JavaFX Basics

Lecture 7

Graphical User Interface

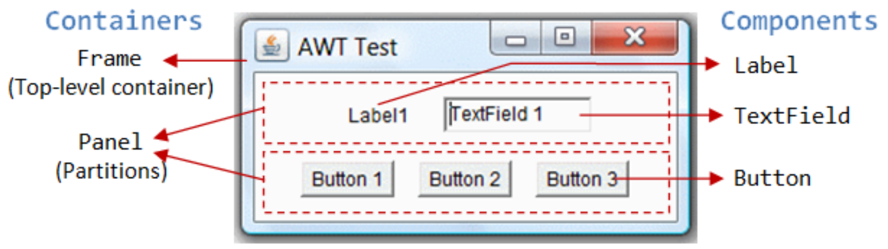
- So far all our interaction with the user has been via terminal (**`System.in`**), command-line arguments (**`args`**), and files

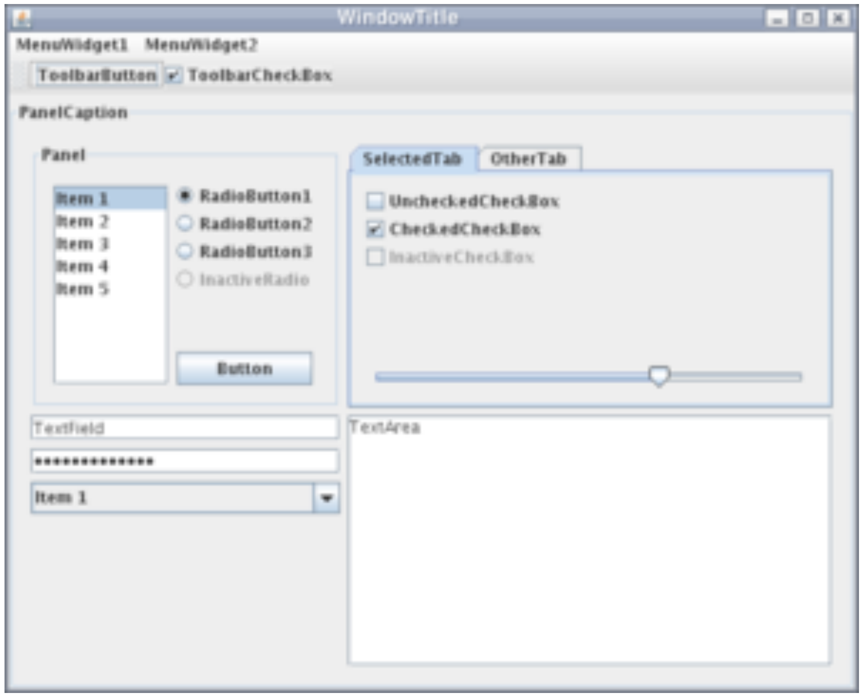
- We now look at the basics of GUIs (pronounced “gooey”) – graphical user interfaces
  – Window(s), menus, buttons, etc.

JavaFX

- JavaFX is a relatively new framework for developing Java GUI programs
- The JavaFX API is an excellent example of OOP
- JavaFX replaces older frameworks
  - Abstract Window Toolkit (AWT): prone to platform-specific bugs, original GUI framework
  - Swing: replaced AWT, now superseded by JavaFX

Older Java GUIs





JavaFX Features

- Runs on a desktop or from a Web browser
- Provides a multi-touch support for touch-enabled devices (tablets and smart phones)
- Has built-in 2D/3D animation support, video and audio playback

Your First JavaFX Project

- Create a new project in Eclipse
  – Name: MyJavaFX
- Create a new class
  – MyJavaFX
    - Extend the “Application” class
  – Include a main method

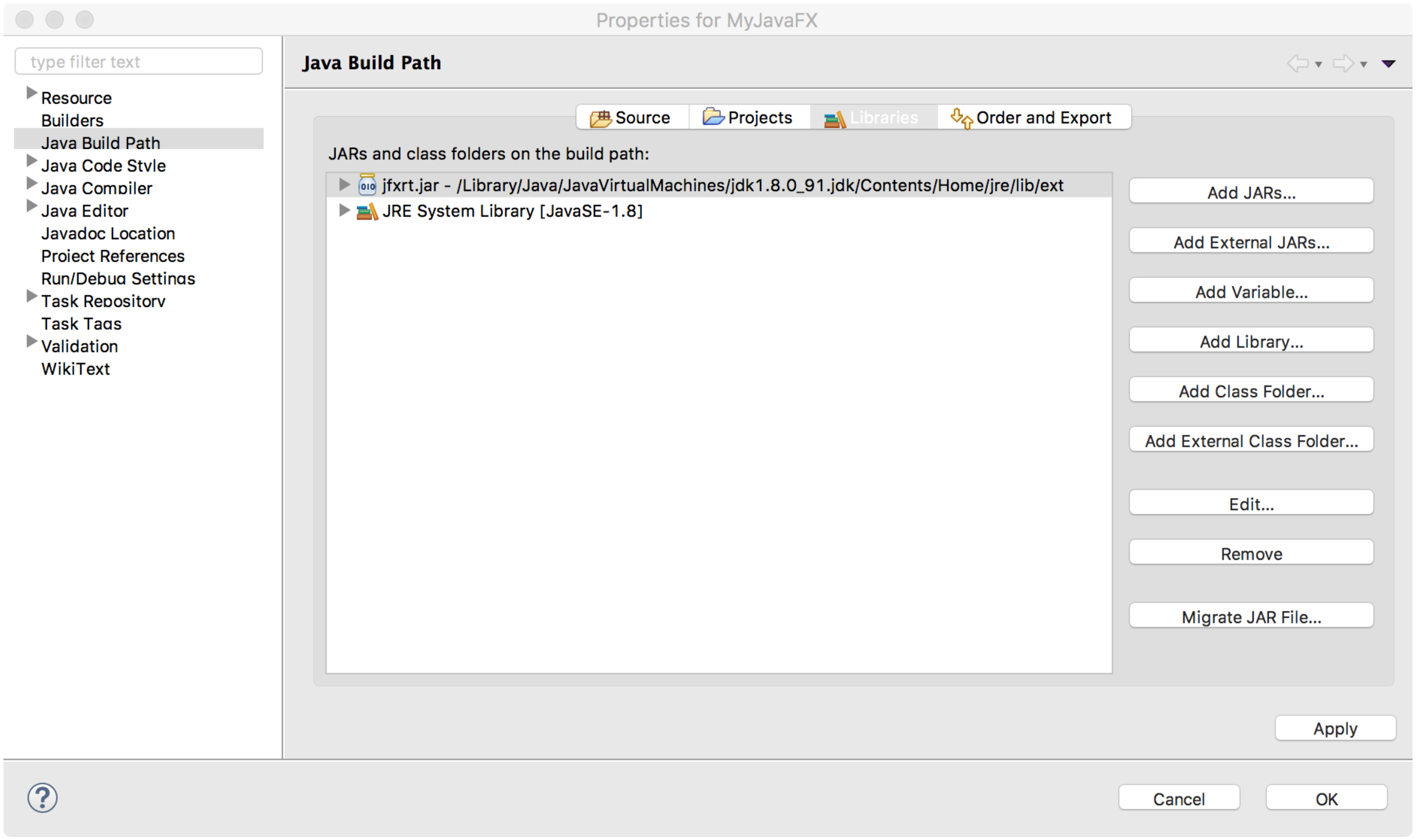
```
public class MyJavaFX extends Application {
    public static void main(String[] args) {
    }
}
```

Including JavaFX

- All JavaFX applications need the JavaFX runtime library (**`jfxrt.jar`**) added to the class path (location java looks for libraries)

- In Eclipse…
  1. Right-click project, Properties
  2. Java Build Path -> Libraries
  3. Add External JARs
     - Mac: **`/Library/Java/JavaVirtualMachines/jdk1.8.X_X.jdk/Contents/Home/jre/lib/ext`**
     - Windows: **`C:\Program Files\Java\jdk1.8.X_X\jre\ext`**
  4. Order and Export -> move **`jfxrt.jar`** to the top, OK

Screenshots (1)



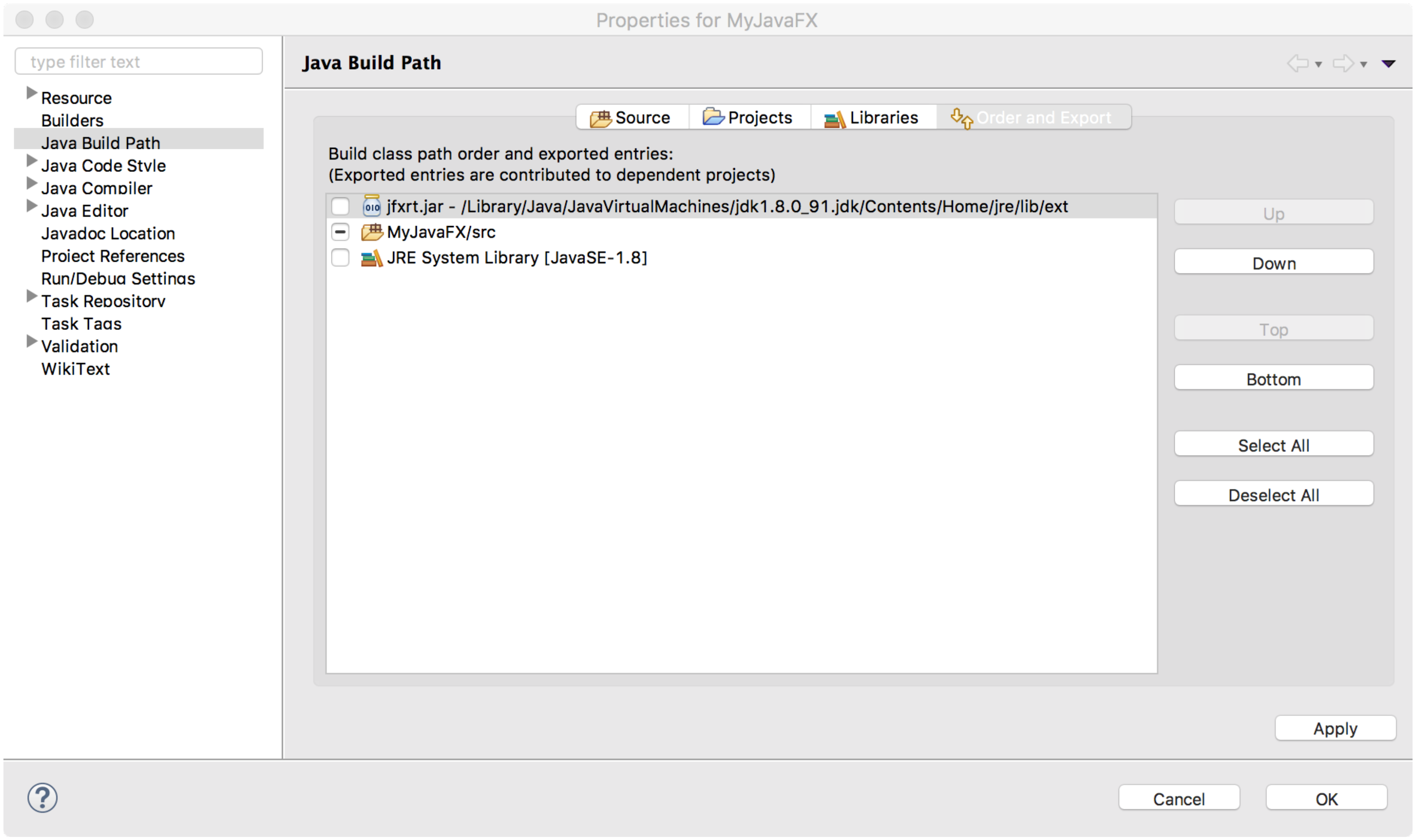
Properties for MyJavaFX

Java Build Path

JARs and class folders on the build path:

```
▶ jfxrt.jar - /Library/Java/JavaVirtualMachines/jdk1.8.0_91.jdk/Contents/Home/jre/lib/ext
▶ JRE System Library [JavaSE-1.8]
```

Screenshots (2)



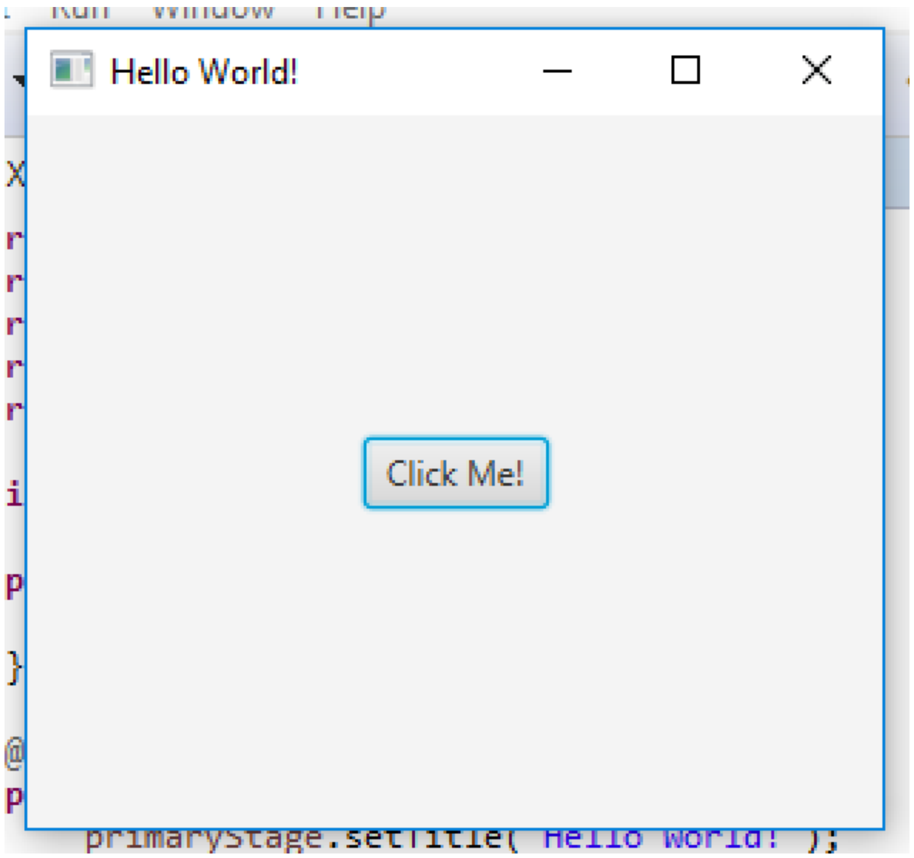
Build class path order and exported entries:
(Exported entries are contributed to dependent projects)

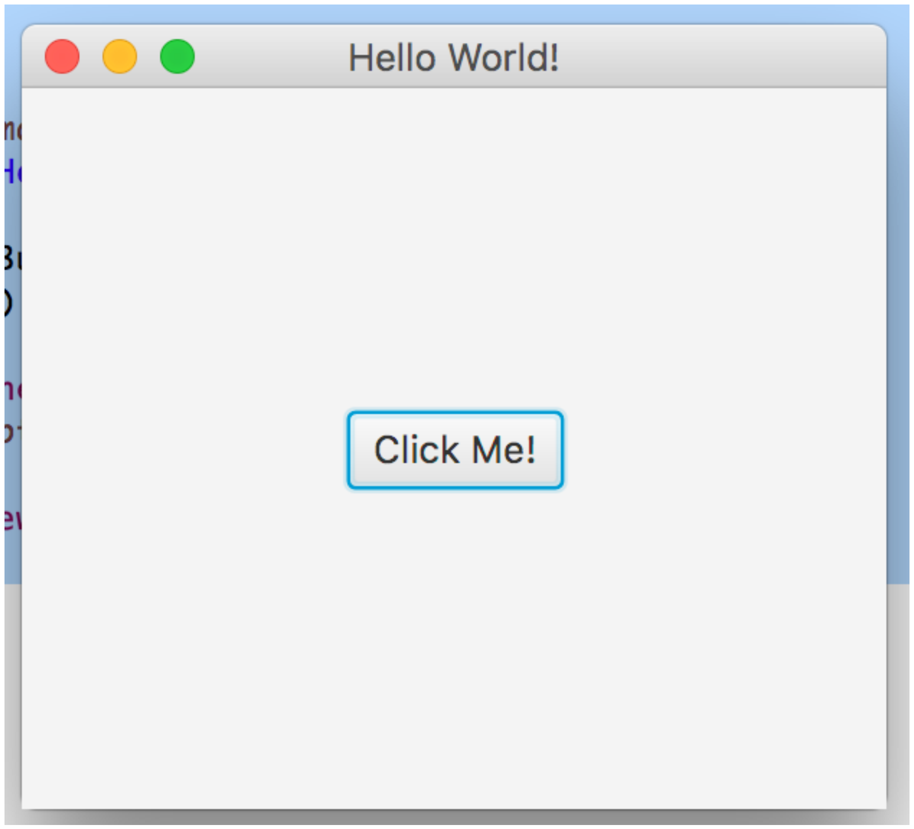
- [ ] jfxrt.jar - /Library/Java/JavaVirtualMachines/jdk1.8.0_91.jdk/Contents/Home/jre/lib/ext
- [–] MyJavaFX/src
- [ ] JRE System Library [JavaSE-1.8]

My First JavaFX Application

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

public class MyJavaFX extends Application {
	public static void main(String[] args) {
		launch(args);
	}

	@Override
	public void start(Stage primaryStage) {
		primaryStage.setTitle("Hello World!");

		final Button btn = new Button();
		btn.setText("Click Me!");

		final StackPane root = new StackPane();
		root.getChildren().add(btn);

		primaryStage.setScene(new Scene(root, 300, 250));
		primaryStage.show();
	}
}
```

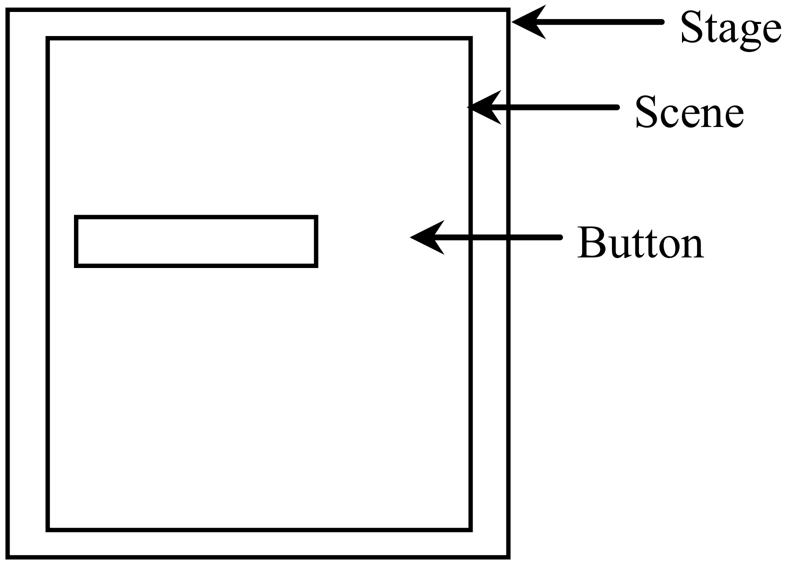
Platform-Independent GUI





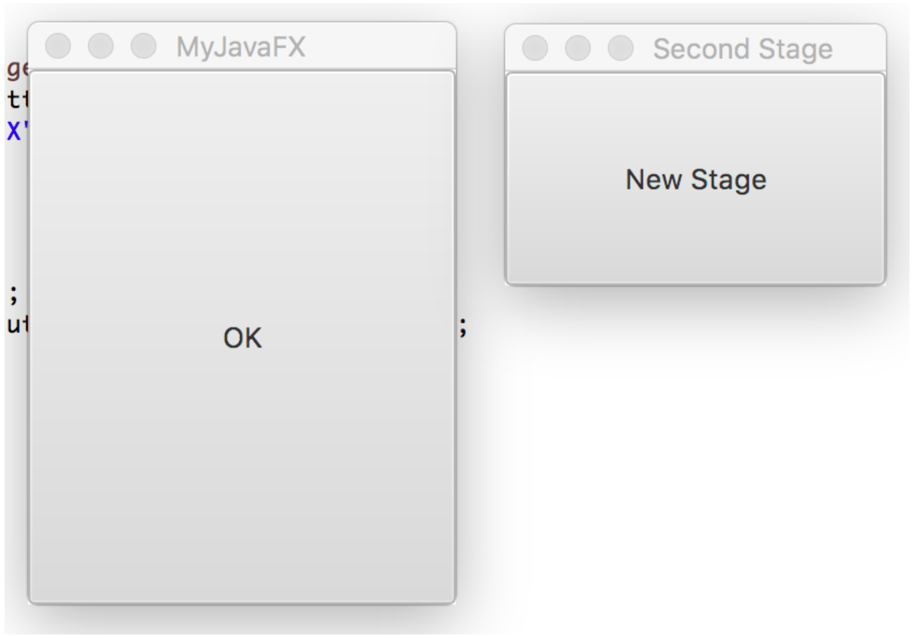
Basic Structure of JavaFX

1. Extend Application
2. `launch(args)` in `main`
3. Override `start(Stage)`
4. Populate
   - Stage (Window): primary=default, can have multiple
   - Scene: hierarchical graph of nodes

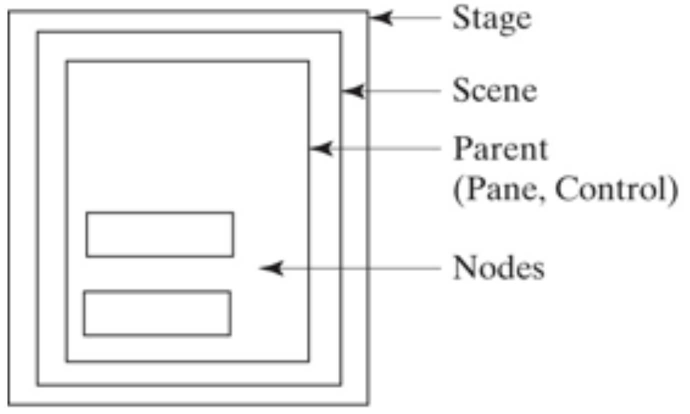


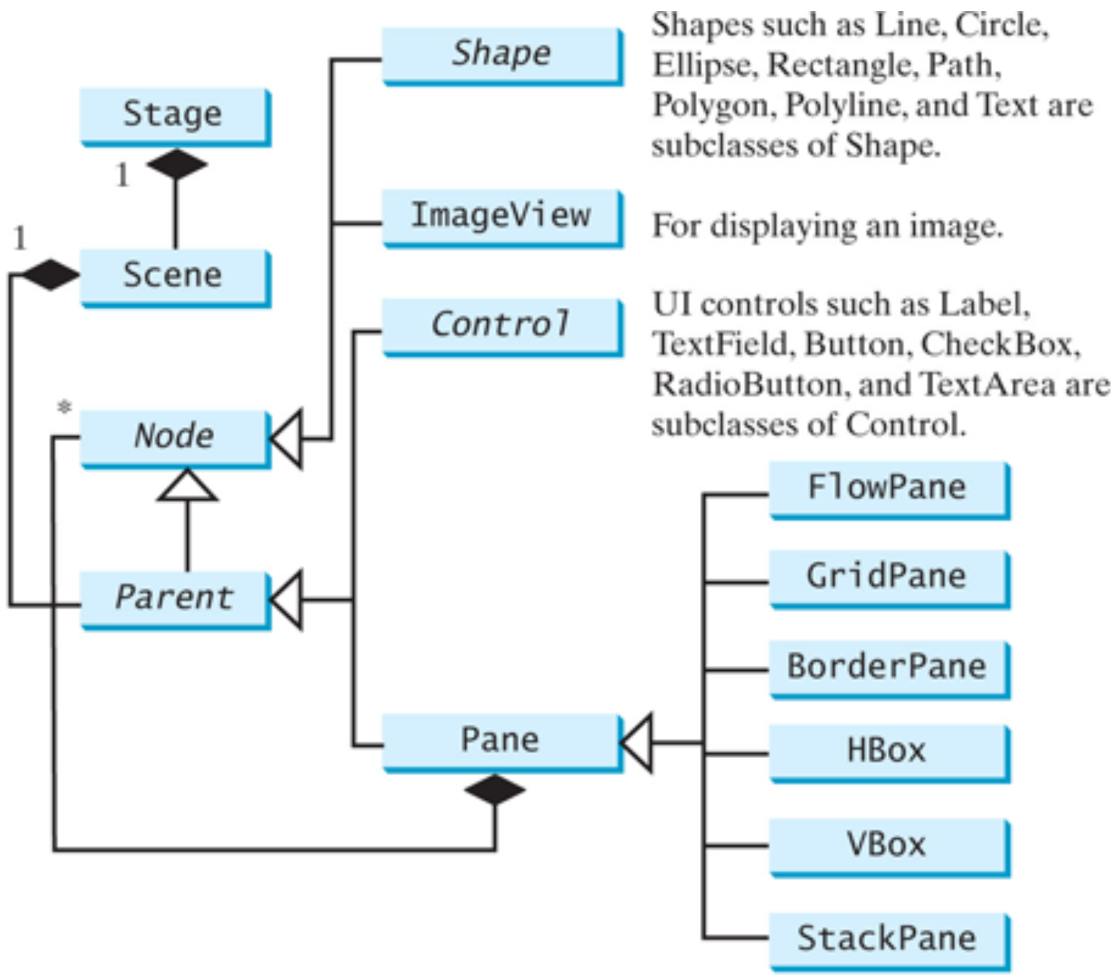
Multiple Windows

```
public void start(Stage primaryStage) {
     Scene scene = new Scene(
          new Button("OK"), 200, 250);
     primaryStage.setTitle("MyJavaFX");
     primaryStage.setScene(scene);
     primaryStage.show();

     Stage stage = new Stage();
     stage.setTitle("Second Stage");
     stage.setScene(
          new Scene(
               new Button("New Stage"),
               100, 100));
     stage.show();
}
```



UML Relationships



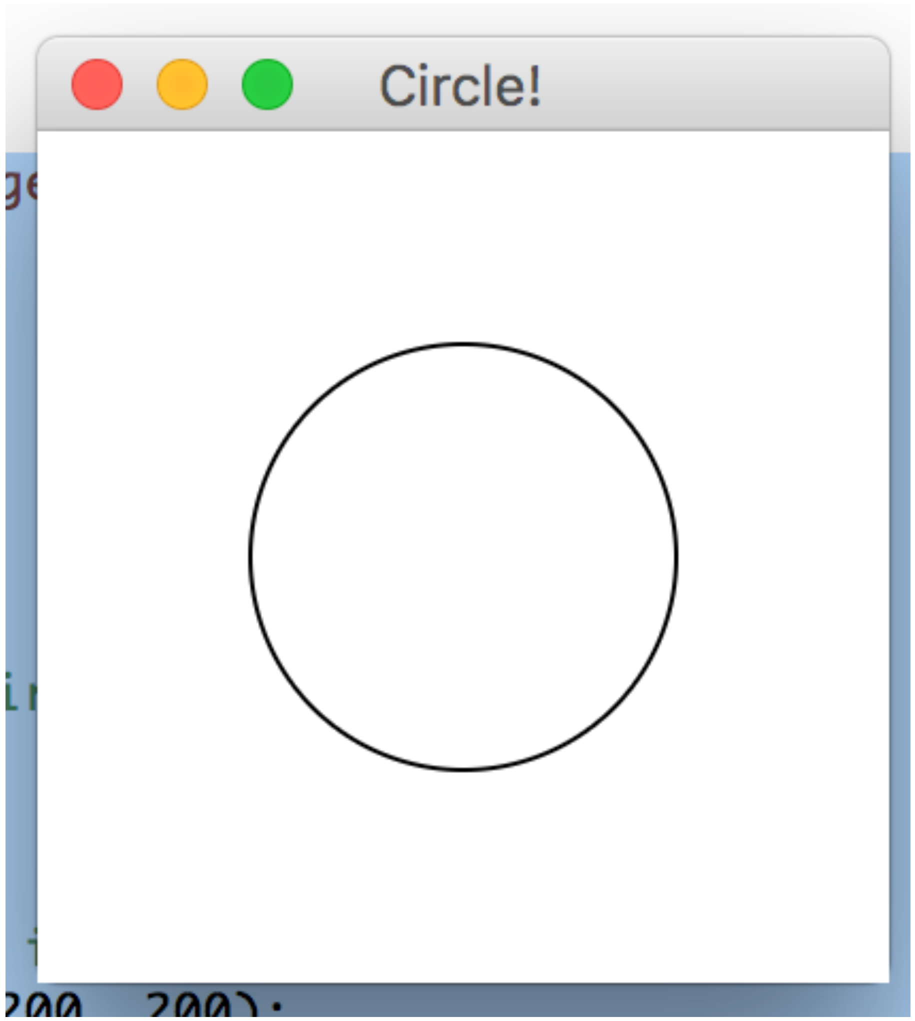

Stage
1
1
Scene
*
Node
Parent
Shape
Shapes such as Line, Circle, Ellipse, Rectangle, Path, Polygon, Polyline, and Text are subclasses of Shape.
ImageView
For displaying an image.
Control
UI controls such as Label, TextField, Button, CheckBox, RadioButton, and TextArea are subclasses of Control.
Pane
FlowPane
GridPane
BorderPane
HBox
VBox
StackPane

Revisit

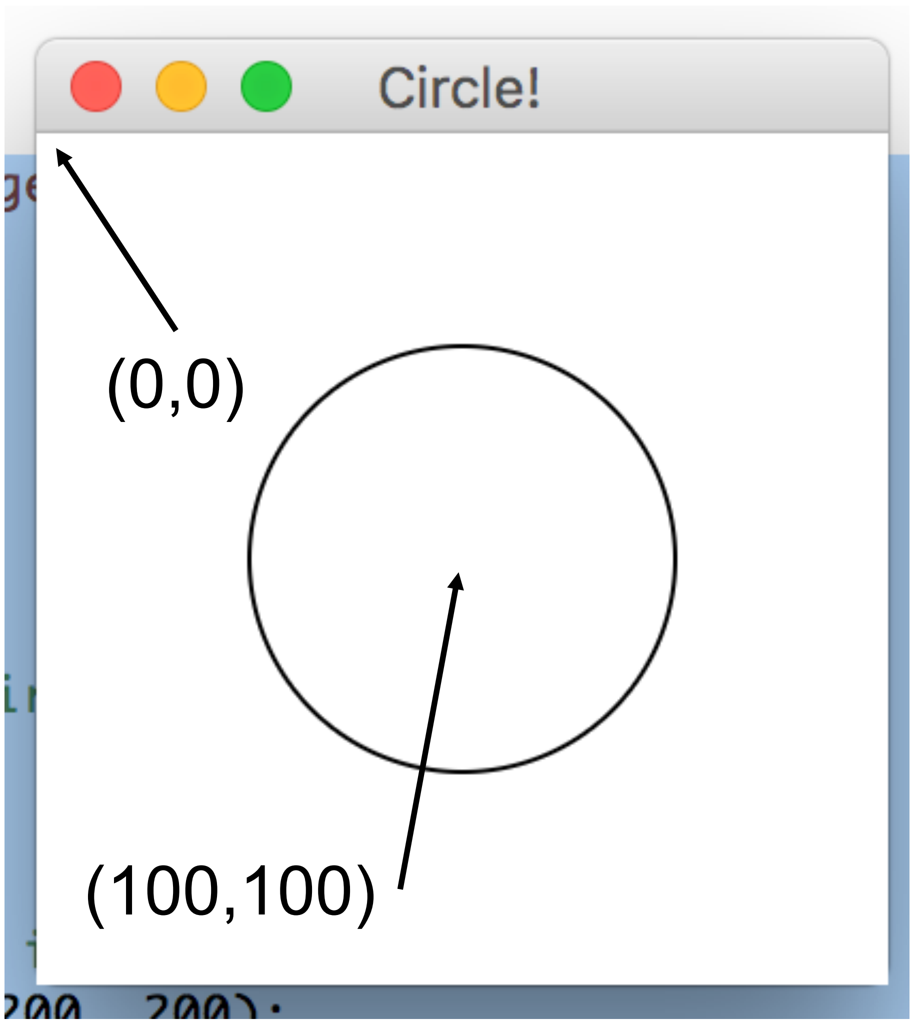
```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class MyJavaFX extends Application {
	public static void main(String[] args) {
		launch(args);
	}

	@Override
	public void start(Stage primaryStage) {
		primaryStage.setTitle("Hello World!");

		final Button btn = new Button();
		btn.setText("Click Me!");

		final StackPane root = new StackPane(); // Forms the root of the nodes, organize vertically
		root.getChildren().add(btn); // Add the button to the root

		primaryStage.setScene(new Scene(root, 300, 250)); // Place the pane in the scene
		primaryStage.show();
	}
}
```

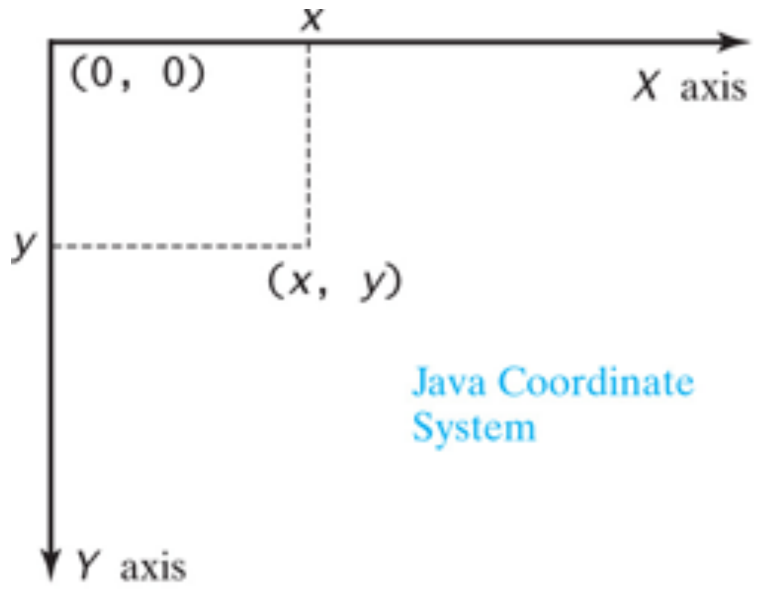
Another Example

```
public void start(Stage primaryStage) {
     final Circle c = new Circle();
     c.setCenterX(100);
     c.setCenterY(100);
     c.setRadius(50);
     c.setStroke(Color.BLACK);
     c.setFill(Color.WHITE);

     Pane pane = new Pane();
     pane.getChildren().add(c);

     Scene scene = new Scene(
                    pane, 200, 200);
     primaryStage.setTitle("Circle!");
     primaryStage.setScene(scene);
     primaryStage.show();
}
```

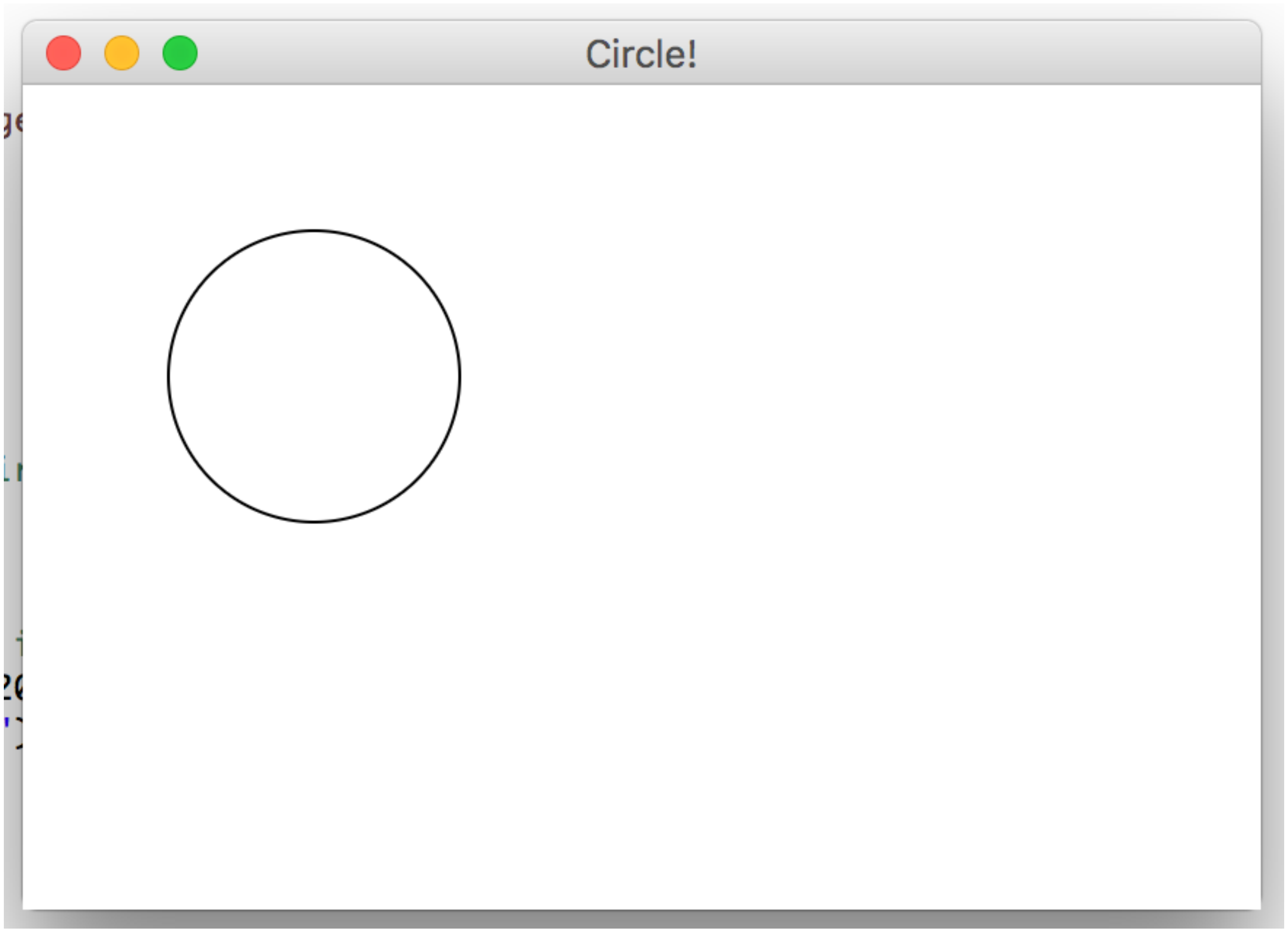


Notes


Circle!
(0,0)
(100,100)


x
(0, 0)
X axis
y
(x, y)
Java Coordinate System
Y axis

Resizing the Window :(



Solution 1: No Resizing

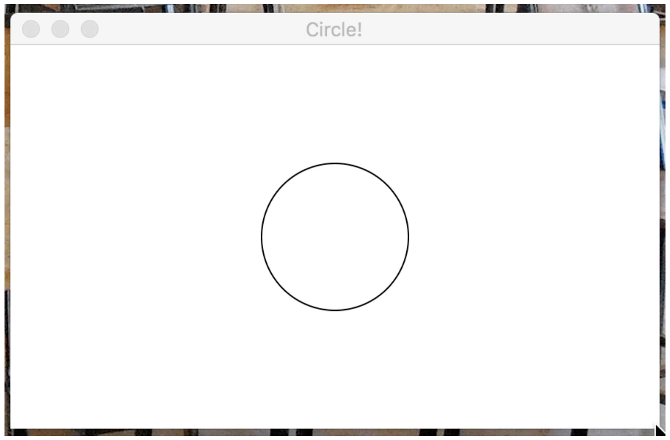
```
primaryStage.setResizable(false);
```

Solution 2: Property Binding

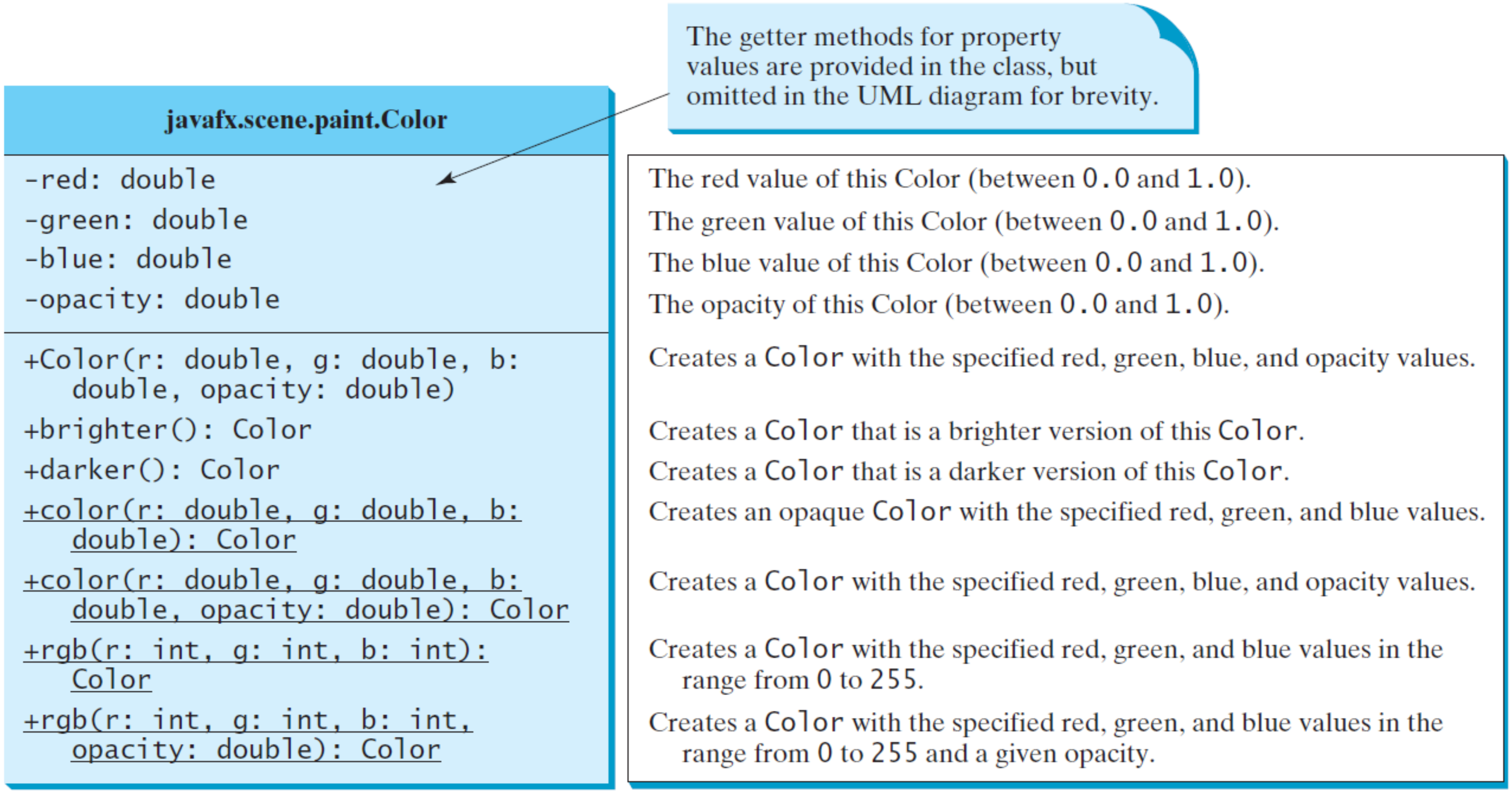
- JavaFX introduces a new concept called **property binding** that enables a target object to be bound to a source object
- If the value in the source object changes, the target object is also changed automatically
- The target object is called a *binding object* or a *binding property* and the source object is called a *bindable object* or *observable object*

Example

```
public void start(Stage primaryStage) {
    Pane pane = new Pane();

    final Circle c = new Circle();
    c.setCenterX(100);
    c.setCenterY(100);
    c.setRadius(50);
    c.setStroke(Color.BLACK);
    c.setFill(Color.WHITE);

    c.centerXProperty().bind(pane.widthProperty().divide(2));
    c.centerYProperty().bind(pane.heightProperty().divide(2));

    pane.getChildren().add(c);

    Scene scene = new Scene(pane, 200, 200);
    primaryStage.setTitle("Circle!");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```



The `Color` Class

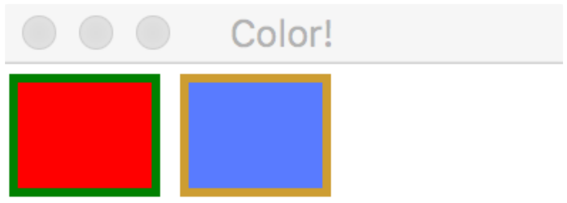

The getter methods for property values are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.paint.Color

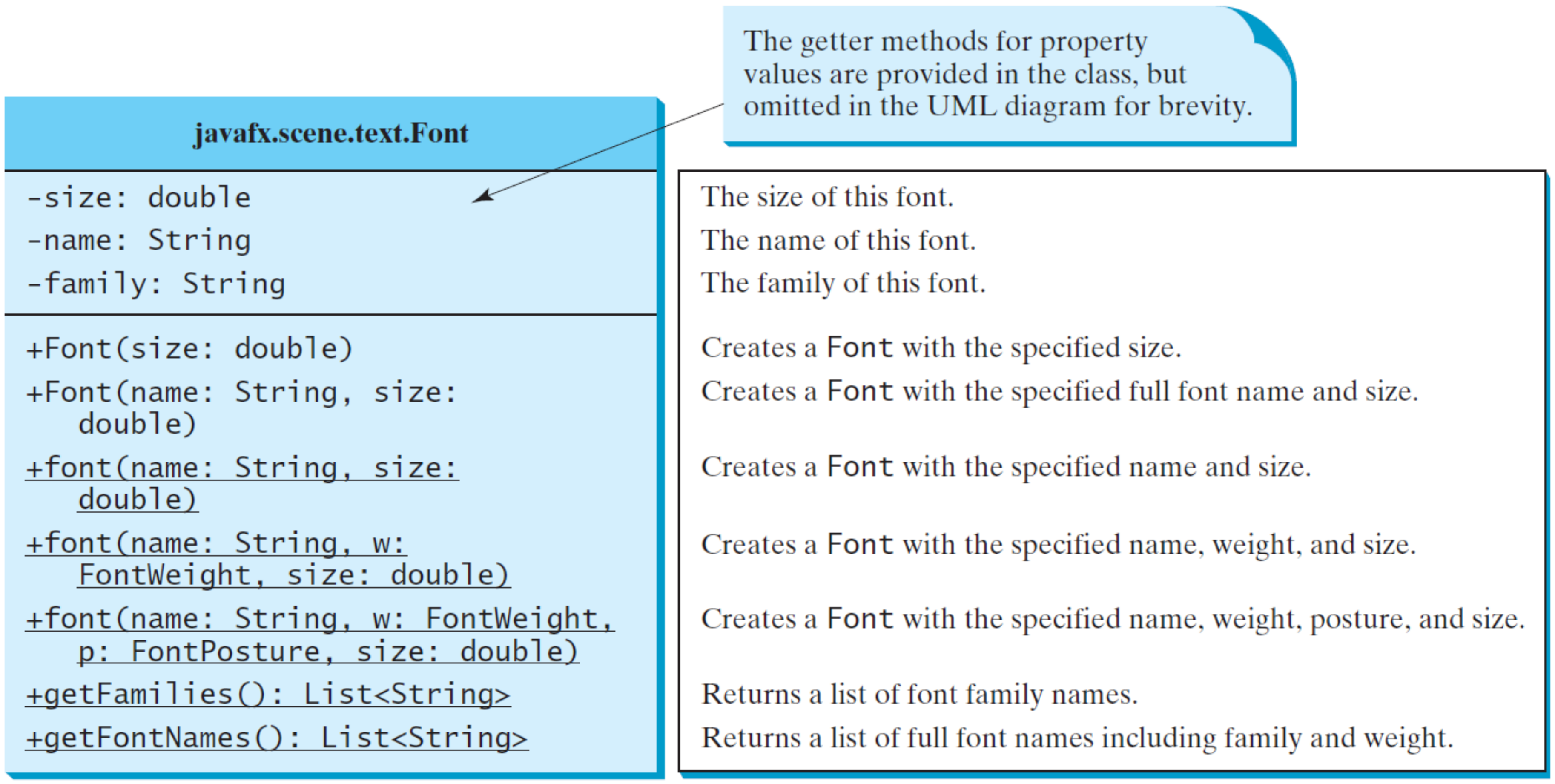
javafx.scene.paint.Color	
`-red: double`	The red value of this Color (between `0.0` and `1.0`).
`-green: double`	The green value of this Color (between `0.0` and `1.0`).
`-blue: double`	The blue value of this Color (between `0.0` and `1.0`).
`-opacity: double`	The opacity of this Color (between `0.0` and `1.0`).
`+Color(r: double, g: double, b: double, opacity: double)`	Creates a `Color` with the specified red, green, blue, and opacity values.
`+brighter(): Color`	Creates a `Color` that is a brighter version of this `Color`.
`+darker(): Color`	Creates a `Color` that is a darker version of this `Color`.
<u>`+color(r: double, g: double, b: double): Color`</u>	Creates an opaque `Color` with the specified red, green, and blue values.
<u>`+color(r: double, g: double, b: double, opacity: double): Color`</u>	Creates a `Color` with the specified red, green, blue, and opacity values.
<u>`+rgb(r: int, g: int, b: int): Color`</u>	Creates a `Color` with the specified red, green, and blue values in the range from `0` to `255`.
<u>`+rgb(r: int, g: int, b: int, opacity: double): Color`</u>	Creates a `Color` with the specified red, green, and blue values in the range from `0` to `255` and a given opacity.

Example

```
Rectangle rec1 =
     new Rectangle(5, 5, 50, 40);
rec1.setFill(Color.RED);
rec1.setStroke(Color.GREEN);
rec1.setStrokeWidth(3);

Rectangle rec2 =
     new Rectangle(65, 5, 50, 40);
rec2.setFill(Color.rgb(91, 127, 255));
rec2.setStroke(
     Color.hsb(40, 0.7, 0.8));
rec2.setStrokeWidth(3);
```



The Font Class



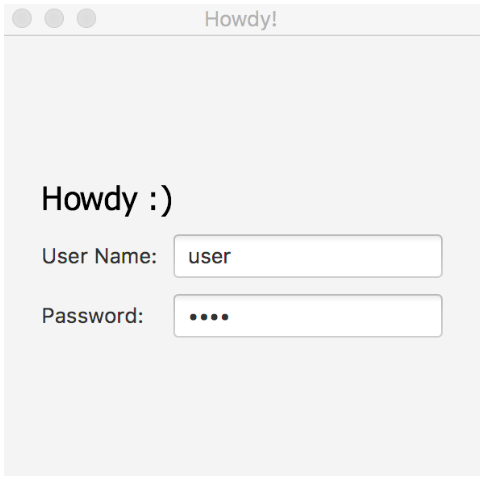
The getter methods for property values are provided in the class, but omitted in the UML diagram for brevity.

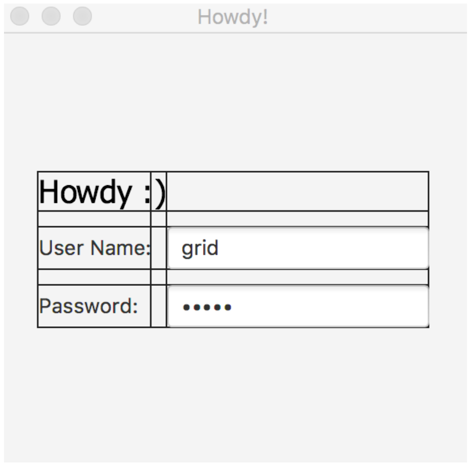
javafx.scene.text.Font

-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.
+getFamilies(): List<String>	Returns a list of font family names.
+getFontNames(): List<String>	Returns a list of full font names including family and weight.

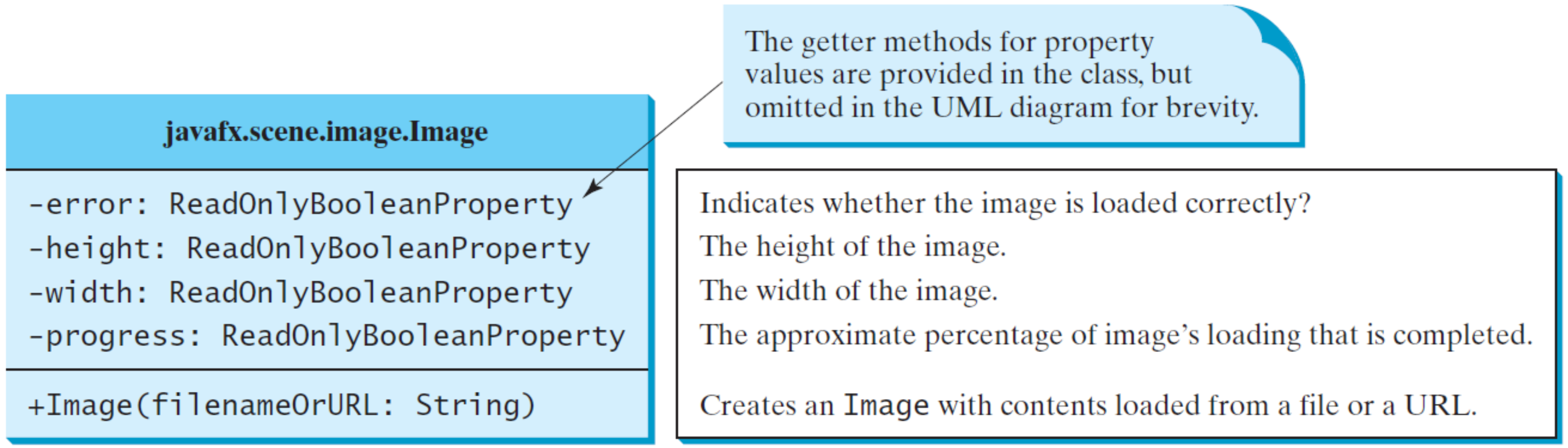
Example

```
primaryStage.setTitle("Howdy!");

GridPane grid = new GridPane();
// grid.setGridLinesVisible(true);
grid.setAlignment(Pos.CENTER);
grid.setHgap(10);
grid.setVgap(10);
grid.setPadding(new Insets(25, 25, 25, 25));

Text scenetitle = new Text("Howdy :)");
scenetitle.setFont(
      Font.font("Tahoma", FontWeight.NORMAL, 20));
grid.add(scenetitle, 0, 0, 2, 1);

Label userName = new Label("User Name:");
grid.add(userName, 0, 1);
TextField userTextField = new TextField();
grid.add(userTextField, 1, 1);

Label pw = new Label("Password:");
grid.add(pw, 0, 2);
PasswordField pwBox = new PasswordField();
grid.add(pwBox, 1, 2);

Scene scene = new Scene(grid, 300, 275);
primaryStage.setScene(scene);
primaryStage.show();
```





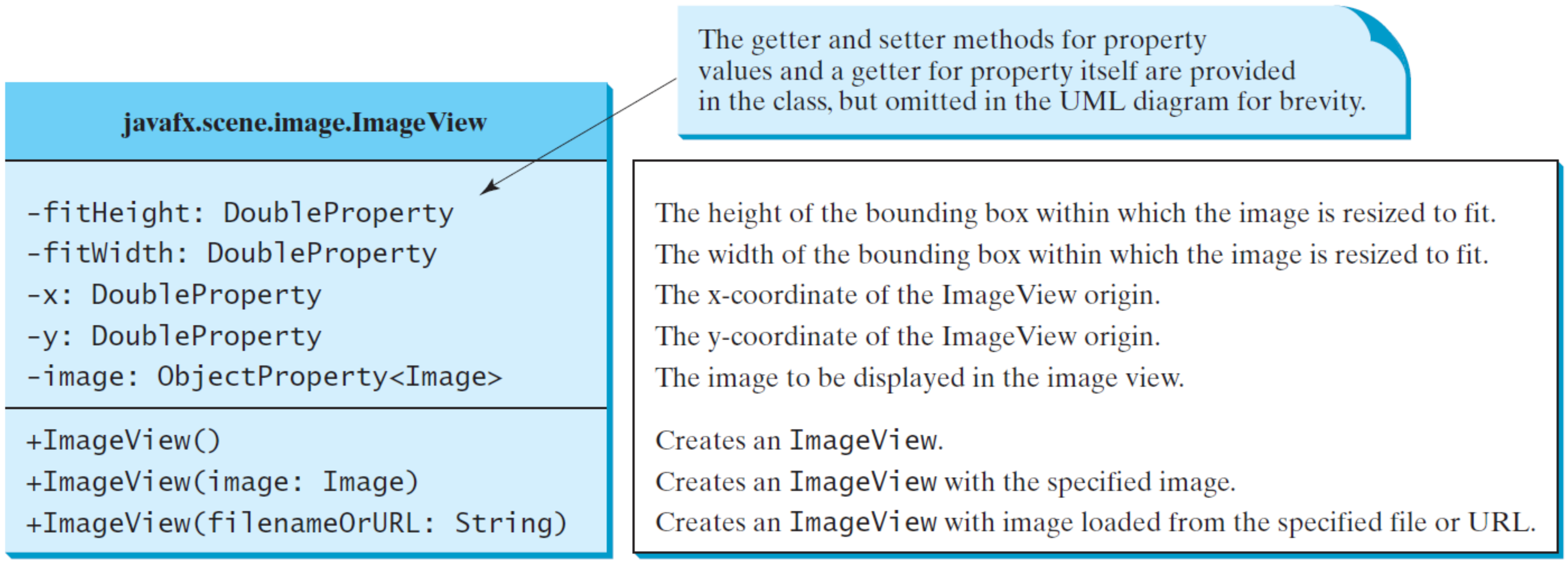
The `Image` Class



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: ReadOnlyBooleanProperty`	The height of the image.
`-width: ReadOnlyBooleanProperty`	The width of the image.
`-progress: ReadOnlyBooleanProperty`	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 `ImageView` Class



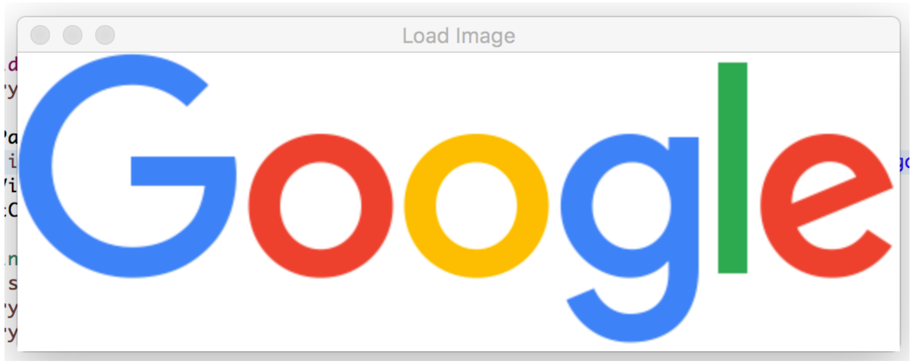
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.image.ImageView

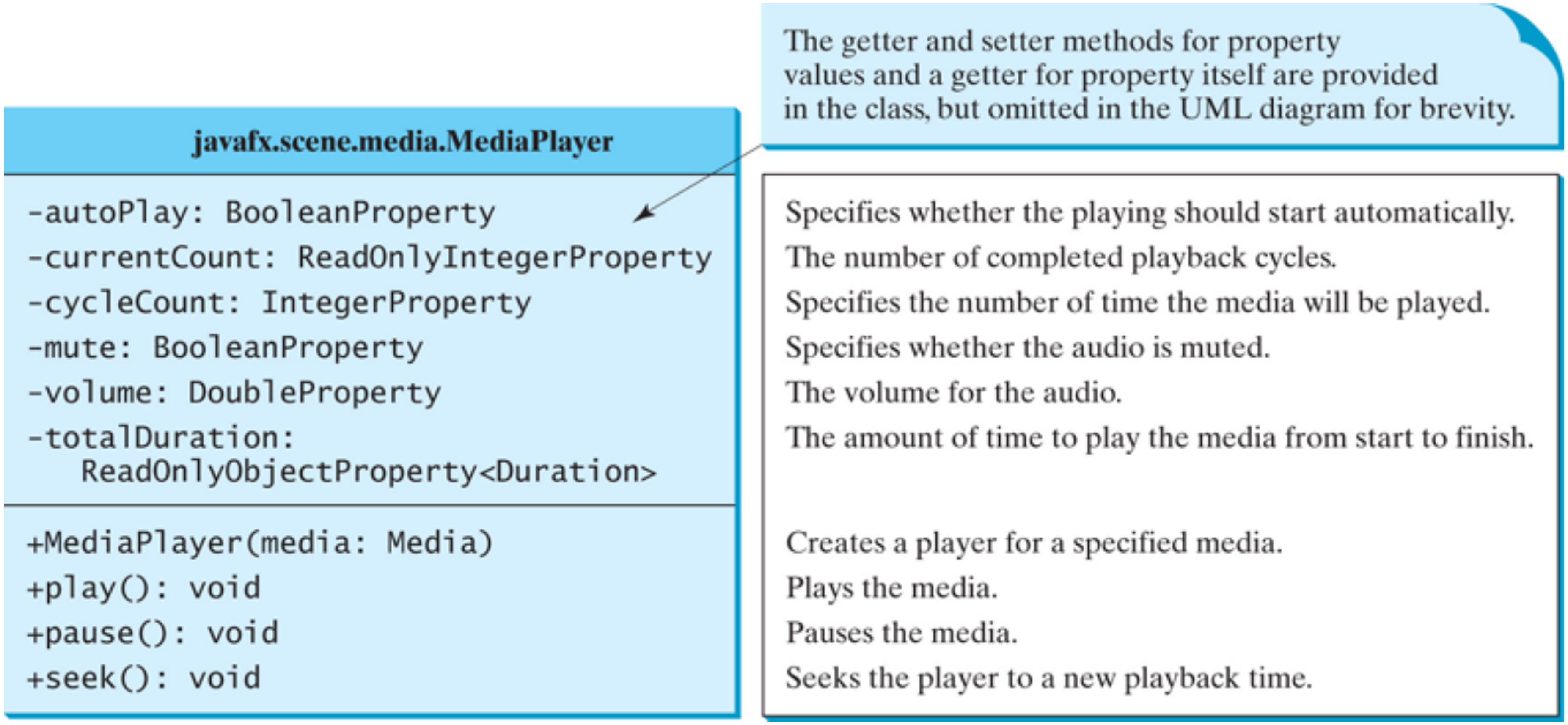
`-fitHeight: DoubleProperty`	The height of the bounding box within which the image is resized to fit.
`-fitWidth: DoubleProperty`	The width of the bounding box within which the image is resized to fit.
`-x: DoubleProperty`	The x-coordinate of the ImageView origin.
`-y: DoubleProperty`	The y-coordinate 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.

Example

```
primaryStage.setTitle("Load Image");
StackPane sp = new StackPane();
Image img = new Image(
     "https://www.google.com/images/bra
     nding/googlelogo/2x/googlelogo_col
     or_272x92dp.png");
ImageView imgView = new ImageView(img);
sp.getChildren().add(imgView);
Scene scene = new Scene(sp);
primaryStage.setScene(scene);
primaryStage.show();
```



The `MediaPlayer` Class



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.media.MediaPlayer	
`-autoPlay: BooleanProperty`	Specifies whether the playing should start automatically.
`-currentCount: ReadOnlyIntegerProperty`	The number of completed playback cycles.
`-cycleCount: IntegerProperty`	Specifies the number of time the media will be played.
`-mute: BooleanProperty`	Specifies whether the audio is muted.
`-volume: DoubleProperty`	The volume for the audio.
`-totalDuration: ReadOnlyObjectProperty<Duration>`	The amount of time to play the media from start to finish.
`+MediaPlayer(media: Media)`	Creates a player for a specified media.
`+play(): void`	Plays the media.
`+pause(): void`	Pauses the media.
`+seek(): void`	Seeks the player to a new playback time.

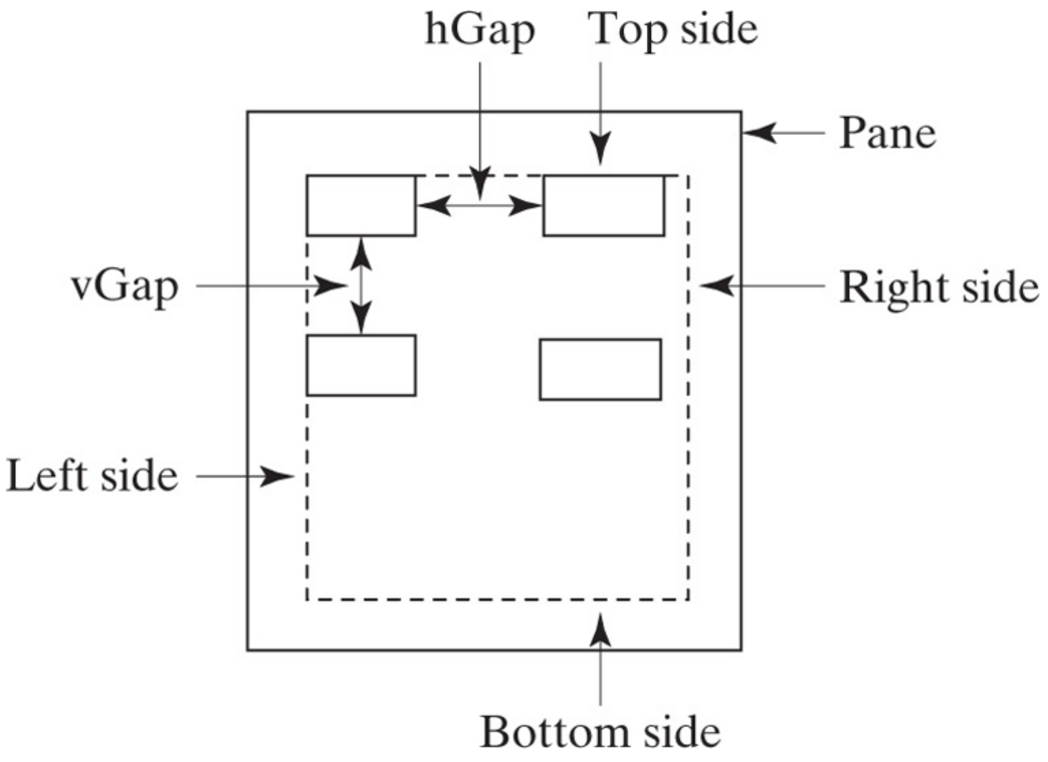
Example

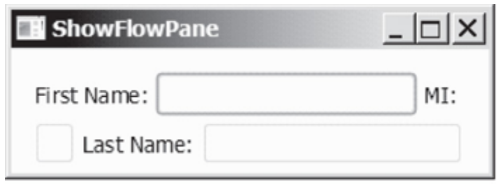
```
MediaPlayer player;

@Override
public void start(Stage primaryStage) throws Exception {
      final Button b = new Button("pause");
      b.setOnAction(new EventHandler<ActionEvent>() { // more on this later!
            @Override
            public void handle(ActionEvent event) {
                  if (player.getStatus()==Status.PAUSED) {
                        player.play();
                        b.setText("pause");
                  } else {
                        player.pause();
                        b.setText("play!");
                  }
            }
      });

      final StackPane sp = new StackPane();
      sp.getChildren().add(b);

      player = new MediaPlayer(new Media(getClass().getResource("flynn.mp3").toString()));
      player.play();

      primaryStage.setScene(new Scene(sp));
      primaryStage.show();
}
```

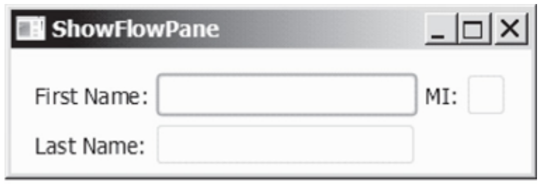
Layout Panes

- JavaFX provides many types of panes for organizing nodes in a container
  - **`Pane`**: base class
  - **`FlowPane`**: row-by-row vertically, or column-by-column horizontally
  - **`BorderPane`**: top-right-left-bottom-center
  - **`StackPane`**: stack vertically in the center
  - **`GridPane`**: 2D grid
  - **`HBox`**: single row
  - **`VBox`**: single column

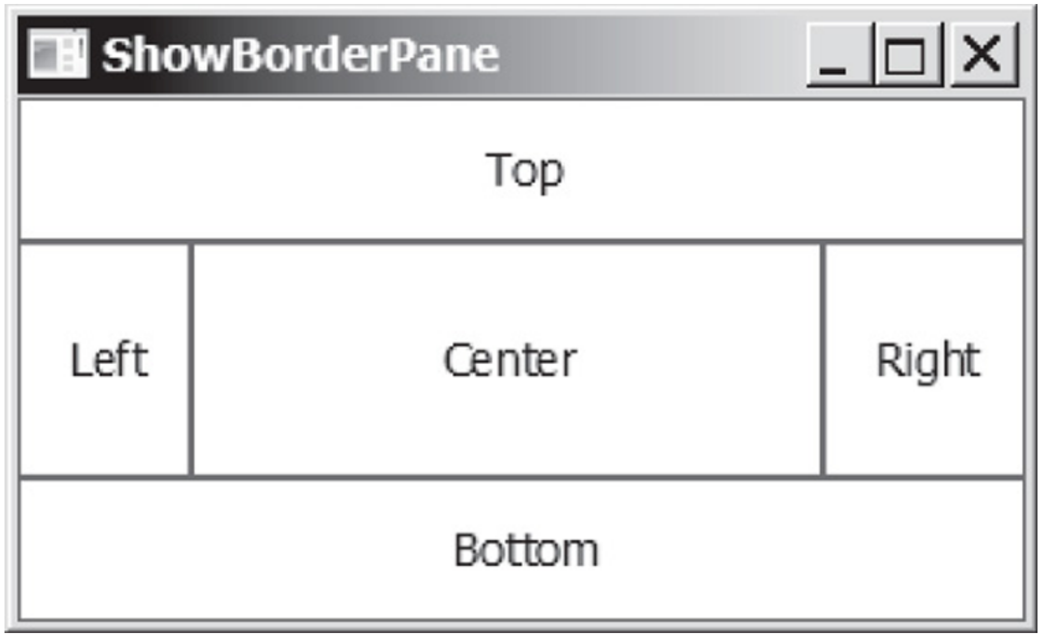
FlowPane


hGap
Top side
Pane
vGap
Right side
Left side
Bottom side


ShowFlowPane
First Name:
MI:
Last Name:


ShowFlowPane
First Name:
MI:
Last Name:

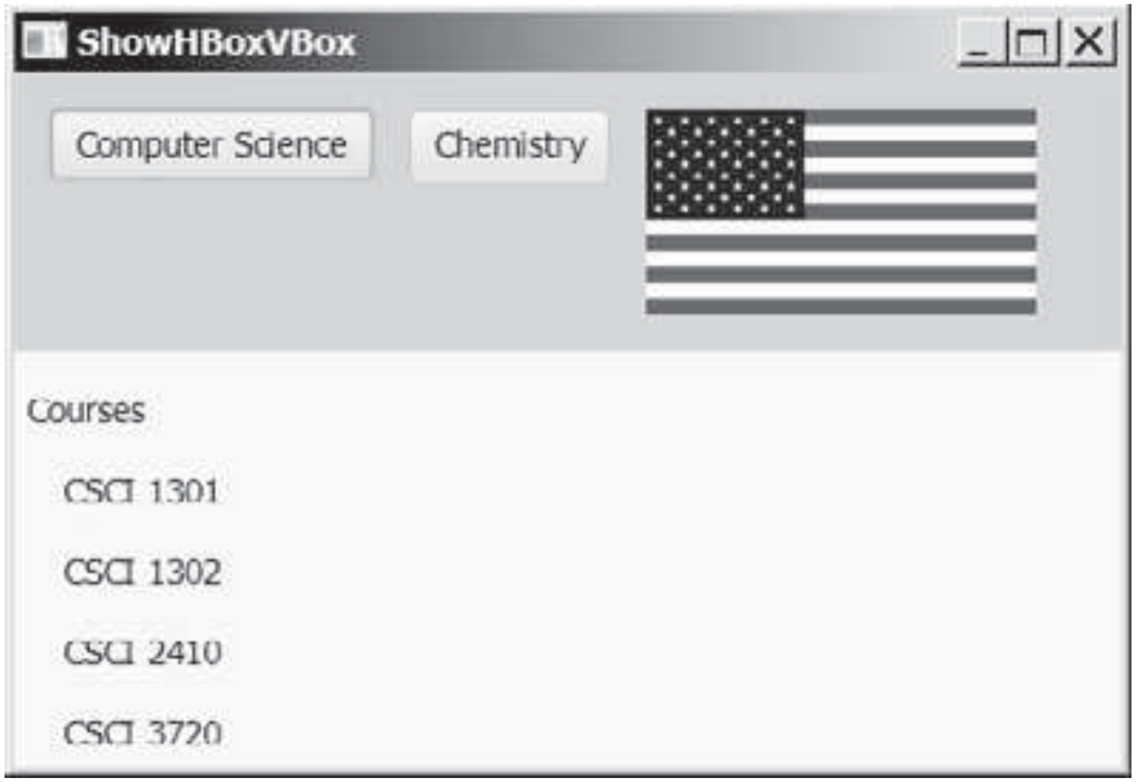
BorderPane



FYI: Code

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

     Scene scene = new Scene(pane);
     primaryStage.setTitle("ShowBorderPane");
     primaryStage.setScene(scene);
     primaryStage.show();
}

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));
     }
}
```

HBox and VBox



Courses

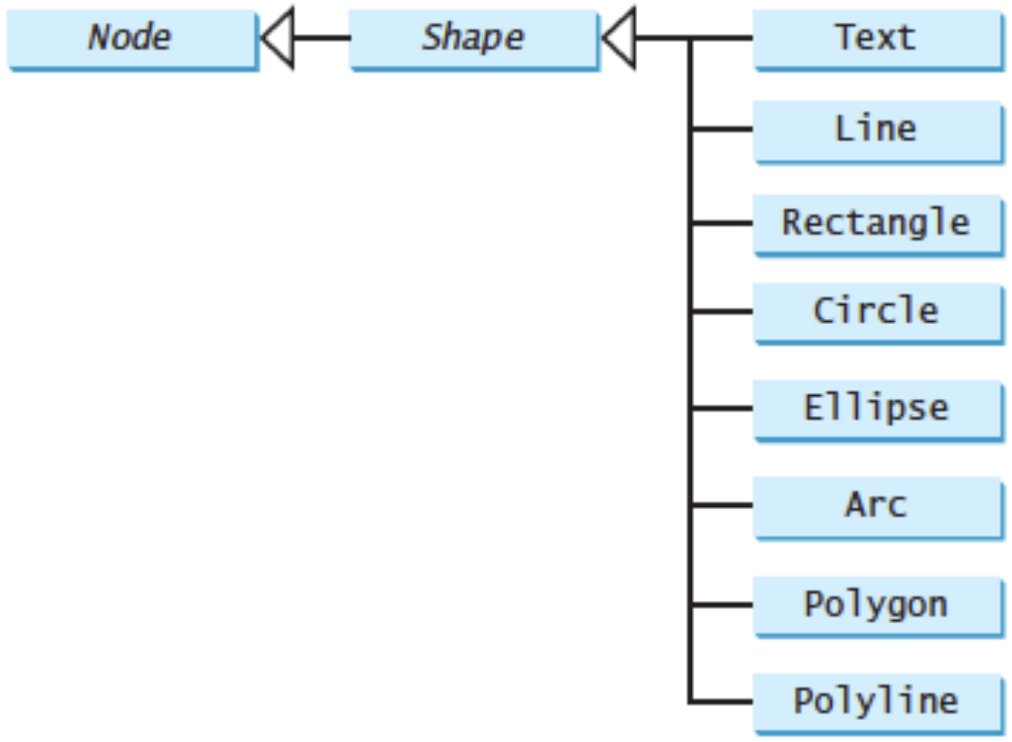
CSCI 1301

CSCI 1302

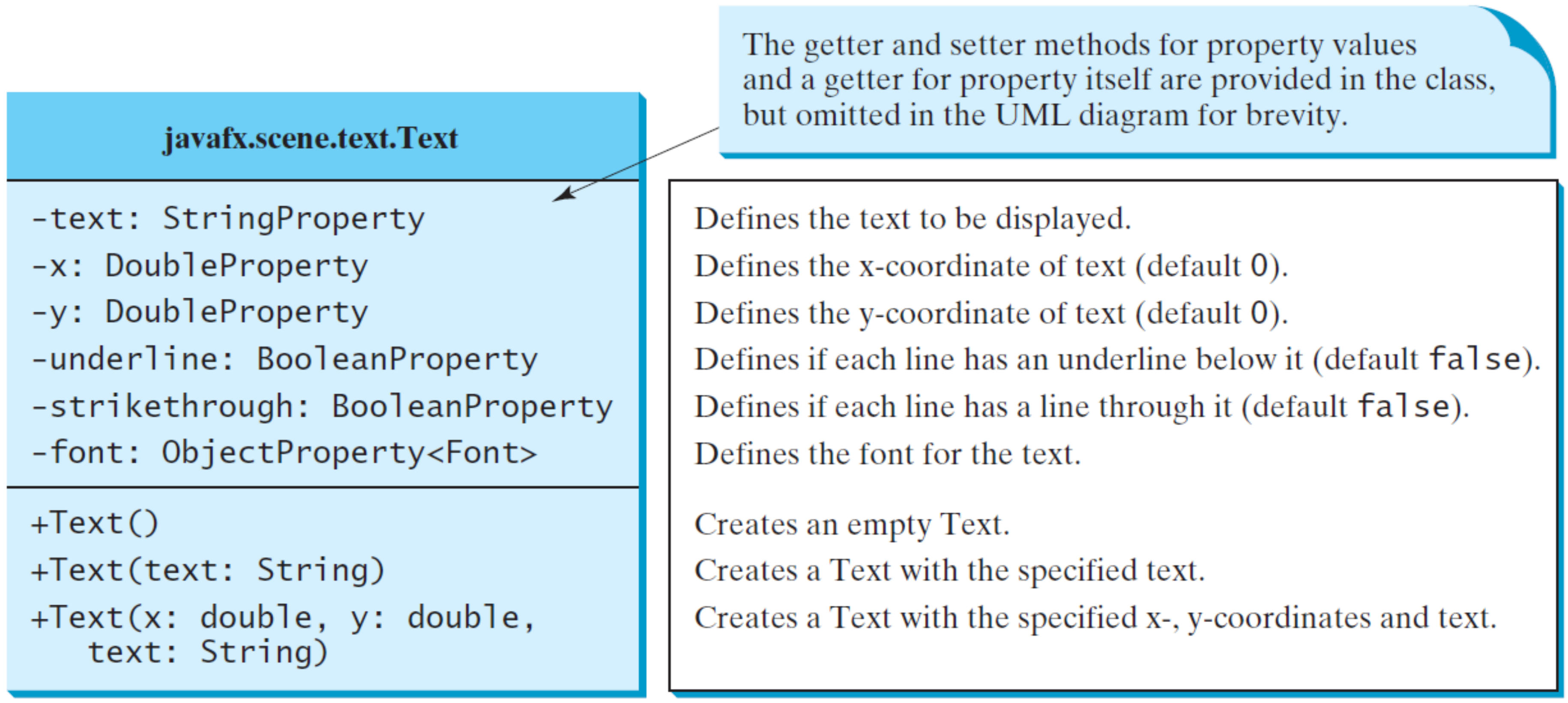
CSCI 2410

CSCI 3720

Shapes

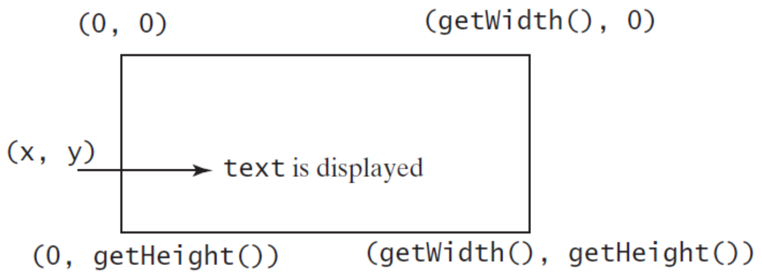

Node
Shape
Text
Line
Rectangle
Circle
Ellipse
Arc
Polygon
Polyline

Text



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.

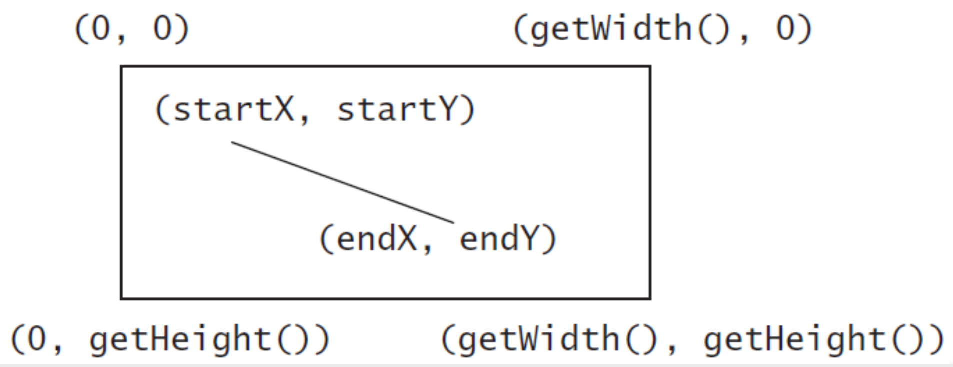
javafx.scene.text.Text	
-text: StringProperty	Defines the text to be displayed.
-x: DoubleProperty	Defines the x-coordinate of text (default 0).
-y: DoubleProperty	Defines the y-coordinate of text (default 0).
-underline: BooleanProperty	Defines if each line has an underline below it (default false).
-strikethrough: BooleanProperty	Defines if each line has a line through it (default false).
-font: ObjectProperty<Font>	Defines the font for the text.
+Text()	Creates an empty Text.
+Text(text: String)	Creates a Text with the specified text.
+Text(x: double, y: double, text: String)	Creates a Text with the specified x-, y-coordinates and text.



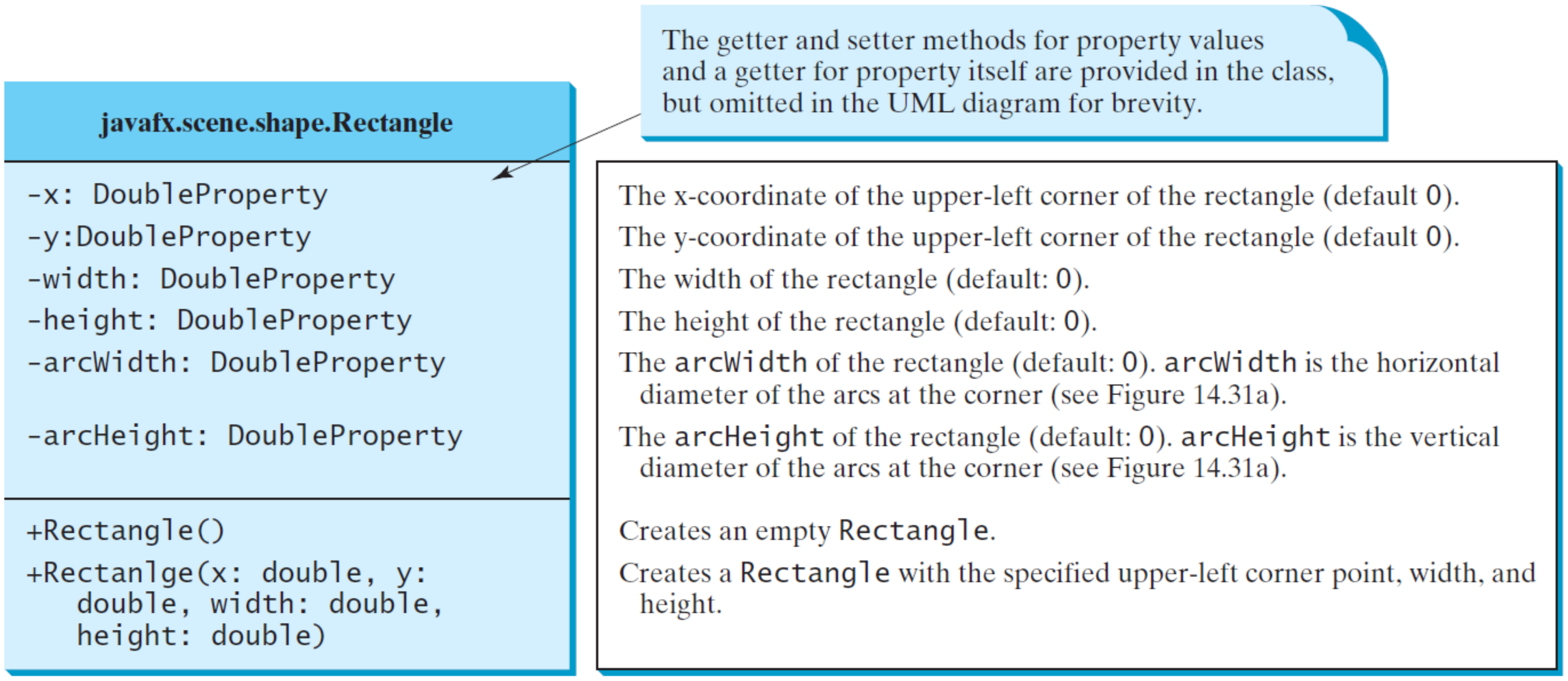
Line

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.shape.Line	
`-startX: DoubleProperty`	The x-coordinate of the start point.
`-startY: DoubleProperty`	The y-coordinate of the start point.
`-endX: DoubleProperty`	The x-coordinate of the end point.
`-endY: DoubleProperty`	The y-coordinate of the end point.
`+Line()`	Creates an empty `Line`.
`+Line(startX: double, startY: double, endX: double, endY: double)`	Creates a `Line` with the specified starting and ending points.

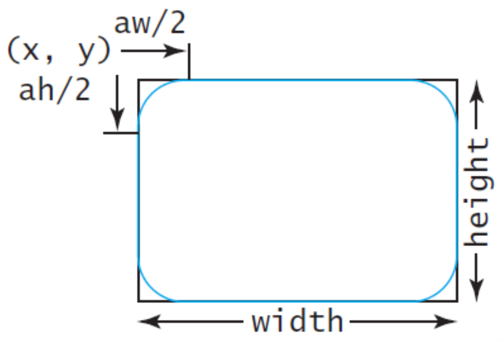


Rectangle

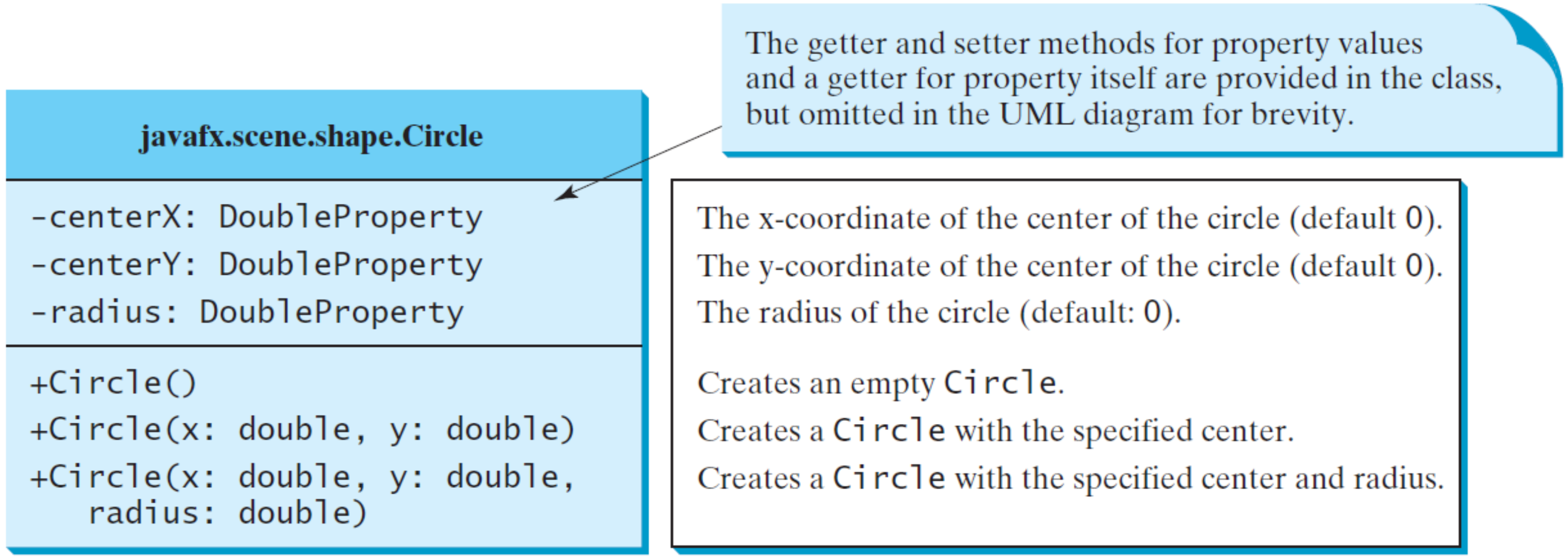


The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.shape.Rectangle	
`-x: DoubleProperty`	The x-coordinate of the upper-left corner of the rectangle (default 0).
`-y:DoubleProperty`	The y-coordinate of the upper-left corner of the rectangle (default 0).
`-width: DoubleProperty`	The width of the rectangle (default: 0).
`-height: DoubleProperty`	The height of the rectangle (default: 0).
`-arcWidth: DoubleProperty`	The `arcWidth` of the rectangle (default: 0). `arcWidth` is the horizontal diameter of the arcs at the corner (see Figure 14.31a).
`-arcHeight: DoubleProperty`	The `arcHeight` of the rectangle (default: 0). `arcHeight` is the vertical diameter of the arcs at the corner (see Figure 14.31a).
`+Rectangle()`	Creates an empty `Rectangle`.
`+Rectanlge(x: double, y: double, width: double, height: double)`	Creates a `Rectangle` with the specified upper-left corner point, width, and height.



Circle

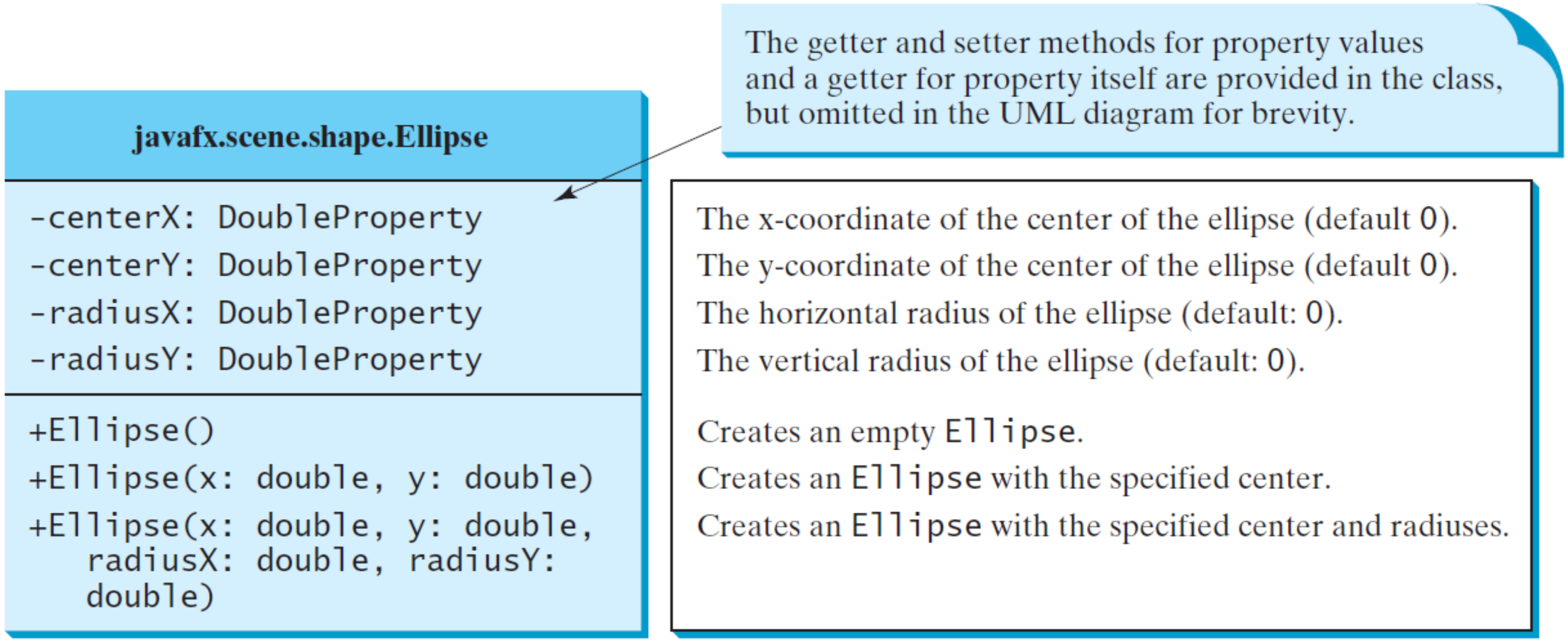


The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.shape.Circle

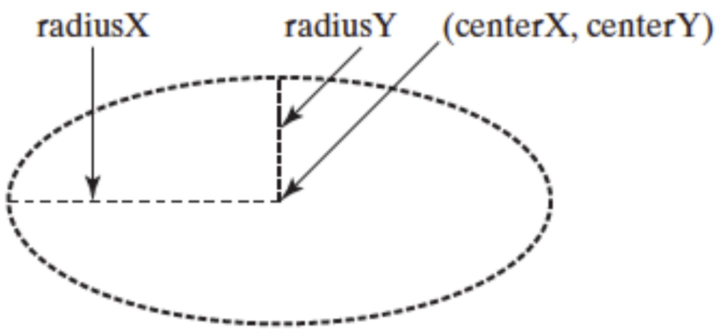
`-centerX: DoubleProperty`	The x-coordinate of the center of the circle (default `0`).
`-centerY: DoubleProperty`	The y-coordinate of the center of the circle (default `0`).
`-radius: DoubleProperty`	The radius of the circle (default: `0`).
`+Circle()`	Creates an empty `Circle`.
`+Circle(x: double, y: double)`	Creates a `Circle` with the specified center.
`+Circle(x: double, y: double, radius: double)`	Creates a `Circle` with the specified center and radius.

Ellipse

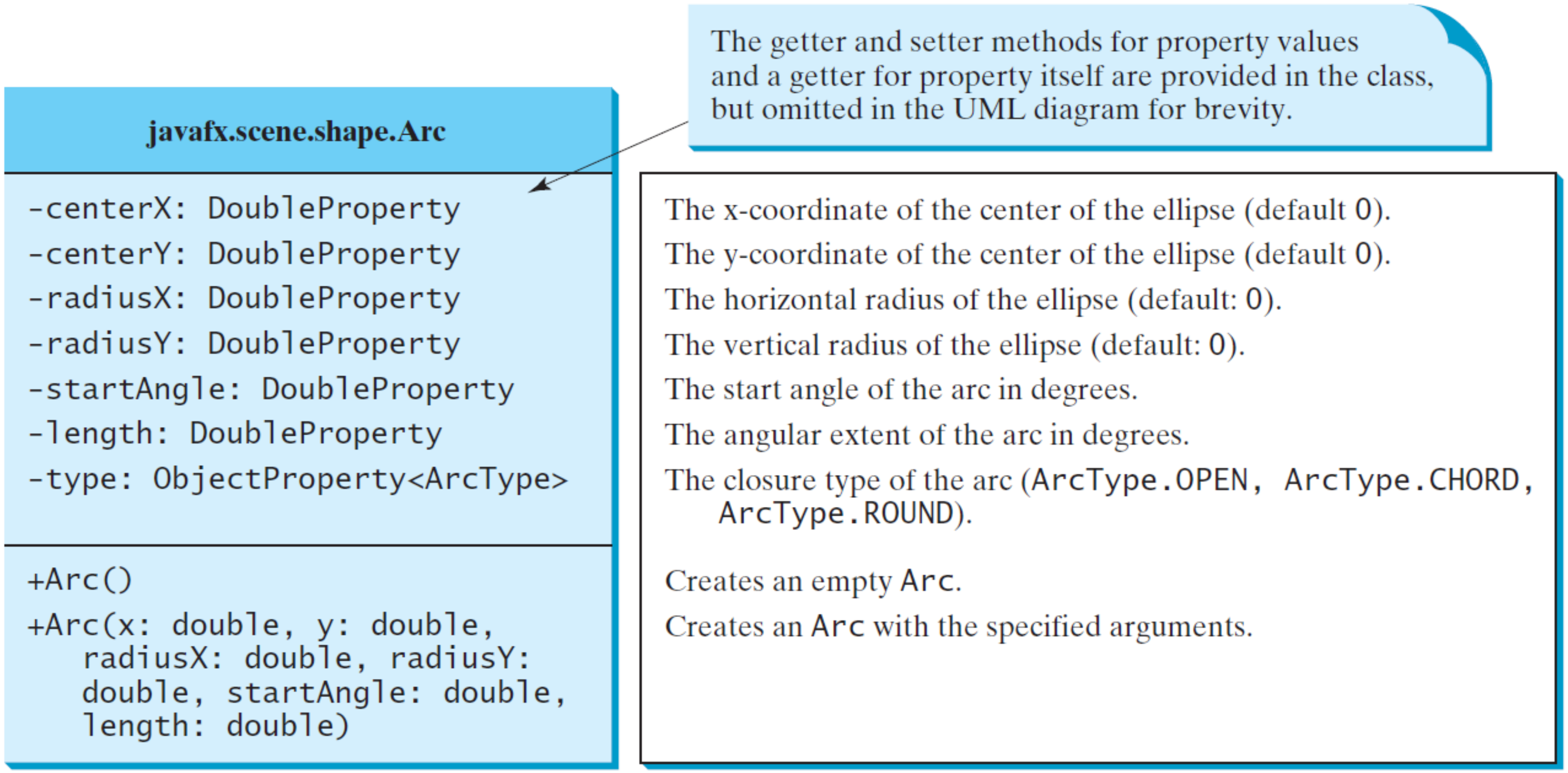


The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.shape.Ellipse	
`-centerX: DoubleProperty`	The x-coordinate of the center of the ellipse (default 0).
`-centerY: DoubleProperty`	The y-coordinate of the center of the ellipse (default 0).
`-radiusX: DoubleProperty`	The horizontal radius of the ellipse (default: 0).
`-radiusY: DoubleProperty`	The vertical radius of the ellipse (default: 0).
`+Ellipse()`	Creates an empty `Ellipse`.
`+Ellipse(x: double, y: double)`	Creates an `Ellipse` with the specified center.
`+Ellipse(x: double, y: double, radiusX: double, radiusY: double)`	Creates an `Ellipse` with the specified center and radiuses.



Arc (1)

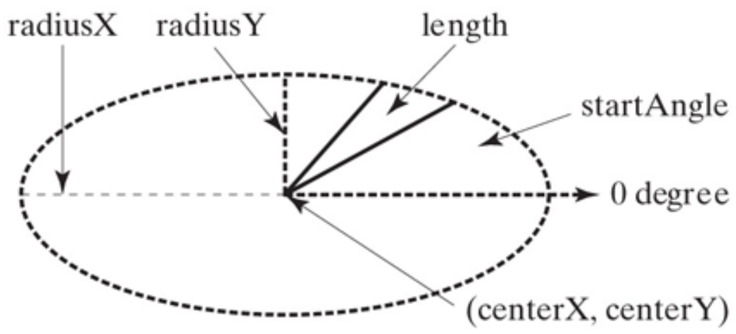


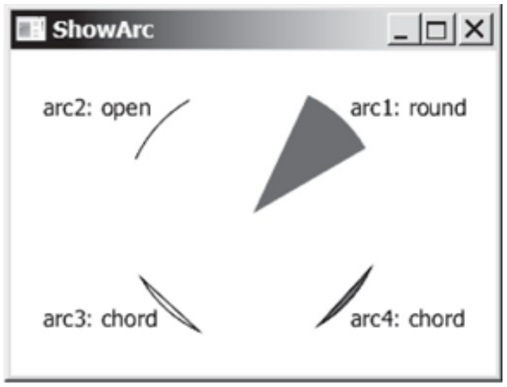
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

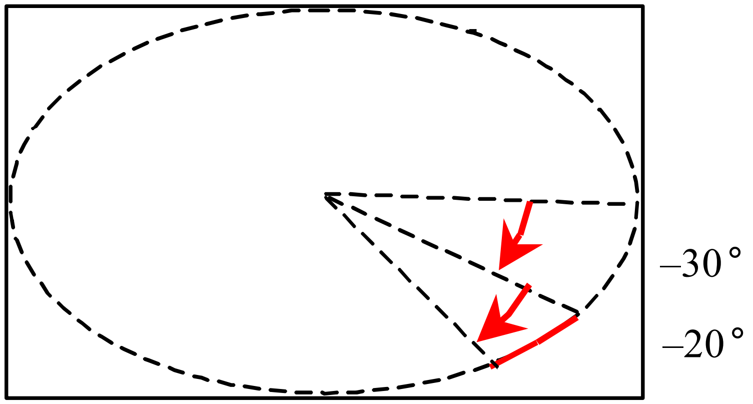
javafx.scene.shape.Arc

`-centerX: DoubleProperty`	The x-coordinate of the center of the ellipse (default 0).
`-centerY: DoubleProperty`	The y-coordinate of the center of the ellipse (default 0).
`-radiusX: DoubleProperty`	The horizontal radius of the ellipse (default: 0).
`-radiusY: DoubleProperty`	The vertical radius of the ellipse (default: 0).
`-startAngle: DoubleProperty`	The start angle of the arc in degrees.
`-length: DoubleProperty`	The angular extent of the arc in degrees.
`-type: ObjectProperty<ArcType>`	The closure type of the arc (`ArcType.OPEN`, `ArcType.CHORD`, `ArcType.ROUND`).
`+Arc()`	Creates an empty `Arc`.
`+Arc(x: double, y: double, radiusX: double, radiusY: double, startAngle: double, length: double)`	Creates an `Arc` with the specified arguments.

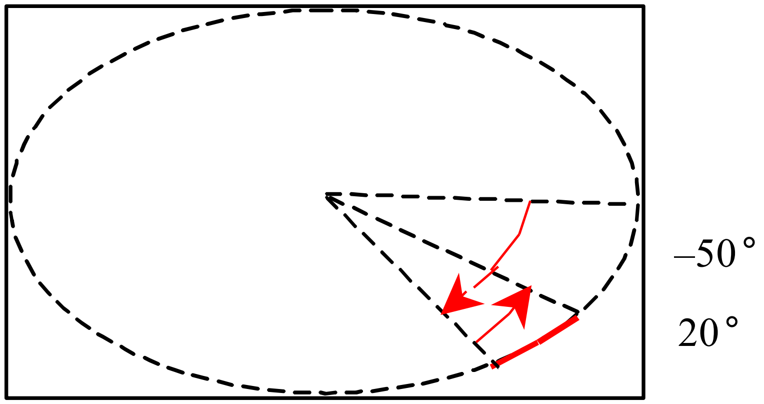
Arc (2)





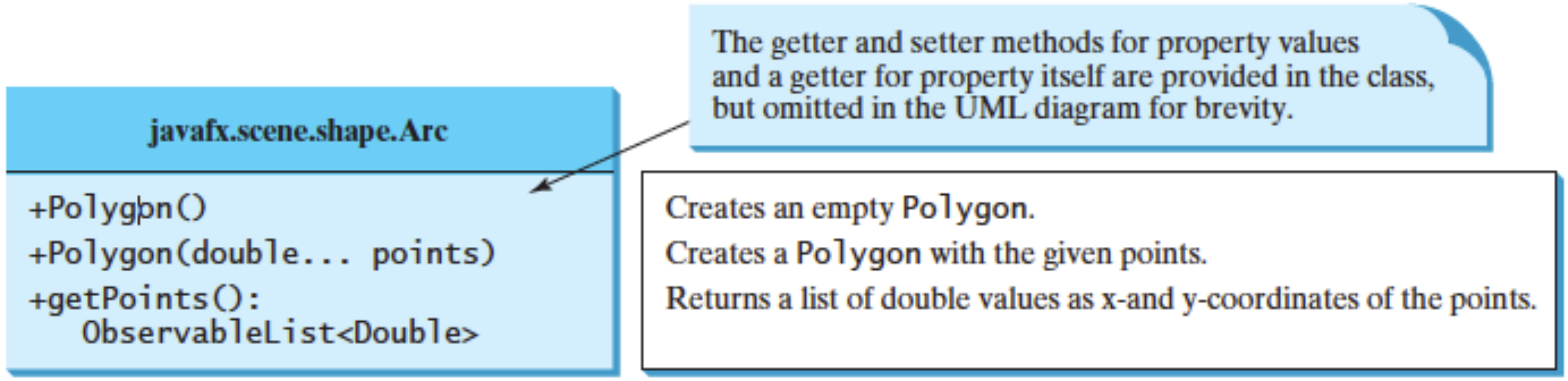


(a) Negative starting angle –30° and negative spanning angle –20°



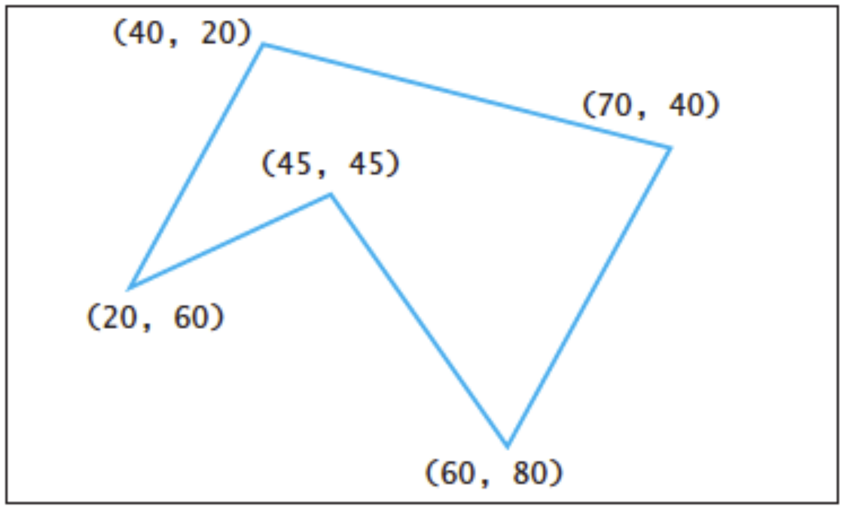
(b) Negative starting angle –50° and positive spanning angle 20°

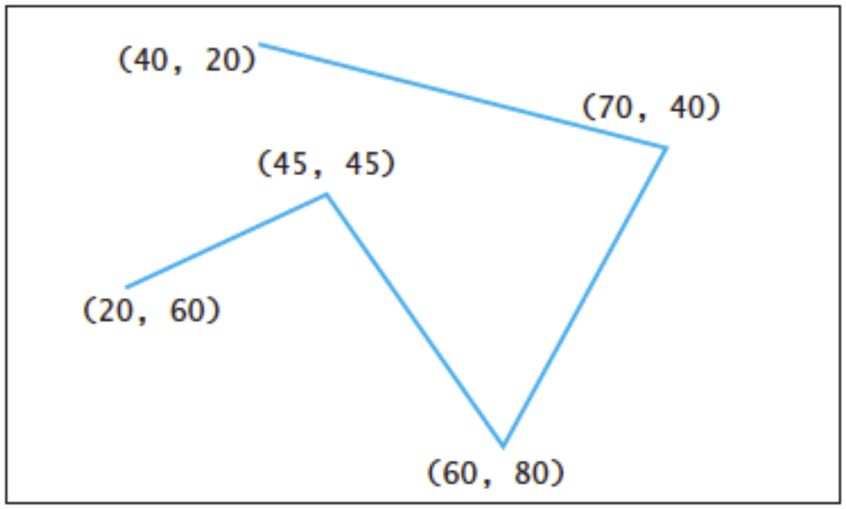
Polygon and Polyline



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.shape.Arc	
+Polygon()	Creates an empty Polygon.
+Polygon(double... points)	Creates a Polygon with the given points.
+getPoints(): ObservableList<Double>	Returns a list of double values as x-and y-coordinates of the points.

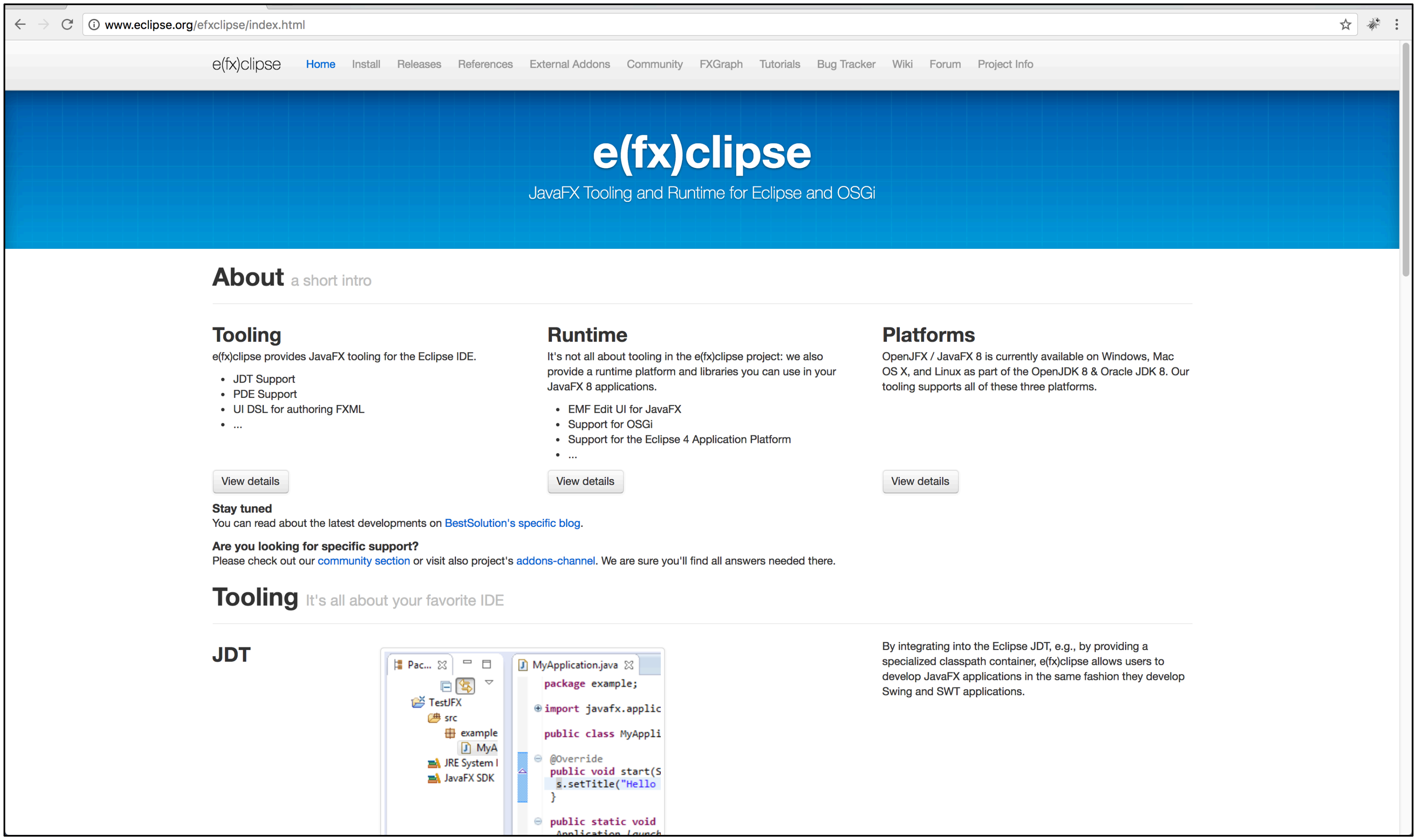




e(fx)clipse

- Provides JavaFX tooling for the Eclipse
- http://www.eclipse.org/efxclipse/

Installing e(fx)clipse (1)



e(fx)clipse

JavaFX Tooling and Runtime for Eclipse and OSGi

About a short intro

Tooling

e(fx)clipse provides JavaFX tooling for the Eclipse IDE.

- JDT Support
- PDE Support
- UI DSL for authoring FXML
- ...

View details

Runtime

It's not all about tooling in the e(fx)clipse project: we also provide a runtime platform and libraries you can use in your JavaFX 8 applications.

- EMF Edit UI for JavaFX
- Support for OSGi
- Support for the Eclipse 4 Application Platform
- ...

View details

Platforms

OpenJFX / JavaFX 8 is currently available on Windows, Mac OS X, and Linux as part of the OpenJDK 8 & Oracle JDK 8. Our tooling supports all of these three platforms.

View details

Stay tuned
You can read about the latest developments on BestSolution's specific blog.

Are you looking for specific support?
Please check out our community section or visit also project's addons-channel. We are sure you'll find all answers needed there.

Tooling It's all about your favorite IDE

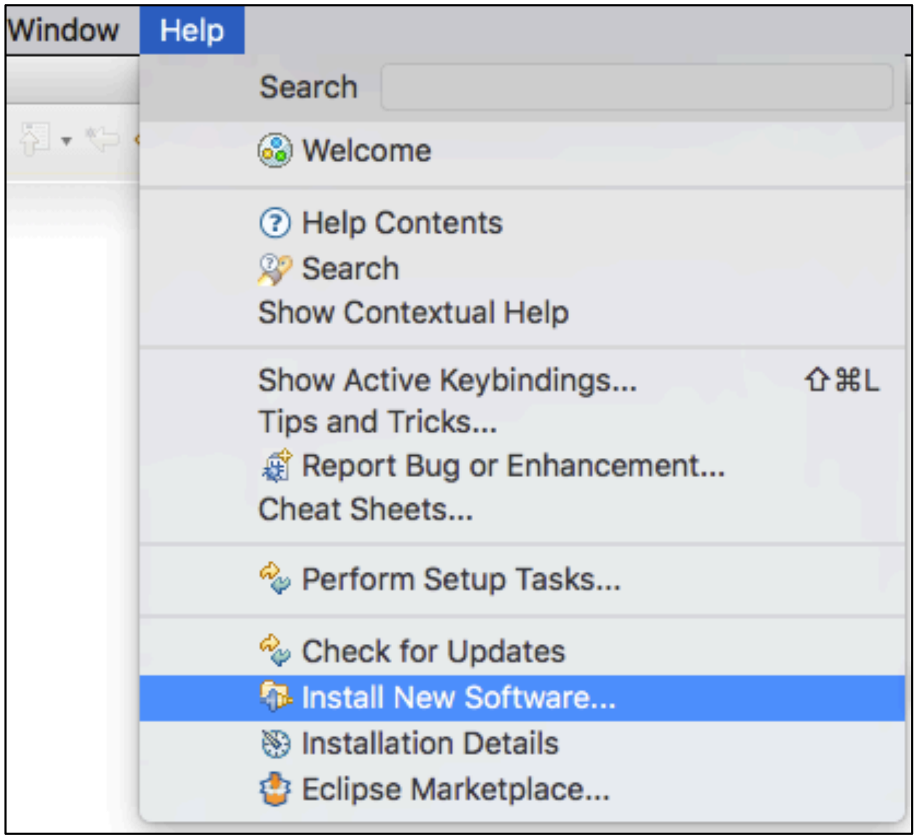
JDT

```
package example;

import javafx.applic

public class MyAppli

  @Override
  public void start(S
    s.setTitle("Hello
  }

  public static void
```

By integrating into the Eclipse JDT, e.g., by providing a specialized classpath container, e(fx)clipse allows users to develop JavaFX applications in the same fashion they develop Swing and SWT applications.

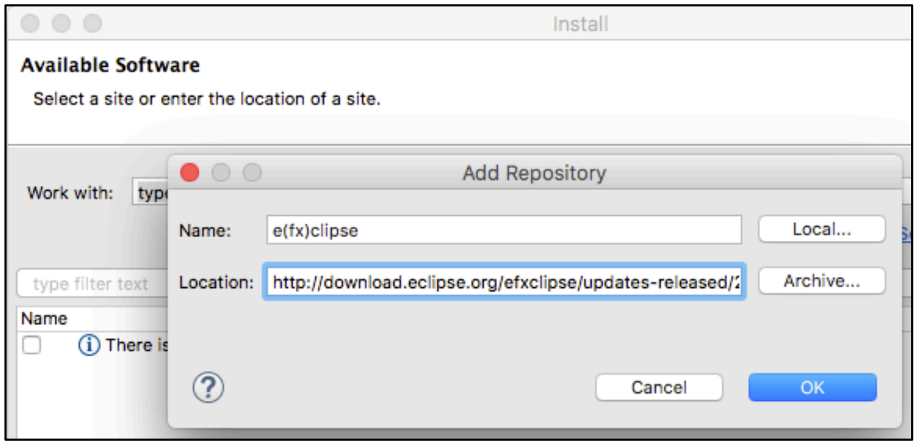
Installing e(fx)clipse (2)

- Eclipse -> Help -> Install New Software



Installing e(fx)clipse (3)

- Add location of e(fx)clipse update site
- http://download.eclipse.org/efxclipse/updates-released/2.4.0/site


Install

Available Software

Select a site or enter the location of a site.

Work with: typ

type filter text

Name

There is

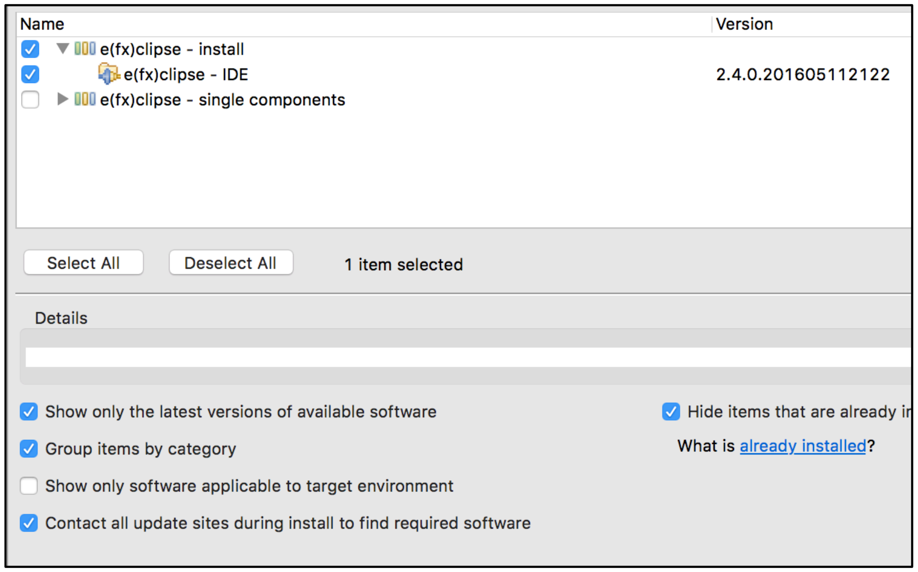
Add Repository

Name: e(fx)clipse

Location: http://download.eclipse.org/efxclipse/updates-released/2

Installing e(fx)clipse (4)

- Select “e(fx)clipse - install” -> “e(fx)clipse - IDE”



Name	Version
✓ ▼ e(fx)clipse - install	
✓ e(fx)clipse - IDE	2.4.0.201605112122
☐ ▶ e(fx)clipse - single components	

Select All | Deselect All | 1 item selected

Details

✓ Show only the latest versions of available software

✓ Hide items that are already i

What is already installed?

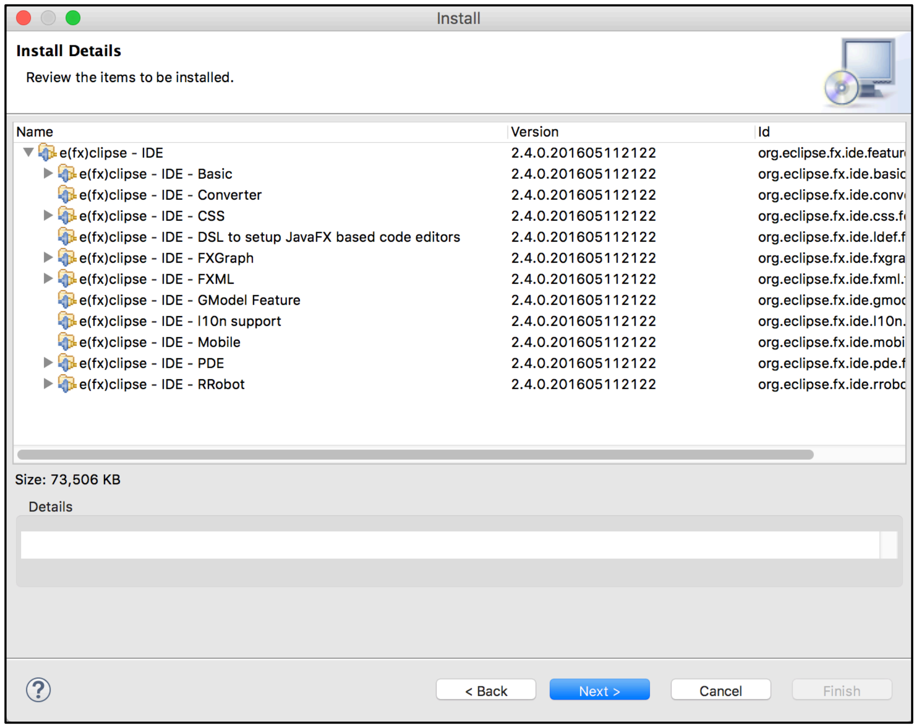
✓ Group items by category

☐ Show only software applicable to target environment

✓ Contact all update sites during install to find required software

Installing e(fx)clipse (5)

- Next

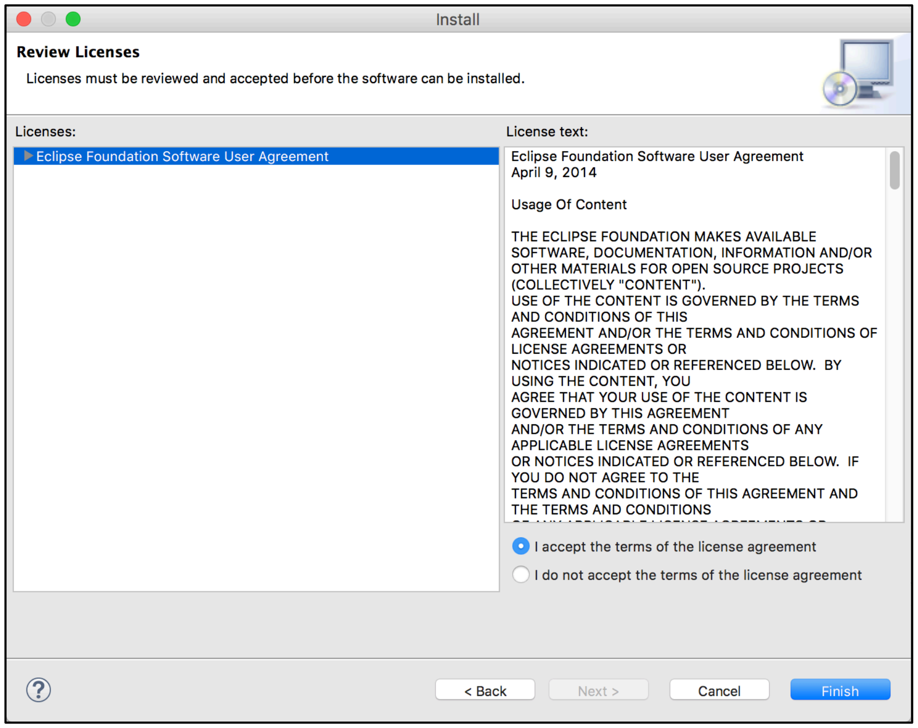


Name	Version	Id
e(fx)clipse - IDE	2.4.0.201605112122	org.eclipse.fx.ide.featur
e(fx)clipse - IDE - Basic	2.4.0.201605112122	org.eclipse.fx.ide.basic
e(fx)clipse - IDE - Converter	2.4.0.201605112122	org.eclipse.fx.ide.conv
e(fx)clipse - IDE - CSS	2.4.0.201605112122	org.eclipse.fx.ide.css.f
e(fx)clipse - IDE - DSL to setup JavaFX based code editors	2.4.0.201605112122	org.eclipse.fx.ide.ldef.f
e(fx)clipse - IDE - FXGraph	2.4.0.201605112122	org.eclipse.fx.ide.fxgra
e(fx)clipse - IDE - FXML	2.4.0.201605112122	org.eclipse.fx.ide.fxml.
e(fx)clipse - IDE - GModel Feature	2.4.0.201605112122	org.eclipse.fx.ide.gmod
e(fx)clipse - IDE - l10n support	2.4.0.201605112122	org.eclipse.fx.ide.l10n.
e(fx)clipse - IDE - Mobile	2.4.0.201605112122	org.eclipse.fx.ide.mobi
e(fx)clipse - IDE - PDE	2.4.0.201605112122	org.eclipse.fx.ide.pde.f
e(fx)clipse - IDE - RRobot	2.4.0.201605112122	org.eclipse.fx.ide.rrobo

Size: 73,506 KB

Installing e(fx)clipse (6)

- Agree, Finish



Review Licenses

Licenses must be reviewed and accepted before the software can be installed.

Licenses:

Eclipse Foundation Software User Agreement

License text:

Eclipse Foundation Software User Agreement
April 9, 2014

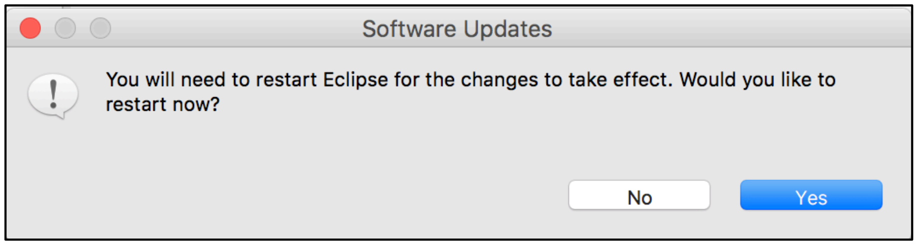
Usage Of Content

THE ECLIPSE FOUNDATION MAKES AVAILABLE SOFTWARE, DOCUMENTATION, INFORMATION AND/OR OTHER MATERIALS FOR OPEN SOURCE PROJECTS (COLLECTIVELY "CONTENT").
USE OF THE CONTENT IS GOVERNED BY THE TERMS AND CONDITIONS OF THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. BY USING THE CONTENT, YOU AGREE THAT YOUR USE OF THE CONTENT IS GOVERNED BY THIS AGREEMENT AND/OR THE TERMS AND CONDITIONS OF ANY APPLICABLE LICENSE AGREEMENTS OR NOTICES INDICATED OR REFERENCED BELOW. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT AND THE TERMS AND CONDITIONS

[✓ I accept the terms of the license agreement]
I do not accept the terms of the license agreement

Installing e(fx)clipse (7)

- Restart

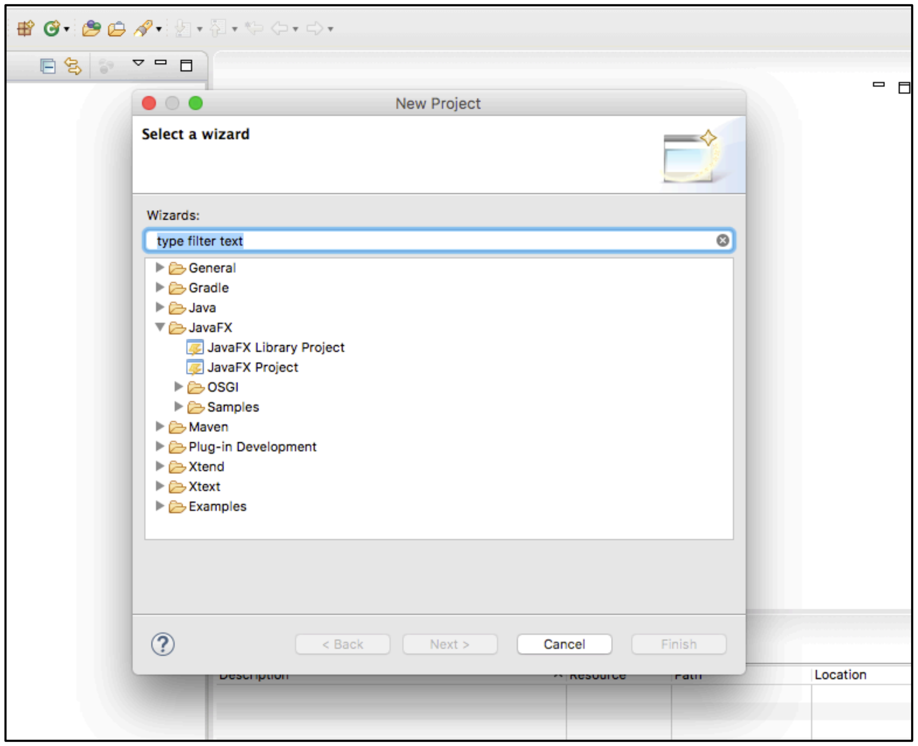


Software Updates

You will need to restart Eclipse for the changes to take effect. Would you like to restart now?

Using e(fx)clipse

- File -> New -> Project



New Project

Select a wizard

Wizards:

type filter text

- General
- Gradle
- Java
- JavaFX
  - JavaFX Library Project
  - JavaFX Project
  - OSGI
  - Samples
- Maven
- Plug-in Development
- Xtend
- Xtext
- Examples

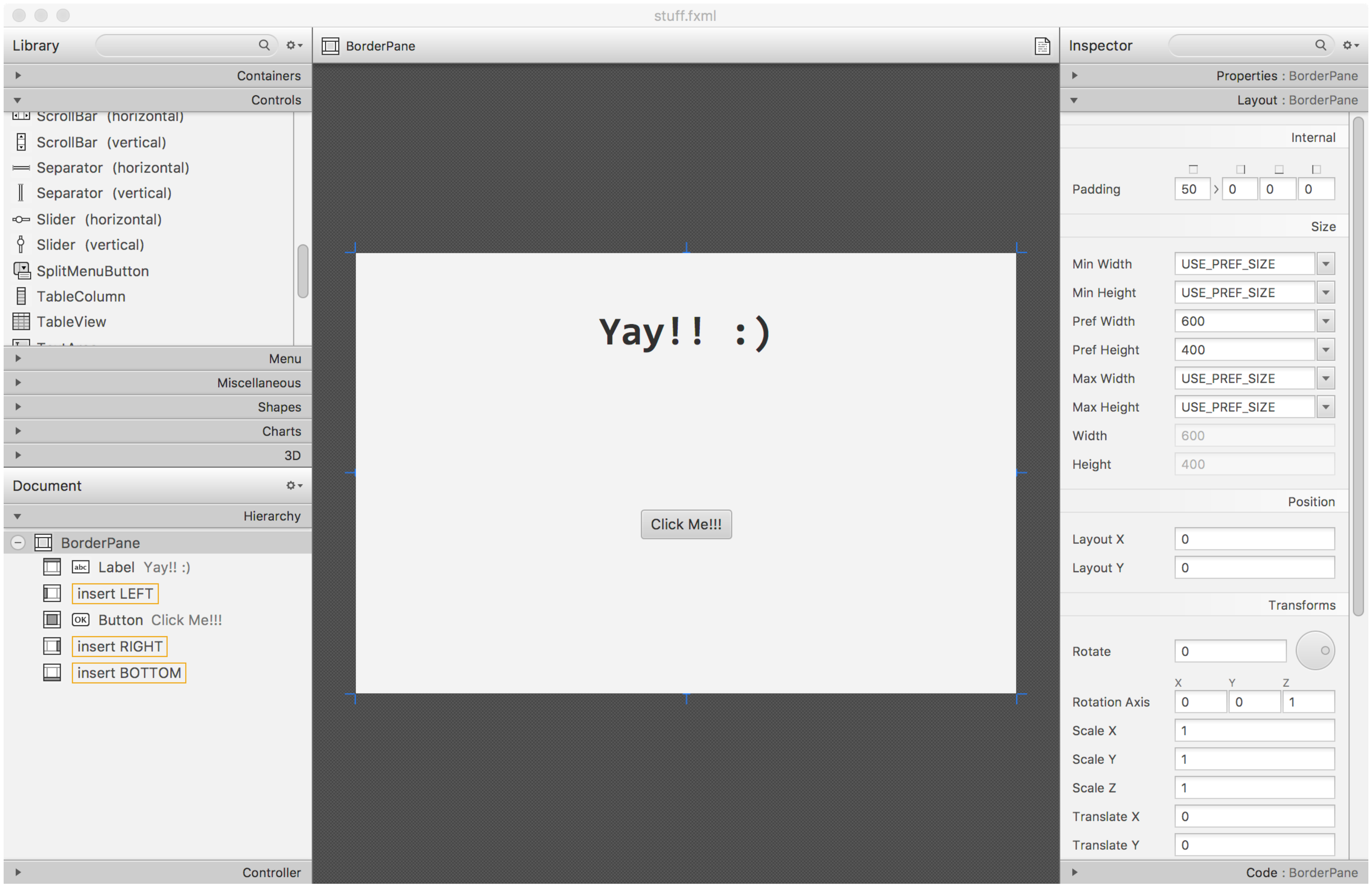
SceneBuilder

- A Visual Layout Tool for JavaFX Applications
- Quickly design JavaFX GUI via drag-and-drop components that write an FXML file
- FXML file can be combined with a Java project
- http://gluonhq.com/products/scene-builder/

Using SceneBuilder

- Install
- In Eclipse -> Preferences -> JavaFX
  - Set path to SceneBuilder
- In Project, New -> Other -> JavaFX -> New FXML Document
- Right click -> Open with SceneBuilder

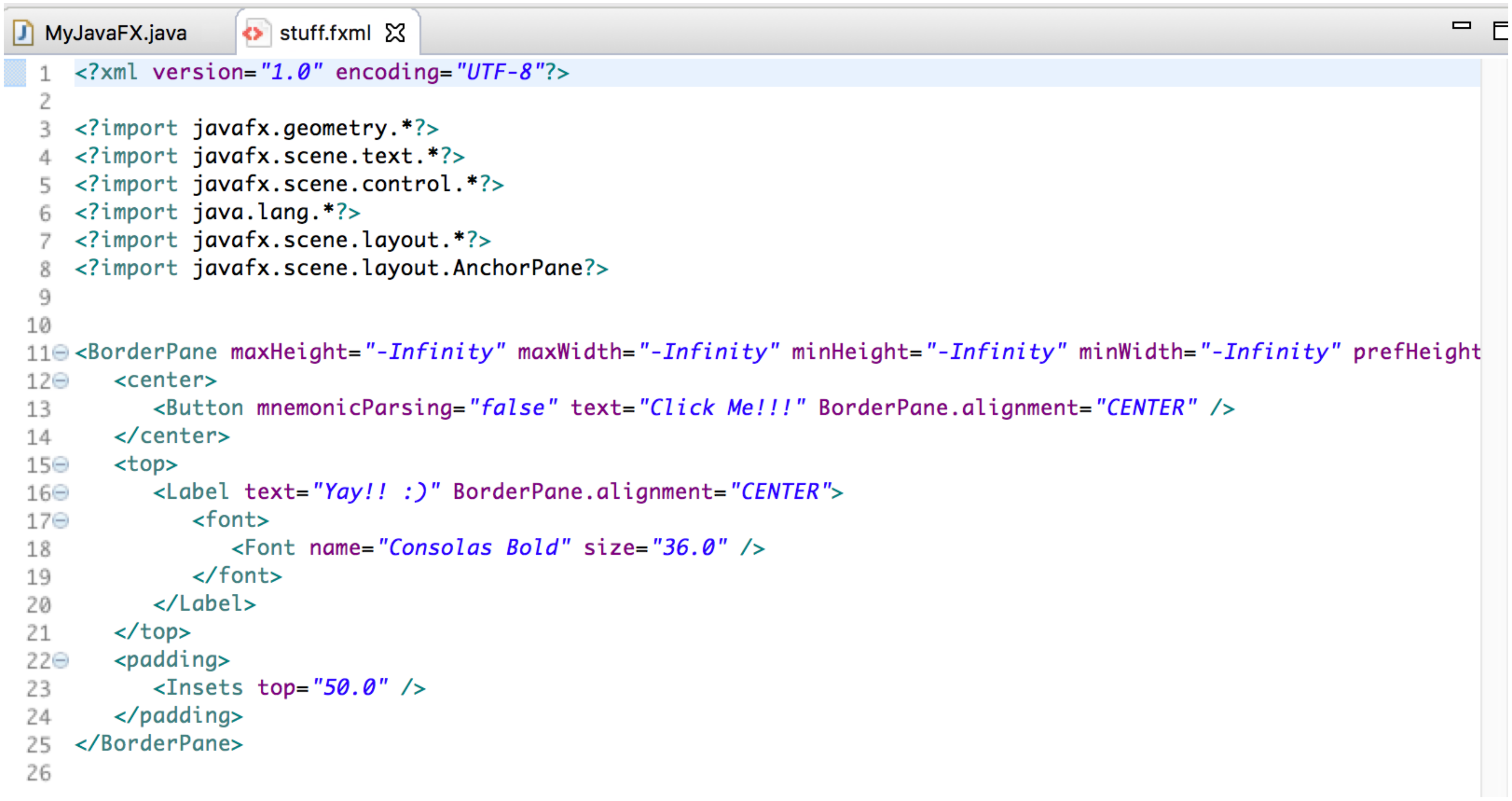
Example



- BorderPane
  - Label Yay!! :)
  - insert LEFT
  - Button Click Me!!!
  - insert RIGHT
  - insert BOTTOM

Padding	50, 0, 0, 0
Min Width	USE_PREF_SIZE
Min Height	USE_PREF_SIZE
Pref Width	600
Pref Height	400
Max Width	USE_PREF_SIZE
Max Height	USE_PREF_SIZE
Width	600
Height	400
Layout X	0
Layout Y	0
Rotate	0
Rotation Axis	X 0, Y 0, Z 1
Scale X	1
Scale Y	1
Scale Z	1
Translate X	0
Translate Y	0

Corresponding FXML



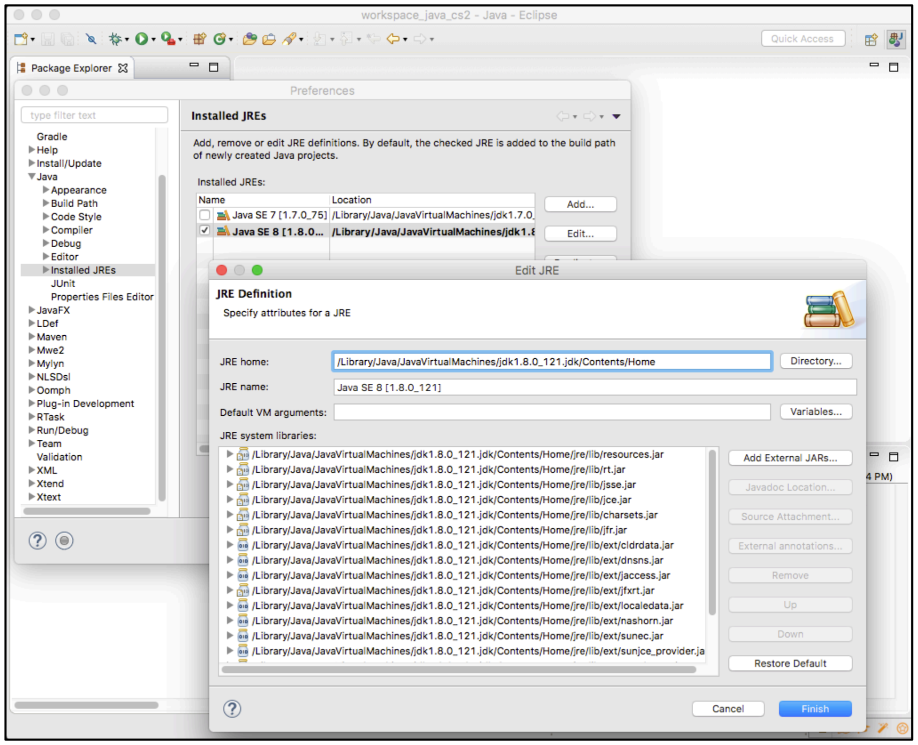
```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.*?>
<?import javafx.scene.text.*?>
<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.AnchorPane?>


<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight
   <center>
      <Button mnemonicParsing="false" text="Click Me!!!" BorderPane.alignment="CENTER" />
   </center>
   <top>
      <Label text="Yay!! :)" BorderPane.alignment="CENTER">
         <font>
            <Font name="Consolas Bold" size="36.0" />
         </font>
      </Label>
   </top>
   <padding>
      <Insets top="50.0" />
   </padding>
</BorderPane>

```

Code

```
@Override
public void start(Stage primaryStage) throws Exception {
    Parent root = FXMLLoader.load(getClass().getResource("stuff.fxml"));
    Scene scene = new Scene(root);
    primaryStage.setTitle("MyExampleApp");
    primaryStage.setScene(scene);
    primaryStage.show();
}
```

Potential Issue

- Update to latest JDK
- Eclipse
  - Eclipse -> Preferences -> Java -> Installed JREs
  - Update it to latest JRE version



Installed JREs

Add, remove or edit JRE definitions. By default, the checked JRE is added to the build path of newly created Java projects.

Installed JREs:

	Name	Location
☐	Java SE 7 [1.7.0_75]	/Library/Java/JavaVirtualMachines/jdk1.7.0_
✓	**Java SE 8 [1.8.0...**	**/Library/Java/JavaVirtualMachines/jdk1.8**

JRE Definition

Specify attributes for a JRE

JRE home: /Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home

JRE name: Java SE 8 [1.8.0_121]

Default VM arguments:

JRE system libraries:

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/resources.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/rt.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/jsse.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/jce.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/charsets.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/jfr.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/ext/cldrdata.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/ext/dnsns.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/ext/jaccess.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/ext/jfxrt.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/ext/localedata.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/ext/nashorn.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/ext/sunec.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_121.jdk/Contents/Home/jre/lib/ext/sunjce_provider.ja
```

```
WARNING: Loading FXML document with JavaFX API of version 8.0.111 by JavaFX runtime of version 8.0.11
Exception in thread "Thread-1"
```

Take Home Points

- You have now seen the basics of using JavaFX for creating **graphical user interfaces (GUIs)**

- Start playing around!
  - This will be necessary for your project
  - Install the tool(s) you plan to use

- More to come: how to respond to events (e.g. user clicks a button)