectangle`, `Arc`, `Polygon`, `Polyline`, and `Text` are subclasses of Shape.

For displaying an image.

UI controls such as `Label`, `TextField`, `Button`, `CheckBox`, `RadioButton`, and `TextArea` are subclasses of Control.

Nodes can be shapes, image views, UI controls, groups, and panes.



Panes and groups are used to hold nodes.

ButtonInPane.java

```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.stage.Stage;
5  import javafx.scene.layout.StackPane;
6
7  public class ButtonInPane extends Application {
8    @Override // Override the start method in the Application class
9    public void start(Stage primaryStage) {
10     // Create a scene and place a button in the scene
11     StackPane pane = new StackPane();
12     pane.getChildren().add(new Button("OK"));
13     Scene scene = new Scene(pane, 200, 50);
14     primaryStage.setTitle("Button in a pane"); // Set the stage title
15     primaryStage.setScene(scene); // Place the scene in the stage
16     primaryStage.show(); // Display the stage
17   }
18 }
```

create a pane

add a button

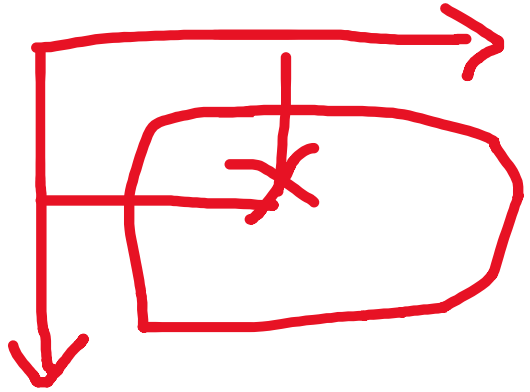
add pane to scene

display stage

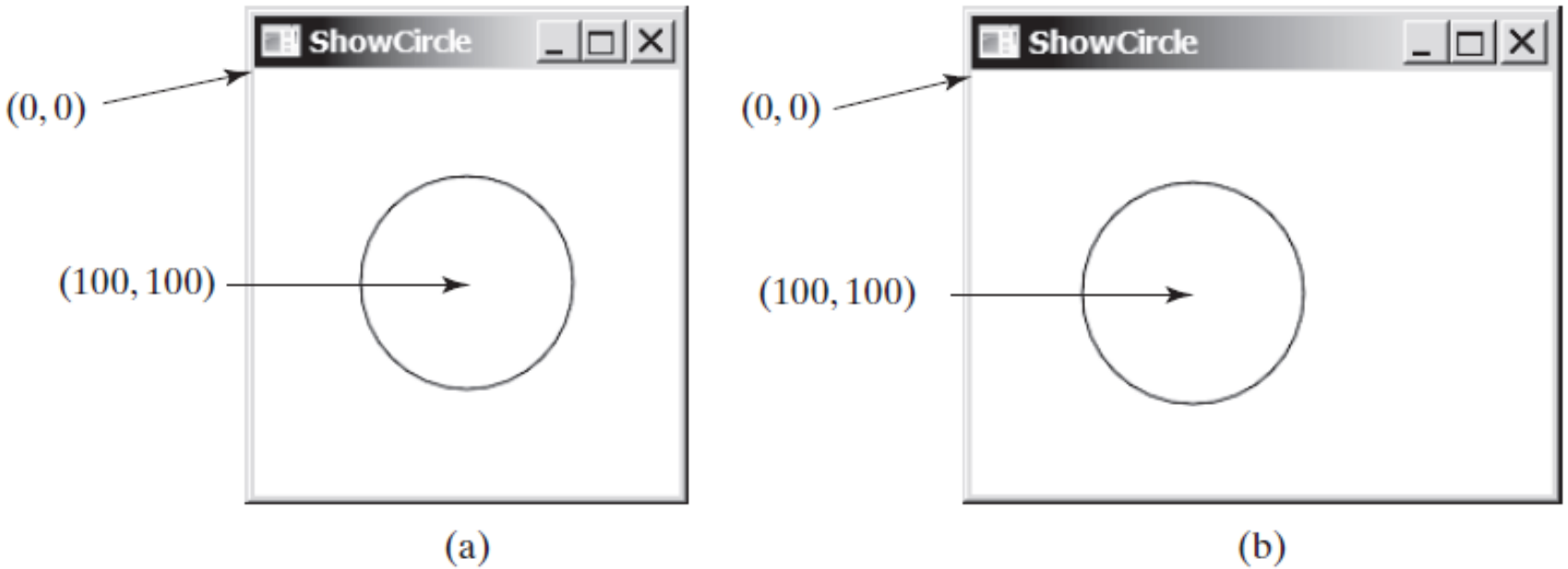
main method omitted

A button is placed in the center of the pane.

ShowCircle.java

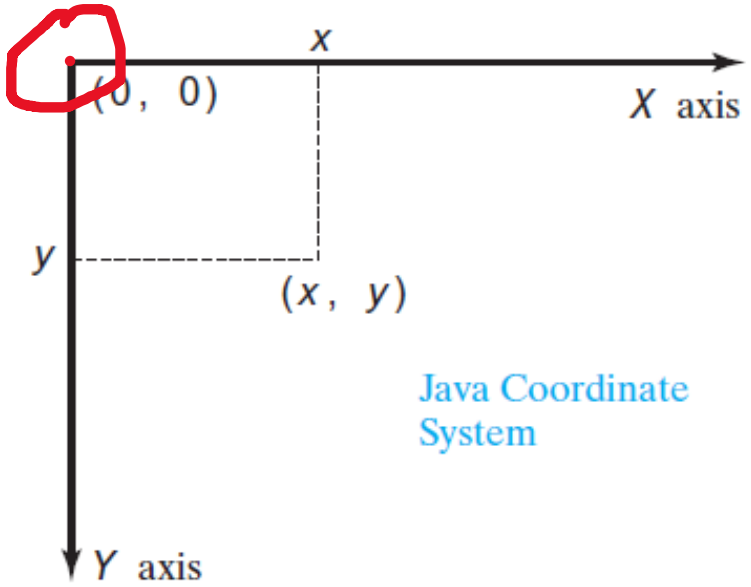


```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.layout.Pane;
4  import javafx.scene.paint.Color;
5  import javafx.scene.shape.Circle;
6  import javafx.stage.Stage;
7
8  public class ShowCircle extends Application {
9    @Override // Override the start method in the Application class
10   public void start(Stage primaryStage) {
11     // Create a circle and set its properties
12     Circle circle = new Circle();                          create a circle
13     circle.setCenterX(100);                                set circle properties
14     circle.setCenterY(100);
15     circle.setRadius(50);
16     circle.setStroke(Color.BLACK);
17     circle.setFill(Color.WHITE);
18
19     // Create a pane to hold the circle
20     Pane pane = new Pane();                                create a pane
21     pane.getChildren().add(circle);                        add circle to pane
22
23     // Create a scene and place it in the stage
24     Scene scene = new Scene(pane, 200, 200);               add pane to scene
25     primaryStage.setTitle("ShowCircle"); // Set the stage title
26     primaryStage.setScene(scene); // Place the scene in the stage
27     primaryStage.show(); // Display the stage              display stage
28   }
29 }                                                         main method omitted
```

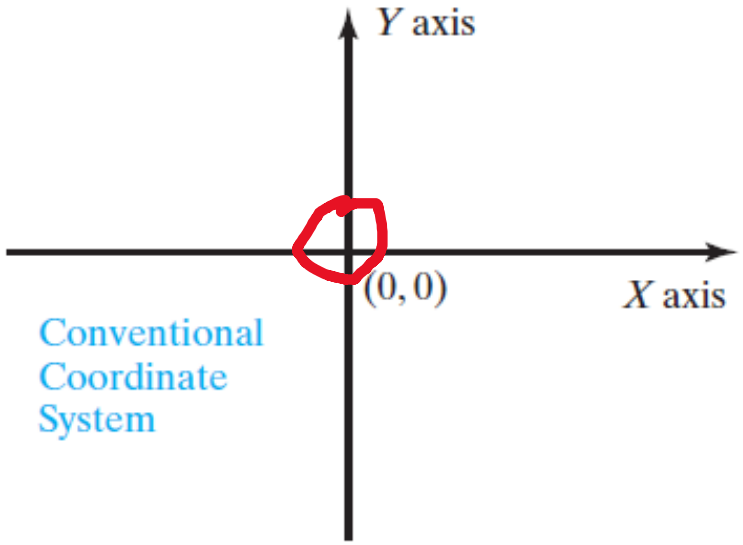


(a) A circle is displayed in the center of the scene.

(b) The circle is not centered after the window is resized.



(a)



(b)

The Java coordinate system is measured in pixels, with `(0, 0)` at its upper-left corner.

If you have a fixed–size window, things are easy. You can say "I want this widget to appear at these coordinates relative to the upper–left corner of the window".

Unfortunately the easy case isn't the most common…

Some containers are fixed

Next: 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.