Chapter 14 JavaFX Basics

Motivations

JavaFX : new framework
for Java GUI programming.

JavaFX API : good example of OOP.

* Online documentation:
https://openjfx.io/javadoc/20/

JavaFX vs Swing and AWT

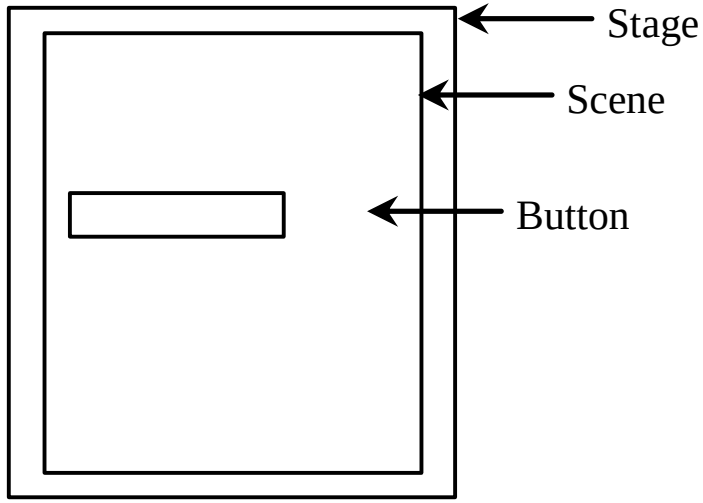
GUI classLibrary/platform :

- ***JavaFX:*** from Java 8, support for rich Internet applications.

- Old:
  - *AWT:*
    - *earliest,*
    - *for* simple GUI;
    - prone to platform-specific bugs.
  - *Swing:*
    - more robust and flexible
    - Replaced by JavaFX

Basic Structure of JavaFX

- extend **Application** class
  - Override the **start(Stage)** method
- **Stage** (Window), **Scene**, **Button** (Node)
  - Stage is a window for displaying a scene that contains nodes




```
1  import javafx.application.Application;
2  import javafx.scene.Scene;
3  import javafx.scene.control.Button;
4  import javafx.stage.Stage;
```

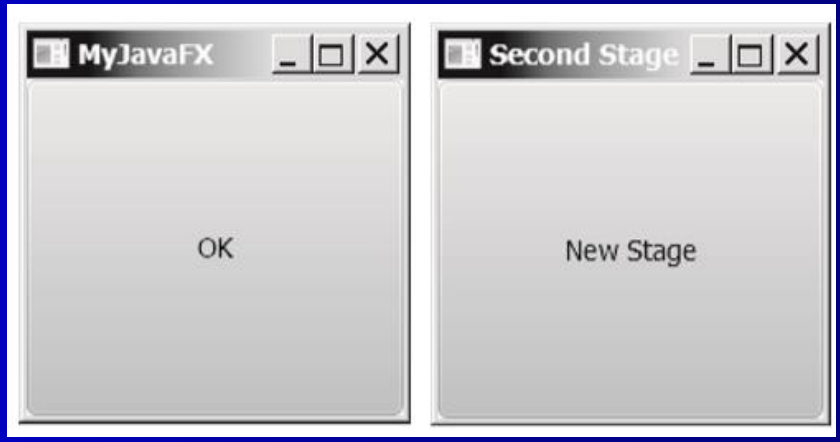
https://openjfx.io/javadoc/20/javafx.graphics/module-summary.html

MyJavaFX

A simple JavaFX displays a button in the window.

```
6  public class MyJavaFX 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     Button btOK = new Button("OK");
11     Scene scene = new Scene(btOK, 200, 250);
12     primaryStage.setTitle("MyJavaFX"); // Set the stage title
13     primaryStage.setScene(scene); // Place the scene in the stage
14     primaryStage.show(); // Display the stage
15   }
16
17   /**
18    * The main method is only needed for the IDE with limited
19    * JavaFX support. Not needed for running from the command line.
20    */
21   public static void main(String[] args) {
22     Application.launch(args);
23   }
24 }
```

Multiple Stages in one prgram




```
 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

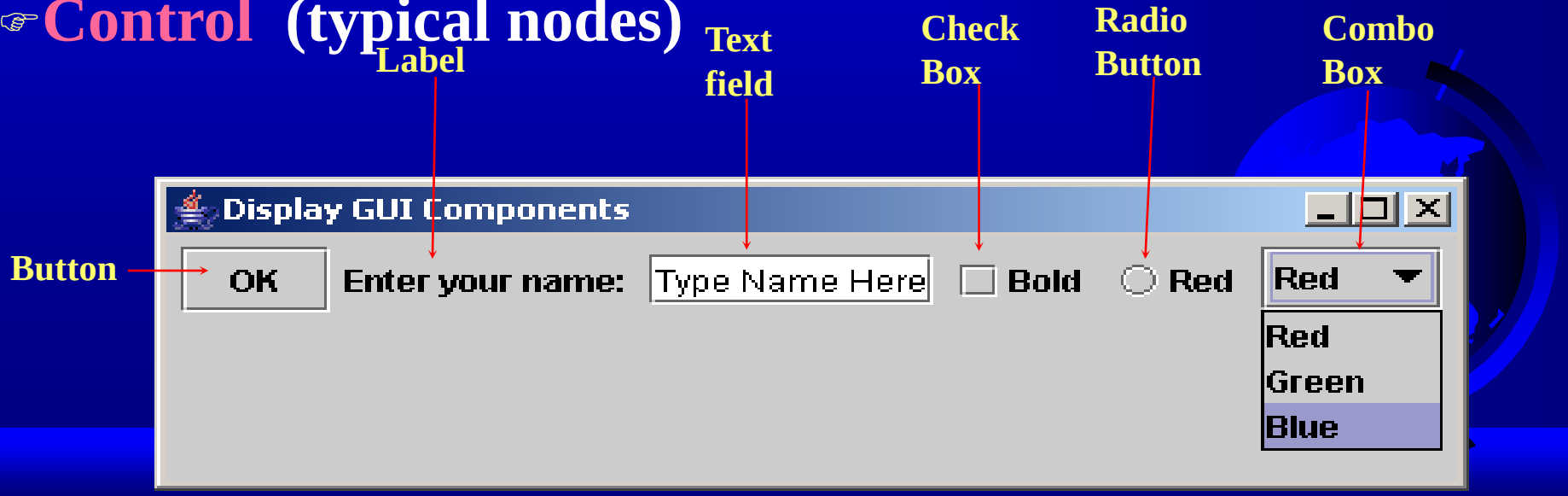
create second stage

display second stage

main method omitted

Pane, Control

- **Pane** :
  - place nodes in a pane ; place the pane in a scene
  - container class: like a canvas of a set of nodes
    - to layout them in a location and size, eg.:
    - StackPane: places nodes in the center of the pane on top of each other
- **Control (typical nodes)**



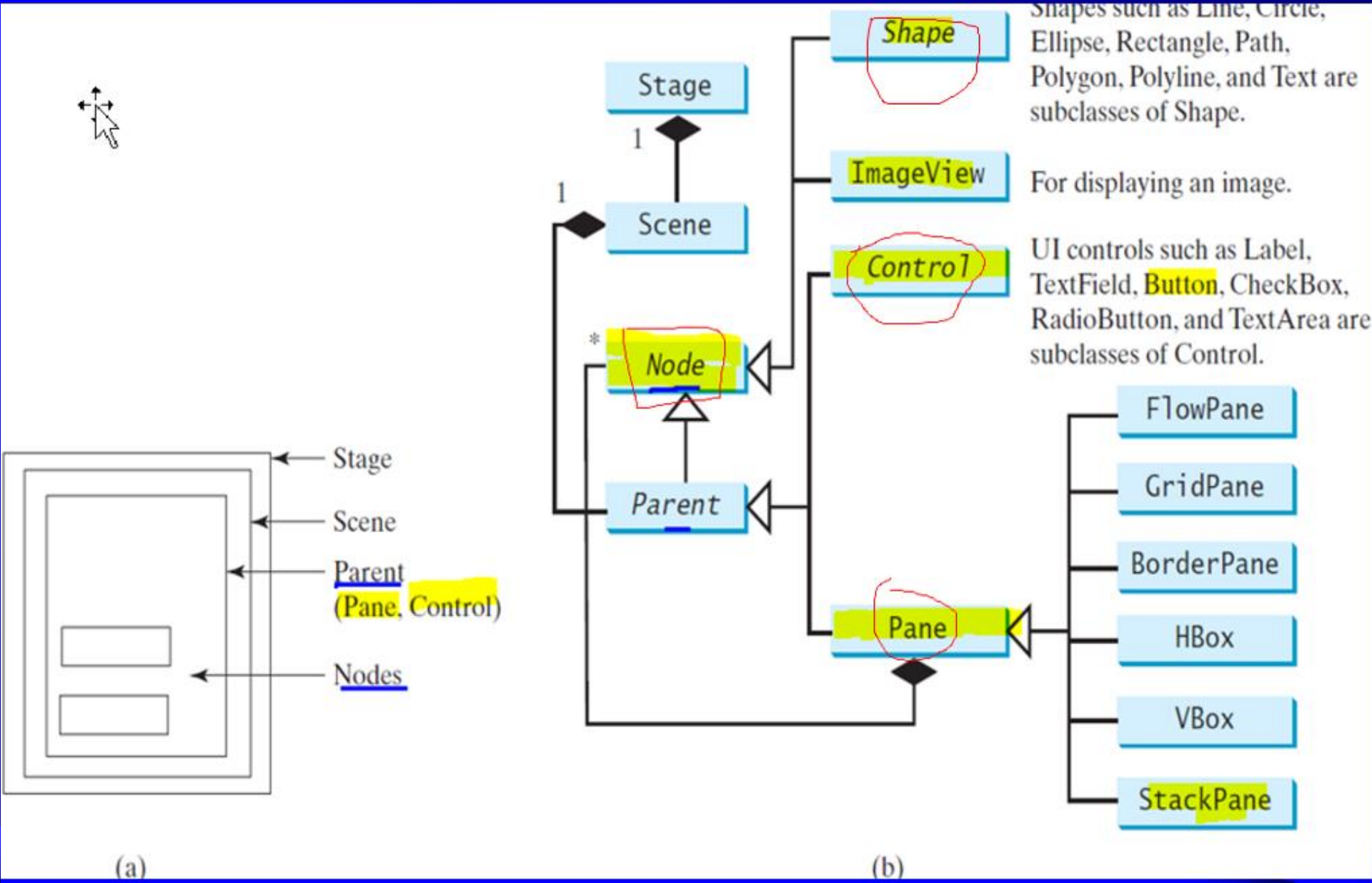
ButtonInPane.java

Pane, Control

```
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 }
```

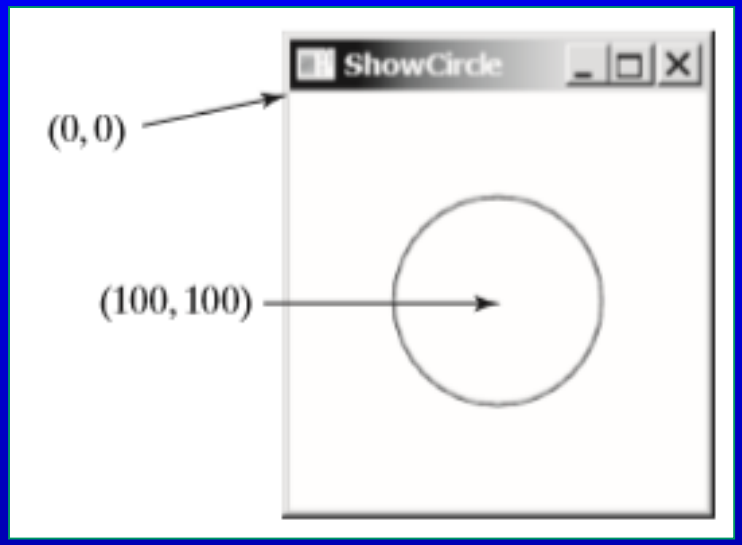
button is placed in the center of the pane.

subclasses of Node

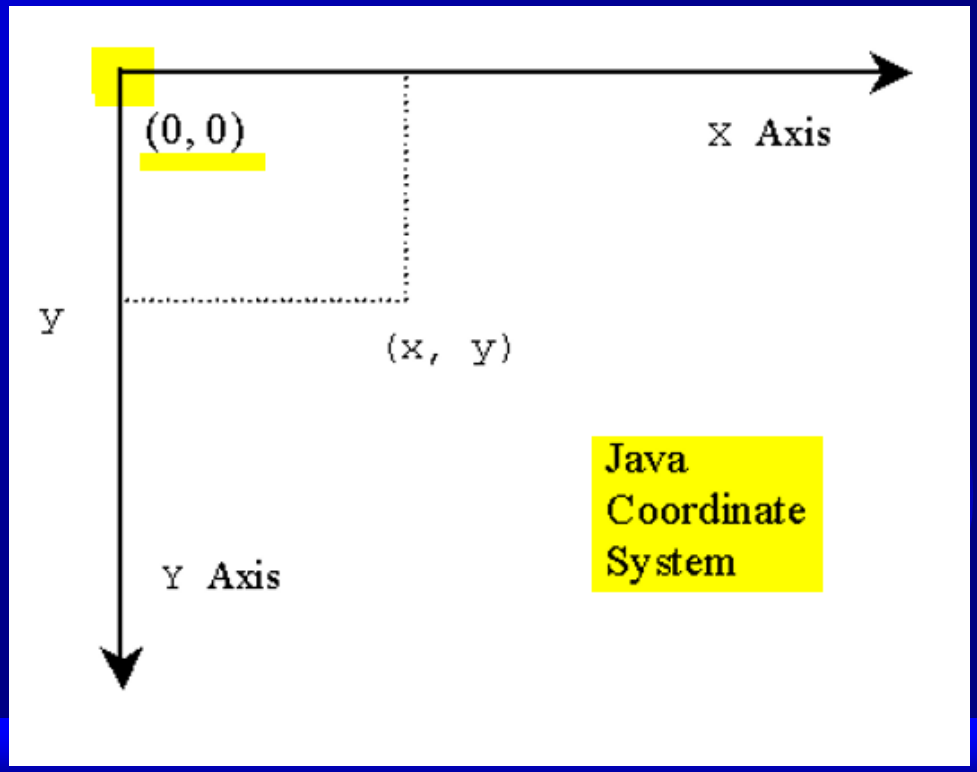


Shape

displays a circle (Shape) in the center of the pane.



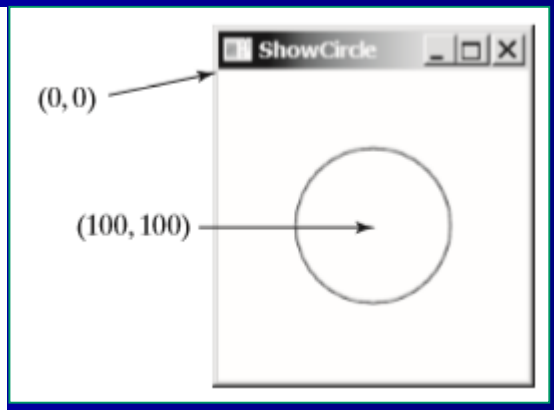
Origin:
at the top-left corner



ShowCircle.java

Displays a circle (Shape) in the center of pane.

```
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();
13     circle.setCenterX(100);
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();
21     pane.getChildren().add(circle);
22
23     // Create a scene and place it in the stage
24     Scene scene = new Scene(pane, 200, 200);
25     primaryStage.setTitle("ShowCircle"); // Set the stage title
26     primaryStage.setScene(scene); // Place the scene in the stage
27     primaryStage.show(); // Display the stage
28   }
29 }
```



create a circle
set circle properties

create a pane
add circle to pane

add pane to scene

display stage

main method omitted

the circle is not centered **after window resized**

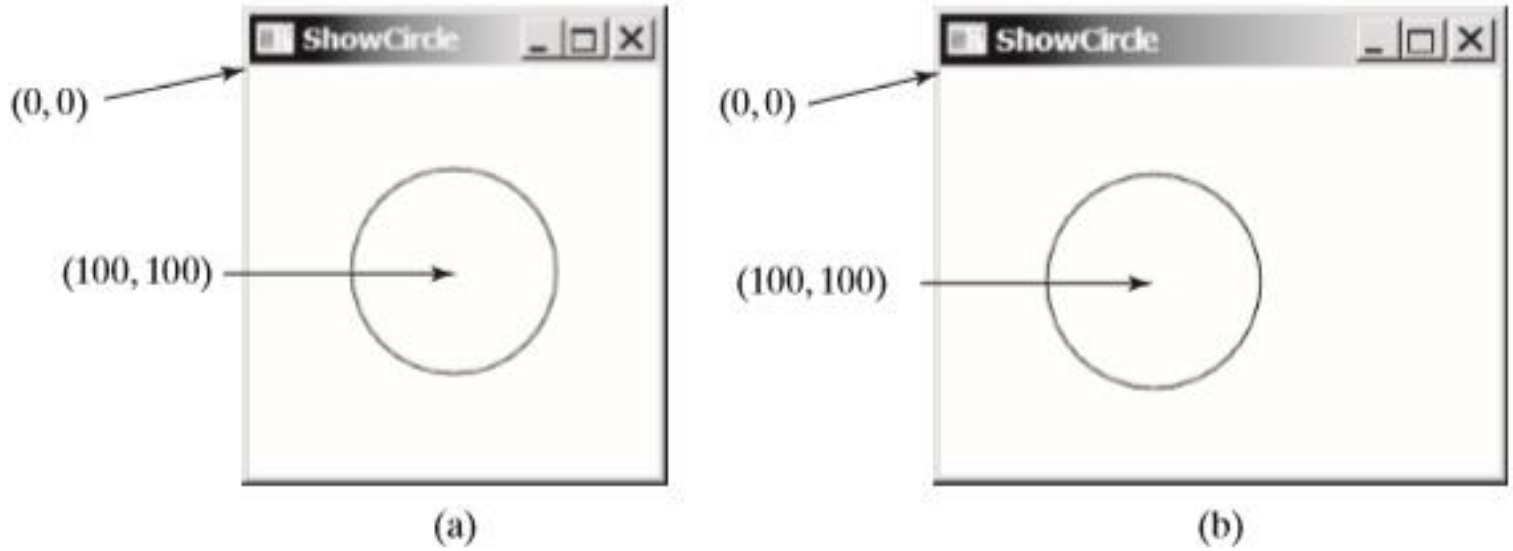


FIGURE 14.5 (a) A circle is displayed in the center of the scene. (b) The circle is not centered after the window is resized.

- In order to display the circle centered as the window resizes, the x- and y-coordinates of the circle center need to be reset to the center of the pane.
  - by binding the centerX with pane's width/2
    centerY with pane's height/2

Binding Properties

```
target.bind(source);
```

a target object listens to the changes in the source object and automatically updates itself

- ***binding property*** ***(target object /binding object)***
  - an object that can be bound to a source object
  - Using bind() method
- ***observable object*** ***(source object /bindable object)***

ShowCircleCentered.java

```
16      circle.centerXProperty().bind(pane.widthProperty().divide(2));
17      circle.centerYProperty().bind(pane.heightProperty().divide(2));
```

Define a binding property

define a binding property centerX of the DoubleProperty class/type;
centerXProperty() return an instance of the DoubleProperty class/type;

- **DoubleProperty class**
  - contain methods:
    add(), subtract(),multiply(), divide()
  - subClass of ObservableValue
    - More numeric property class
      FloatProperty, LongProperty,
      IntegerProperty, BooleanProperty
    - StringProperty

- centerXProperty()
  - Property getter
  - returns an instance of DoubleProperty class

```
public class Circle {

  private DoubleProperty centerX;

  /** Value getter method */
  public double getCenterX() { ... }

  /** Value setter method */
  public void setCenterX(double value) { ... }

  /** Property getter method */
  public DoubleProperty centerXProperty() { ... }
}
```

(b) centerX is binding property in the Circle class

Use a binding property

DoubelProperty centerX;

```
public class Circle {

  private DoubleProperty centerX;

  /** Value getter method */
  public double getCenterX() { ... }

  /** Value setter method */
  public void setCenterX(double value) { ... }

  /** Property getter method */
  public DoubleProperty centerXProperty() { ... }
}
```

(b) centerX is binding property in the Circle class

ShowCircleCentered.java

```
circle.centerXProperty().bind(pane.widthProperty().divide(2));
```

is the same as

```
DoubleProperty centerX = circle.centerXProperty();
DoubleProperty width = pane.widthProperty();
centerX.bind(width.divide(2));
```

Another example

BindingDemo.java

```
1  import javafx.beans.property.DoubleProperty;
2  import javafx.beans.property.SimpleDoubleProperty;
3
4  public class BindingDemo {
5    public static void main(String[] args) {
6      DoubleProperty d1 = new SimpleDoubleProperty(1);
7      DoubleProperty d2 = new SimpleDoubleProperty(2);
8      d1.bind(d2);
9      System.out.println("d1 is " + d1.getValue()
10       + " and d2 is " + d2.getValue());
11     d2.setValue(70.2);
12     System.out.println("d1 is " + d1.getValue()
13       + " and d2 is " + d2.getValue());
14   }
15 }
```

```
d1 is 2.0 and d2 is 2.0
d1 is 70.2 and d2 is 70.2
```

(line 6). Note that `DoubleProperty`, `FloatProperty`, `LongProperty`, `IntegerProperty`, and `BooleanProperty` are abstract classes. Their concrete subclasses `SimpleDoubleProperty`, `SimpleFloatProperty`, `SimpleLongProperty`, `SimpleIntegerProperty`, and `SimpleBooleanProperty` are used to create instances of these properties. These classes are very much like wrapper classes `Double`, `Float`, `Long`,

Common Properties for Nodes

eg. sets two properties for a circle

```
circle.setStroke(Color.BLACK);
circle.setFill(Color.RED);
```

the same as:

```
circle.setStyle("-fx-stroke: black; -fx-fill: red;");
```

styleName:value;

- **Style:**
  - set node's style property with prefix -fx-
  - docs.oracle.com/javafx/2/api/javafx/scene/docfiles/cssref.html

Common Properties for Nodes

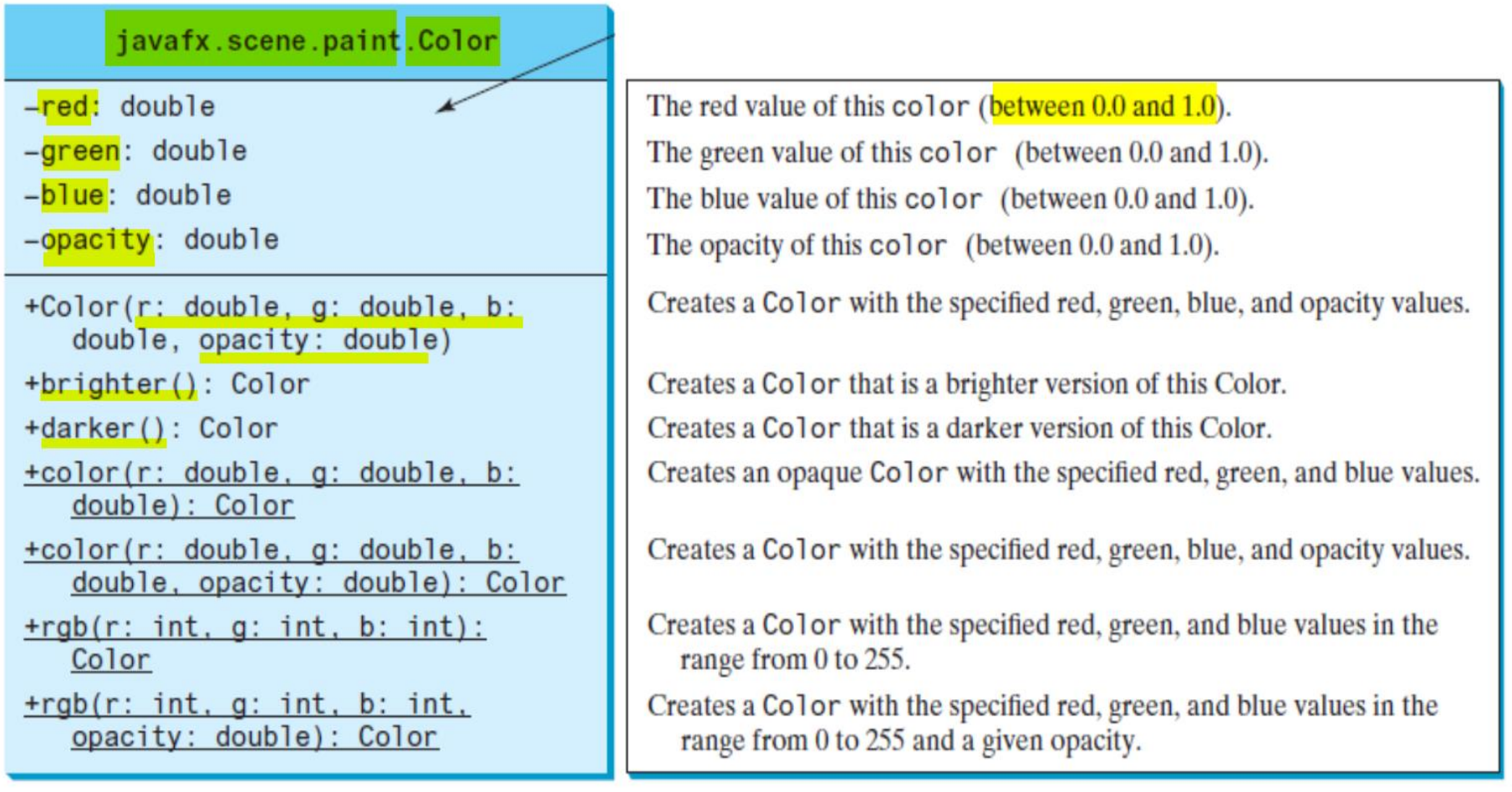
- **Rotate :**
  - rotate a node with an angle in degrees

eg. rotates a button 80 degrees:

```
button.setRotate(80);
```

The Color Class

0.0 Black (darkest) ~ 1.0 White(lightest)




```
Color color = new Color(0.25, 0.14, 0.333, 0.51);
```

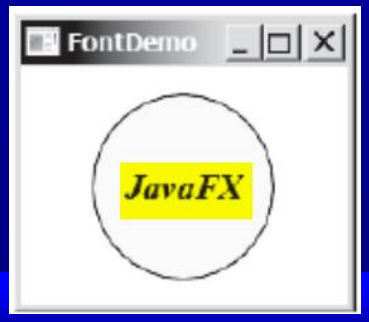
The `Color` Class

- standard colors, constants:
  - BEIGE, BLACK, BLUE, BROWN, CYAN, DARKGRAY, GOLD, GRAY, GREEN, LIGHTGRAY, MAGENTA, NAVY, ORANGE, PINK, RED, SILVER, WHITE, YELLOW
  - Example:

```
circle.setFill(Color.RED);
```

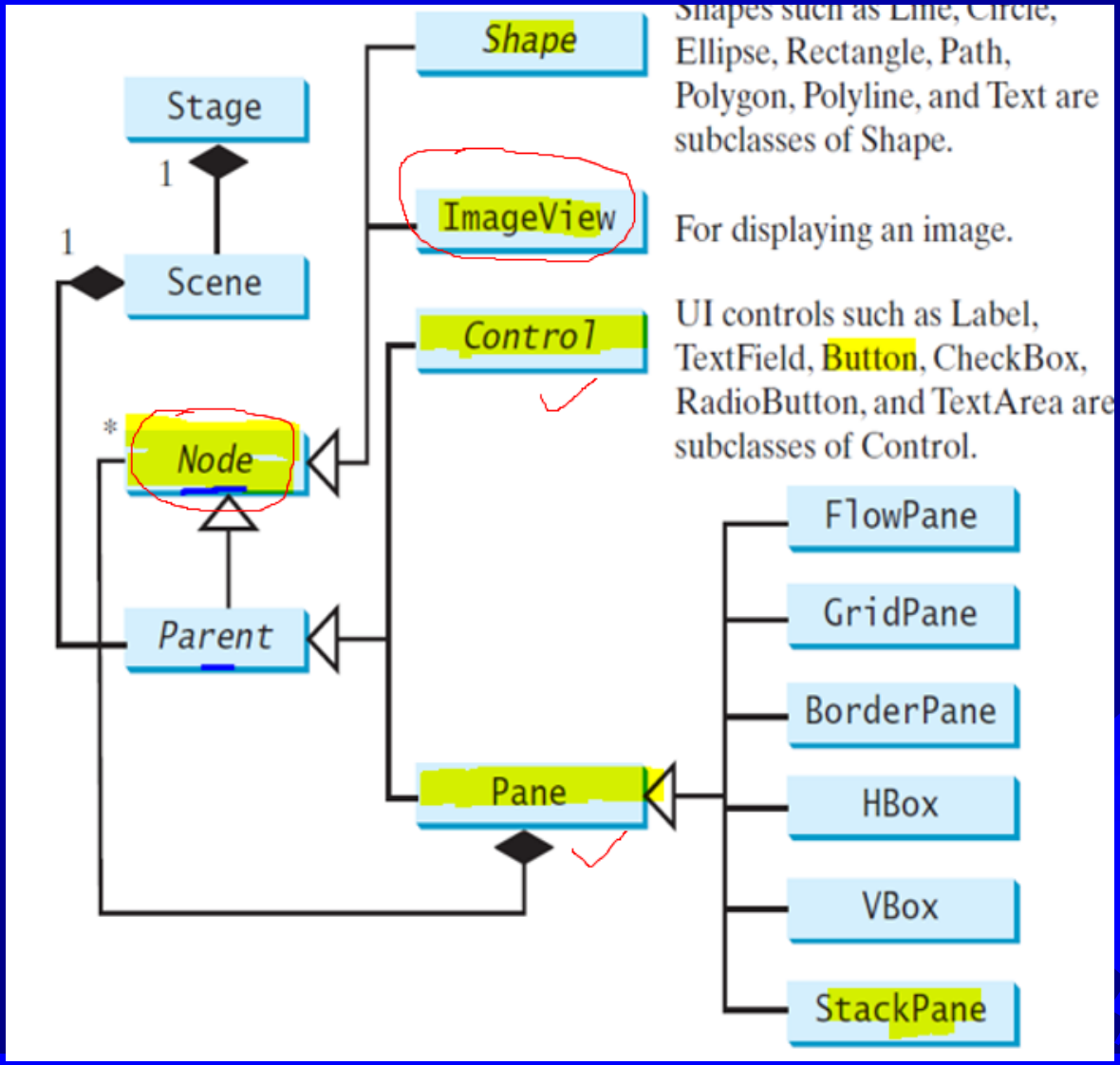
The Font Class

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.




```
23      // Create a label and set its properties
24      Label label = new Label("JavaFX");
25      label.setFont(Font.font("Times New Roman",
26        FontWeight.BOLD, FontPosture.ITALIC, 20));
27      pane.getChildren().add(label);
```

ImageView


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

The Image and ImageView Class

```
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
```

- **Image**: load an image from a filename/URL

```
new Image("image/us.gif");
new Image("http://liveexample.pearsoncmg.com/book/image/us.gif");
```

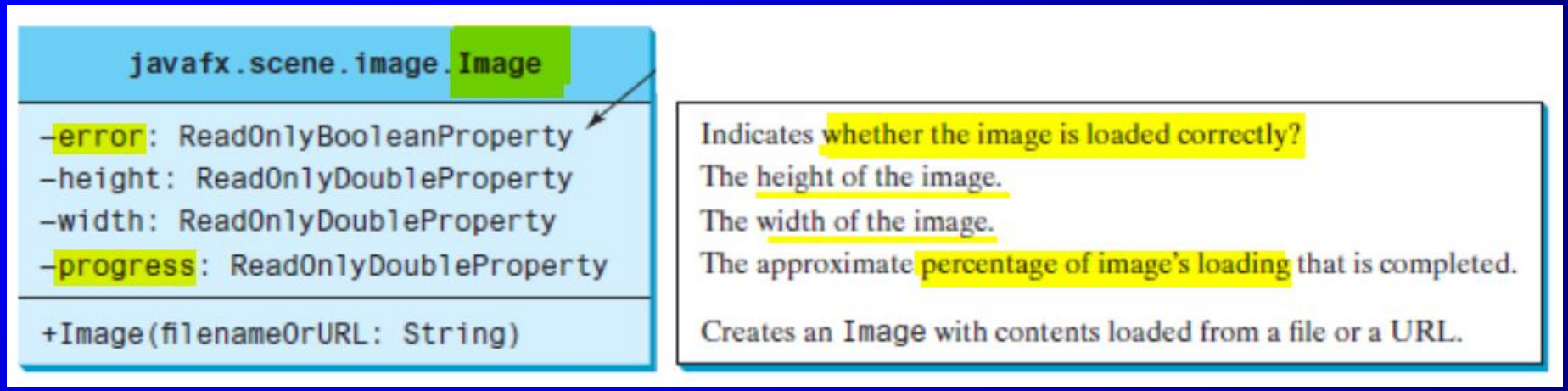
- **ImageView**: display an image

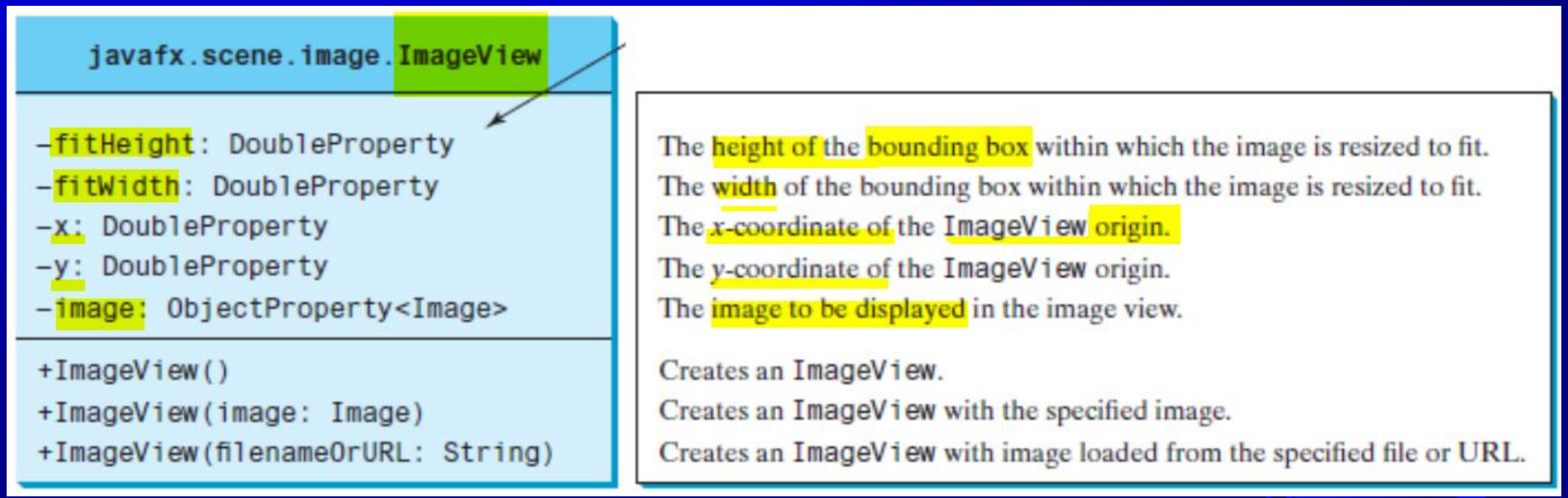
```
Image image = new Image("image/us.gif");
ImageView imageView = new ImageView (image);
```

or:

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

The Image and ImageView Class

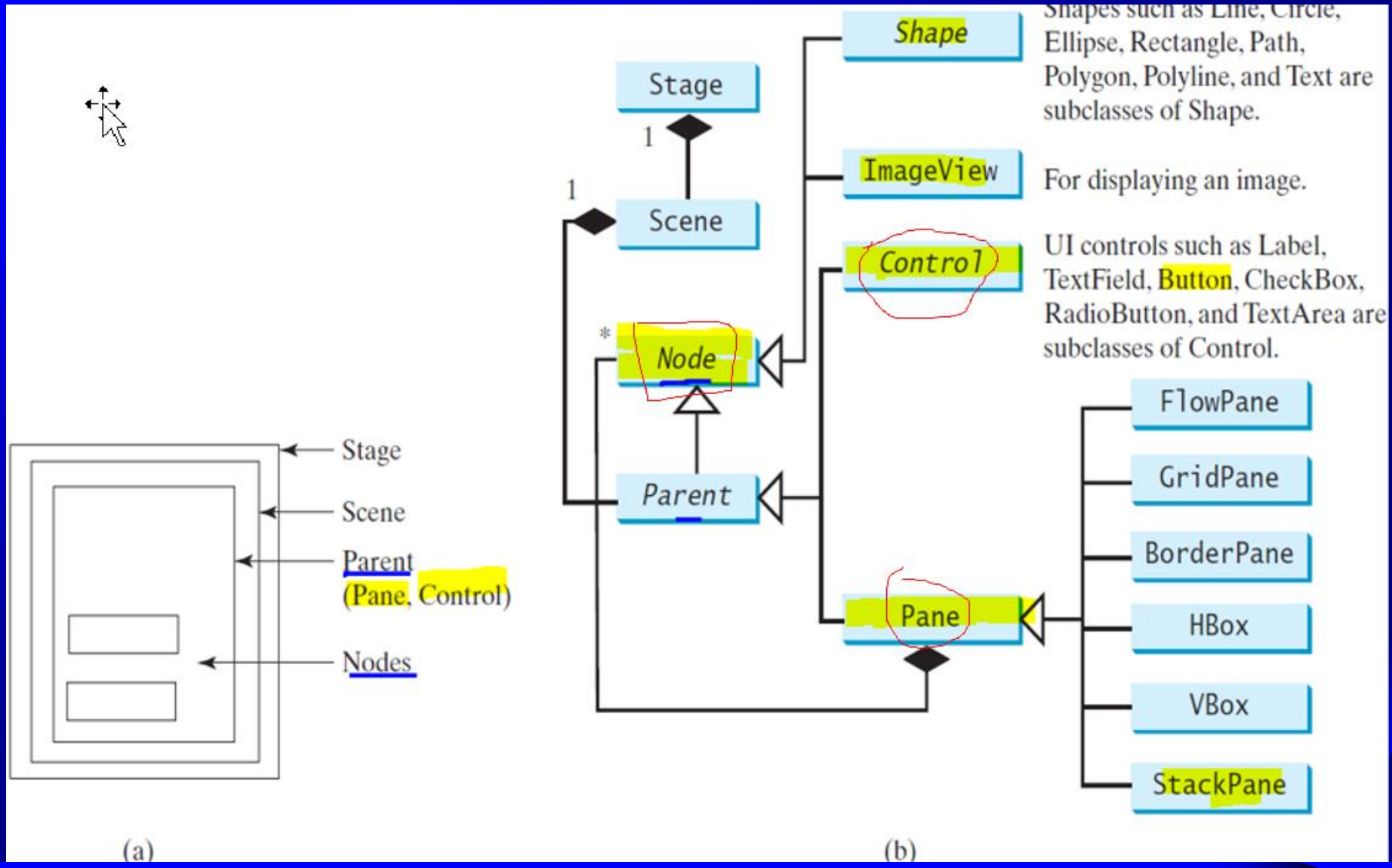




ShowImage.java


```
13      // Create a pane to hold the image views
14      Pane pane = new HBox(10);                                  create an HBox
15      pane.setPadding(new Insets(5, 5, 5, 5));
16      Image image = new Image("image/us.gif");                   create an image
17      pane.getChildren().add(new ImageView(image));              add an image view to pane
18
19      ImageView imageView2 = new ImageView(image);               create an image view
20      imageView2.setFitHeight(100);                              set image view properties
21      imageView2.setFitWidth(100);
22      pane.getChildren().add(imageView2);                        add an image to pane
23
24      ImageView imageView3 = new ImageView(image);               create an image view
25      imageView3.setRotate(90);                                  rotate an image view
26      pane.getChildren().add(imageView3);                        add an image to pane
```

subclasses of Node



Layout Panes

- Pane
  - container of nodes
  - organize the nodes' layout

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.

FlowPane

FIGURE 14.16 The nodes fill in the rows in the `FlowPane` one after another.

```
12      // Create a pane and set its properties
13      FlowPane pane = new FlowPane();
14      pane.setPadding(new Insets(11, 12, 13, 14));
15      pane.setHgap(5);
16      pane.setVgap(5);
```

https://openjfx.io/javadoc/20/javafx.graphics/javafx/scene/layout/Pane.html

```
18      // Place nodes in the pane
19      pane.getChildren().addAll(new Label("First Name:"),
20        new TextField(), new Label("MI:"));
21      TextField tfMi = new TextField();
22      tfMi.setPrefColumnCount(1);
23      pane.getChildren().addAll(tfMi, new Label("Last Name:"),
24        new TextField());
```

https://openjfx.io/javadoc/20/javafx.controls/javafx/scene/control/TextArea.html#setPrefColumnCount(int)

ShowFlowPane

FlowPane

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.layout.FlowPane

```
-alignment: ObjectProperty<Pos>
-orientation:
   ObjectProperty<Orientation>
-hgap: DoubleProperty
-vgap: DoubleProperty
```

```
+FlowPane()
+FlowPane(hgap: double, vgap:
   double)
+FlowPane(orientation:
   ObjectProperty<Orientation>)
+FlowPane(orientation:
   ObjectProperty<Orientation>,
   hgap: double, vgap: double
```

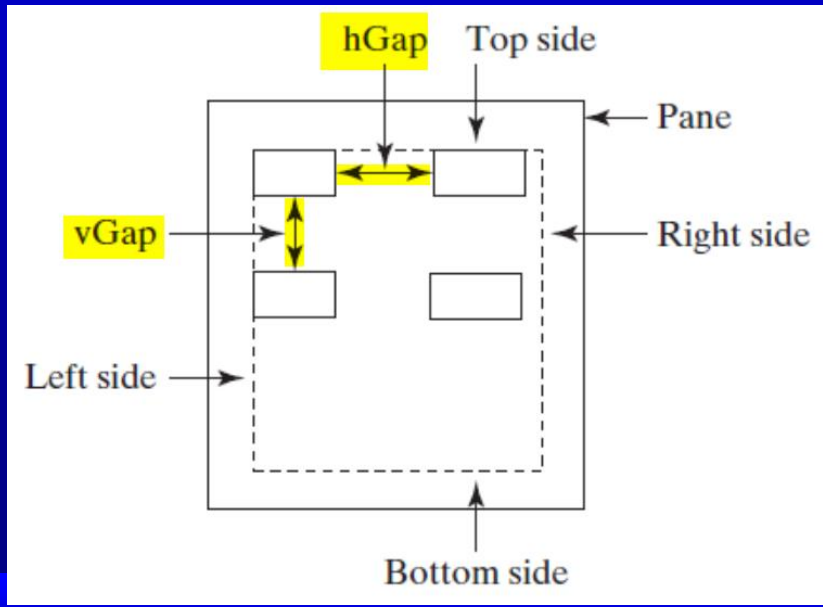
The overall alignment of the content in this pane (default: `Pos.LEFT`).
The orientation in this pane (default: `Orientation.HORIZONTAL`).

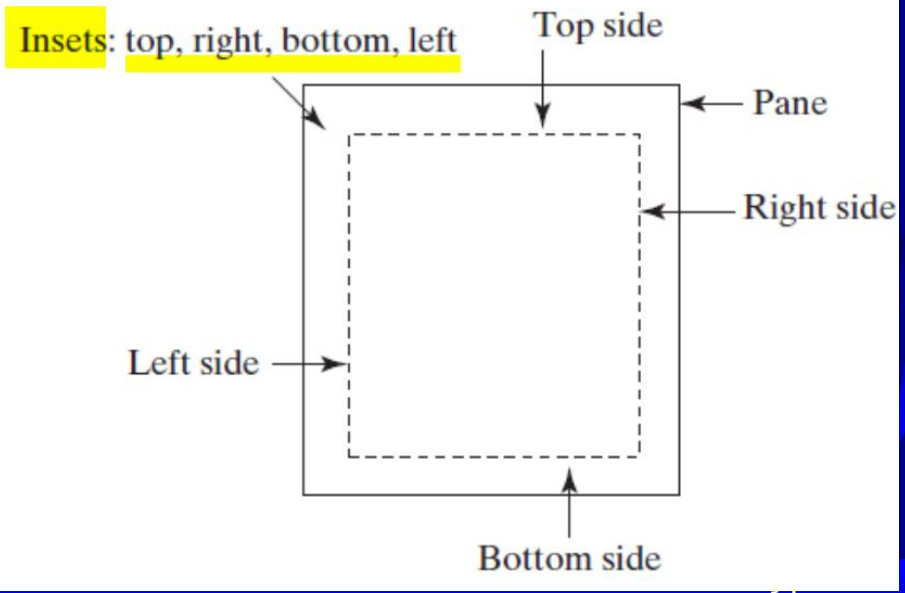
The horizontal gap between the nodes (default: `0`).
The vertical gap between the nodes (default: `0`).

Creates a default `FlowPane`.
Creates a `FlowPane` with a specified horizontal and vertical gap.

Creates a `FlowPane` with a specified orientation.

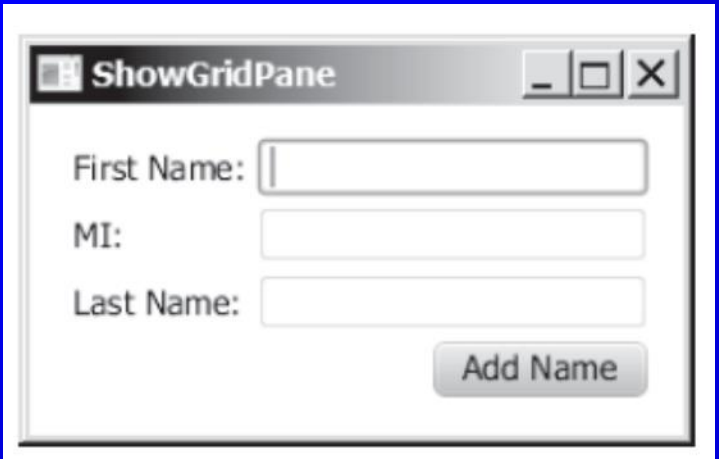
Creates a `FlowPane` with a specified orientation, horizontal gap and vertical gap.





ShowGridPane

GridPane

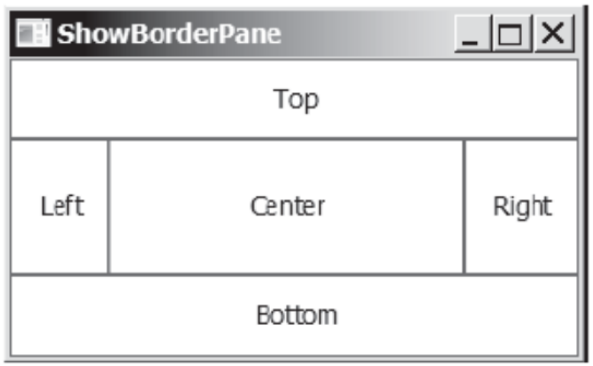


	[0]	[1]
[0]	(0, 0)	(0, 1)
[1]	(1, 0)	(1, 1)
[2]	(2, 0)	(2, 1)
[3]	(3, 0)	(3, 1)

```
15      // Create a pane and set its properties
16      GridPane pane = new GridPane();
17      pane.setAlignment(Pos.CENTER);
18      pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
19      pane.setHgap(5.5);
20      pane.setVgap(5.5);
21
22      // Place nodes in the pane
23      pane.add(new Label("First Name:"), 0, 0);
24      pane.add(new TextField(), 1, 0);
25      pane.add(new Label("MI:"), 0, 1);
26      pane.add(new TextField(), 1, 1);
27      pane.add(new Label("Last Name:"), 0, 2);
28      pane.add(new TextField(), 1, 2);
29      Button btAdd = new Button("Add Name");
30      pane.add(btAdd, 1, 3);
31      GridPane.setHalignment(btAdd, HPos.RIGHT);
```

ShowBorderPane

BorderPane

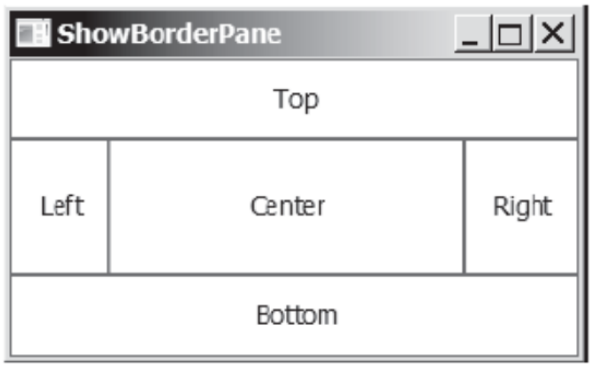


The BorderPane places the nodes in five regions of the pane.

```
12      // Create a border pane
13      BorderPane pane = new BorderPane();
14
15      // Place nodes in the pane
16      pane.setTop(new CustomPane("Top"));
17      pane.setRight(new CustomPane("Right"));
18      pane.setBottom(new CustomPane("Bottom"));
19      pane.setLeft(new CustomPane("Left"));
20      pane.setCenter(new CustomPane("Center"));
```

ShowBorderPane

BorderPane



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

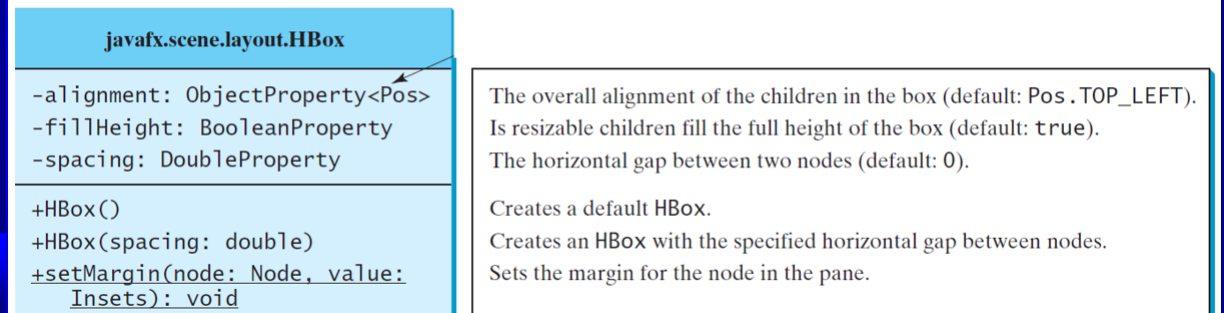
```
30  // Define a custom pane to hold a label in the center of the pane
31  class CustomPane extends StackPane {
32    public CustomPane(String title) {
33      getChildren().add(new Label(title));
34      setStyle("-fx-border-color: red");
35      setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
36    }
```

ShowHBoxVBox

HBox

The HBox places the nodes in one row

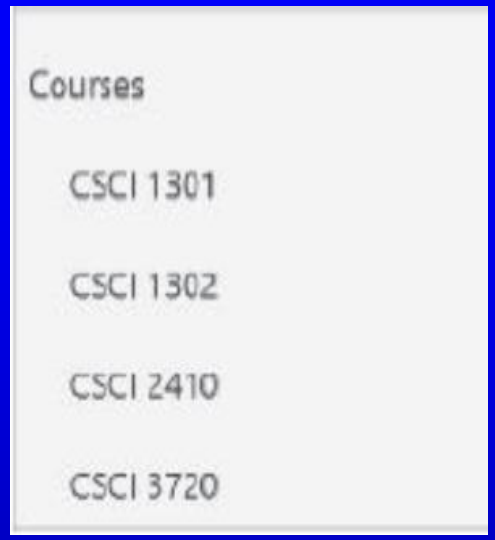

```
31      HBox hBox = new HBox(15);
32      hBox.setPadding(new Insets(15, 15, 15, 15));
33      hBox.setStyle("-fx-background-color: gold");
34      hBox.getChildren().add(new Button("Computer Science"));
35      hBox.getChildren().add(new Button("Chemistry"));
36      ImageView imageView = new ImageView(new Image("image/us.gif"));
37      hBox.getChildren().add(imageView);
```



ShowHBoxVBox

VBox

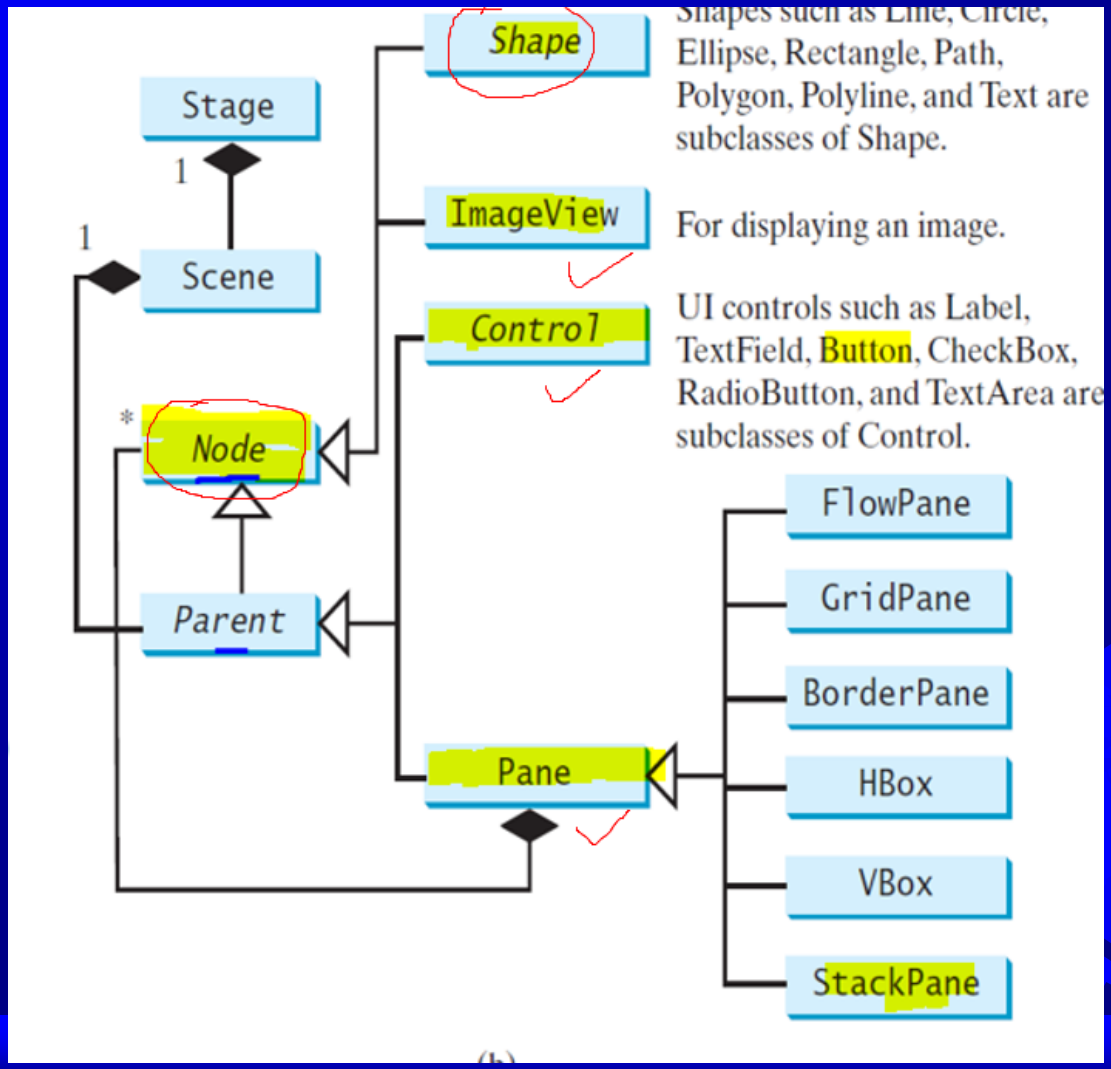
The VBox places the nodes in one column




```
42      VBox vBox = new VBox(15);
43      vBox.setPadding(new Insets(15, 5, 5, 5));
44      vBox.getChildren().add(new Label("Courses"));
45
46      Label[] courses = {new Label("CSCI 1301"), new Label("CSCI 1302"),
47          new Label("CSCI 2410"), new Label("CSCI 3720")};
48
49      for (Label course: courses) {
50        VBox.setMargin(course, new Insets(0, 0, 0, 15));
51        vBox.getChildren().add(course);
52      }
```

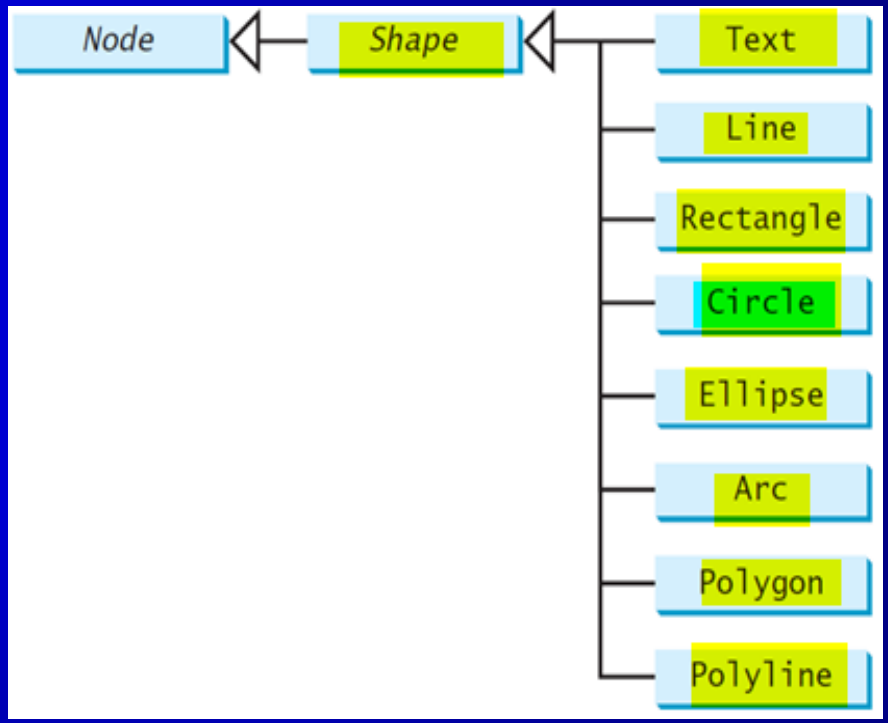
Shapes

Shape classes： for drawing various shapes.



Shapes

- Shape classes :
  - for drawing text, ……



- Shape attributes:
  - fill : color that fills the shape.
  - stroke : color that draws the outline of the shape
  - strokeWidth : width of the outline of the shape.

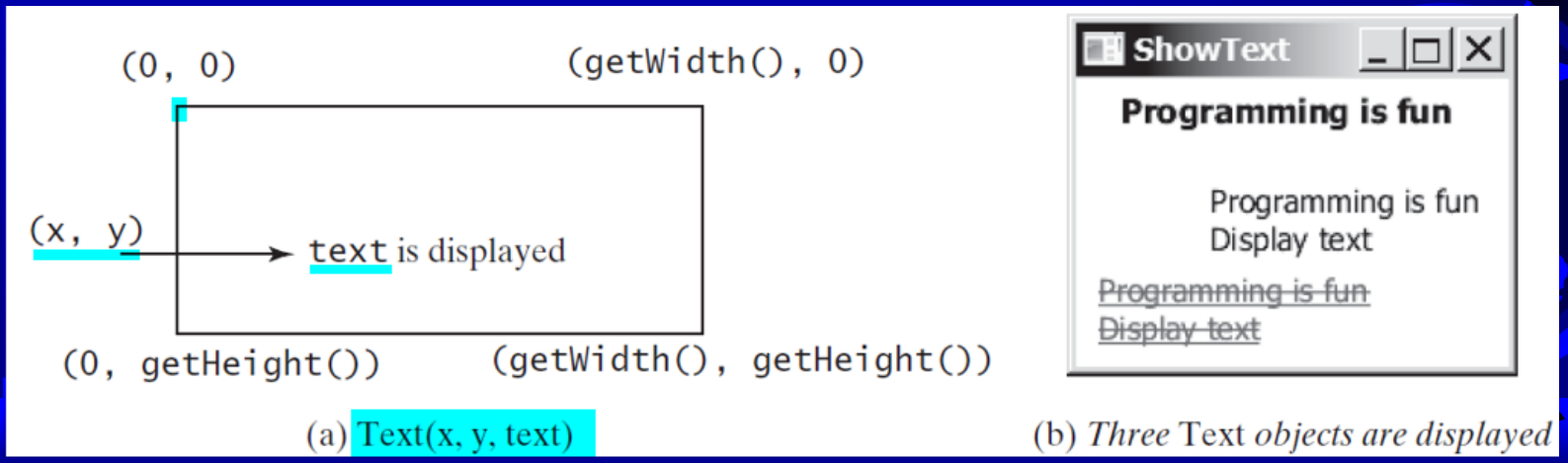
Text

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.

A Text object is usually placed in a pane.

- upper-left point is (0, 0) ;
- bottom-right point is (pane.getWidth(), pane.getHeight()).



(a) Text(x, y, text)

(b) *Three* Text *objects are displayed*

ShowText.java

```
15      // Create a pane to hold the texts
16      Pane pane = new Pane();
17      pane.setPadding(new Insets(5, 5, 5, 5));
18      Text text1 = new Text(20, 20, "Programming is fun");
19      text1.setFont(Font.font("Courier", FontWeight.BOLD,
20        FontPosture.ITALIC, 15));
21      pane.getChildren().add(text1);
```

```
26      Text text3 = new Text(10, 100, "Programming is fun\nDisplay text");
27      text3.setFill(Color.RED);
28      text3.setUnderline(true);
29      text3.setStrikethrough(true);
30      pane.getChildren().add(text3);
```

Line

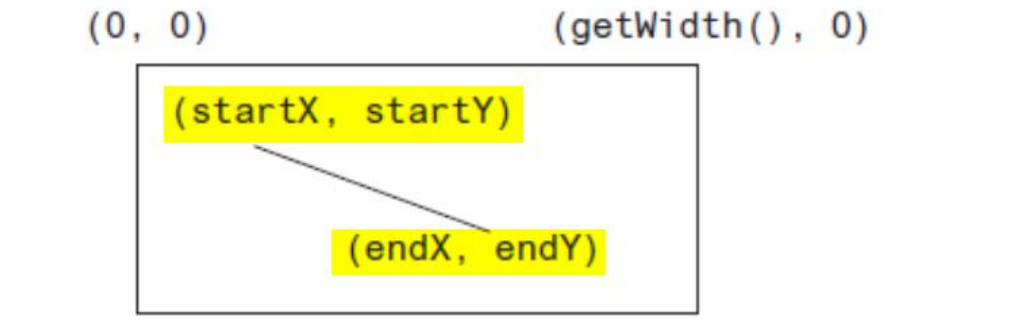
javafx.scene.shape.Line

```
-startX: DoubleProperty
-startY: DoubleProperty
-endX: DoubleProperty
-endY: DoubleProperty

+Line()
+Line(startX: double, startY:
   double, endX: double, endY:
   double)
```

The x-coordinate of the start point.
The y-coordinate of the start point.
The x-coordinate of the end point.
The y-coordinate of the end point.

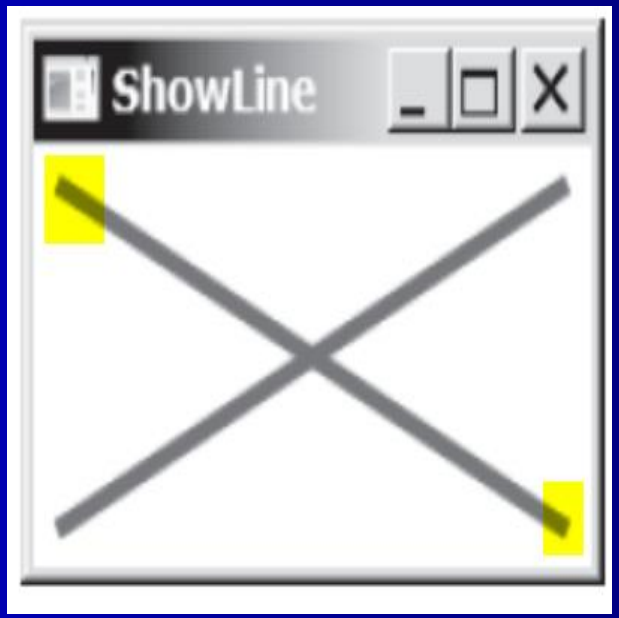
Creates an empty `Line`.
Creates a `Line` with the specified starting and ending points.



(a) `Line(startX, startY, endX, endY)`

ShowLine.java

```
21      Line line1 = new Line(10, 10, 10, 10);
22      line1.endXProperty().bind(widthProperty().subtract(10));
23      line1.endYProperty().bind(heightProperty().subtract(10));
24      line1.setStrokeWidth(5);
25      line1.setStroke(Color.GREEN);
26      getChildren().add(line1);
```



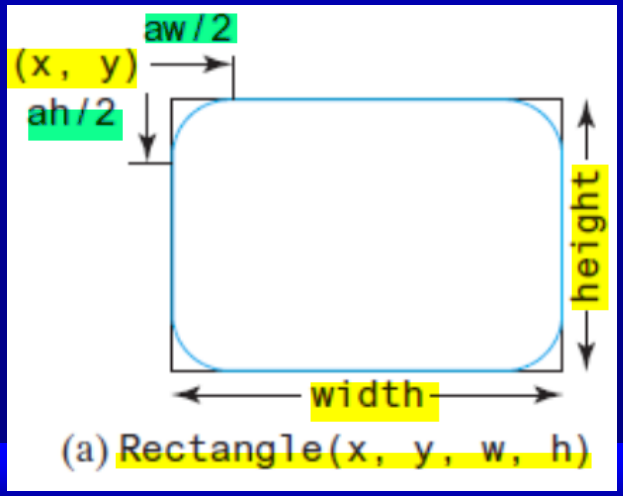
Rectangle

javafx.scene.shape.Rectangle

- -x: DoubleProperty
- -y: DoubleProperty
- -width: DoubleProperty
- -height: DoubleProperty
- -arcWidth: DoubleProperty
- -arcHeight: DoubleProperty

- +Rectangle()
- +Rectangle(x: double, y: double, width: double, height: double)

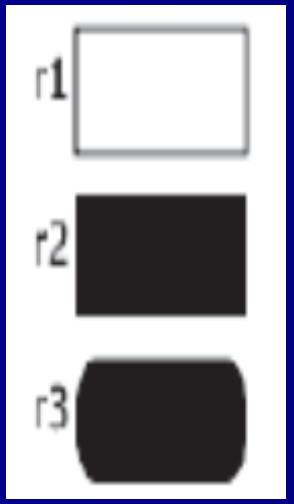
-x	The *x*-coordinate of the upper-left corner of the rectangle (default 0).
-y	The *y*-coordinate of the upper-left corner of the rectangle (default 0).
-width	The width of the rectangle (default: 0).
-height	The height of the rectangle (default: 0).
-arcWidth	The arcWidth of the rectangle (default: 0). arcWidth is the horizontal diameter of the arcs at the corner (see Figure 14.31a).
-arcHeight	The arcWidth 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.
+Rectangle(x: double, y: double, width: double, height: double)	Creates a Rectangle with the specified upper-left corner point, width, and height.



(a) Rectangle(x, y, w, h)

ShowRectangle.java

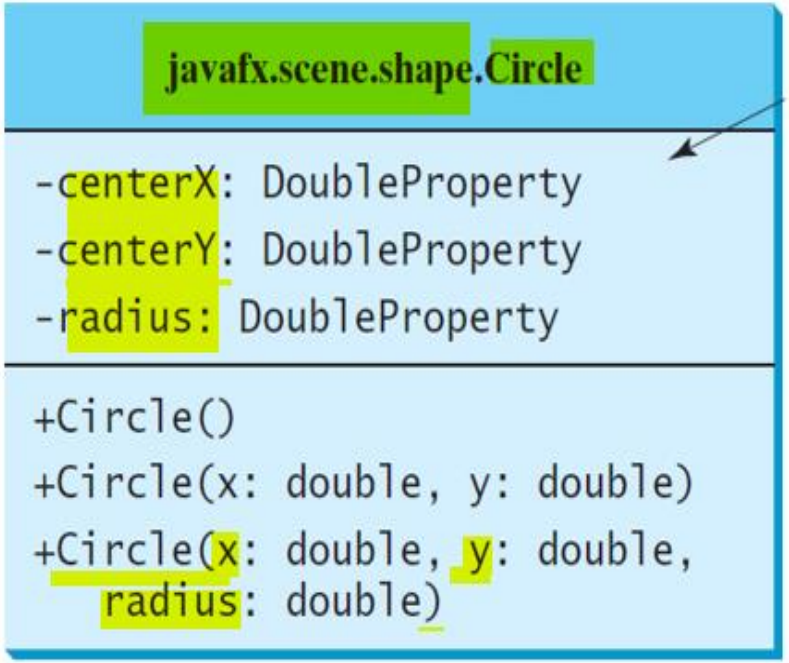
```
14      Rectangle r1 = new Rectangle(25, 10, 60, 30);
15      r1.setStroke(Color.BLACK);
16      r1.setFill(Color.WHITE);
17      Rectangle r2 = new Rectangle(25, 50, 60, 30);
18      Rectangle r3 = new Rectangle(25, 90, 60, 30);
19      r3.setArcWidth(15);
20      r3.setArcHeight(25);
```



- By default r2, r3 :
  - the fill color is black.
  - the stroke color is white

```
22      // Create a group and add nodes to the group
23      Group group = new Group();
24      group.getChildren().addAll(new Text(10, 27, "r1"), r1,
25        new Text(10, 67, "r2"), r2, new Text(10, 107, "r3"), r3);
```

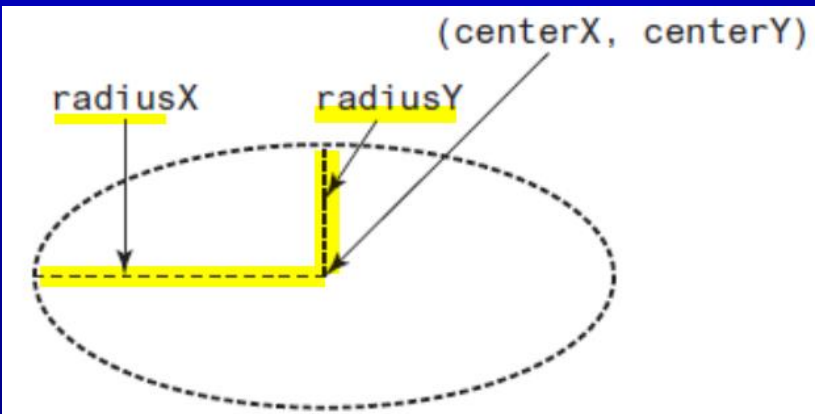
Circle



-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

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.



(a) Ellipse(centerX, centerY, radiusX, radiusY)

ShowEllipse.java

```
 8  public class ShowEllipse extends Application {
 9    @Override // Override the start method in the Application class
10    public void start(Stage primaryStage) {
11      // Create a scene and place it in the stage
12      Scene scene = new Scene(new MyEllipse(), 300, 200);
13      primaryStage.setTitle("ShowEllipse"); // Set the stage title
14      primaryStage.setScene(scene); // Place the scene in the stage
15      primaryStage.show(); // Display the stage
16    }
17  }
18
19  class MyEllipse extends Pane {
20    private void paint() {
21      getChildren().clear();
22      for (int i = 0; i < 16; i++) {
23        // Create an ellipse and add it to pane
24        Ellipse e1 = new Ellipse(getWidth() / 2, getHeight() / 2,
25          getWidth() / 2 - 50, getHeight() / 2 - 50);
26        e1.setStroke(Color.color(Math.random(), Math.random(),
27          Math.random()));
28        e1.setFill(Color.WHITE);
29        e1.setRotate(i * 180 / 16);
30        getChildren().add(e1);
31      }
32    }
```

create a pane

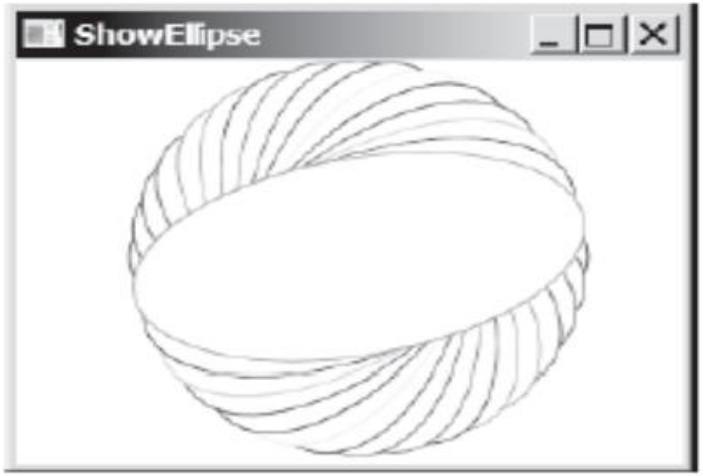
create an ellipse

set random color for stroke

set fill color

rotate ellipse

add ellipse to pane



(b) Multiple ellipses are displayed.

Arc

javafx.scene.shape.Arc

```
-centerX: DoubleProperty
-centerY: DoubleProperty
-radiusX: DoubleProperty
-radiusY: DoubleProperty
-startAngle: DoubleProperty
-length: DoubleProperty
-type: ObjectProperty<ArcType>

+Arc()
+Arc(x: double, y: double,
   radiusX: double, radiusY:
   double, startAngle: double,
   length: double)
```

The x-coordinate of the center of the ellipse (default 0).

The y-coordinate of the center of the ellipse (default 0).

The horizontal radius of the ellipse (default: 0).

The vertical radius of the ellipse (default: 0).

The start angle of the arc in degrees.

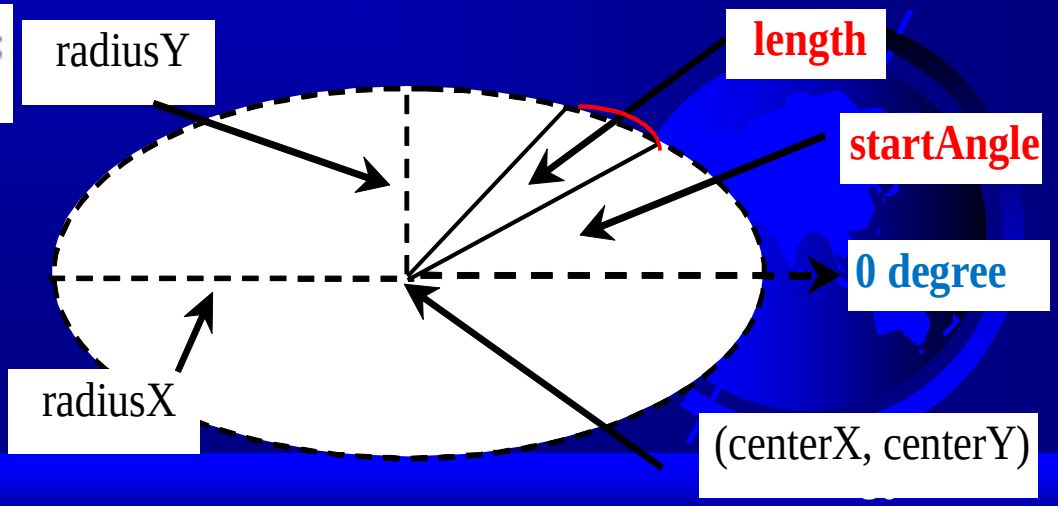
The angular extent of the arc in degrees.

The closure type of the arc (`ArcType.OPEN`, `ArcType.CHORD`, `ArcType.ROUND`).

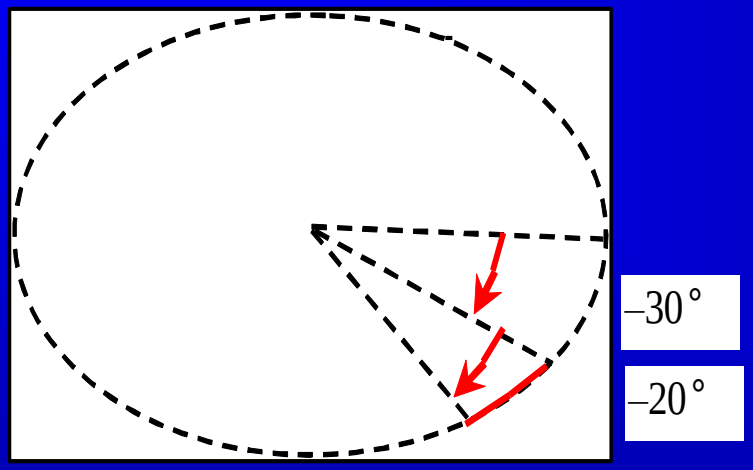
Creates an empty `Arc`.

Creates an `Arc` with the specified arguments.

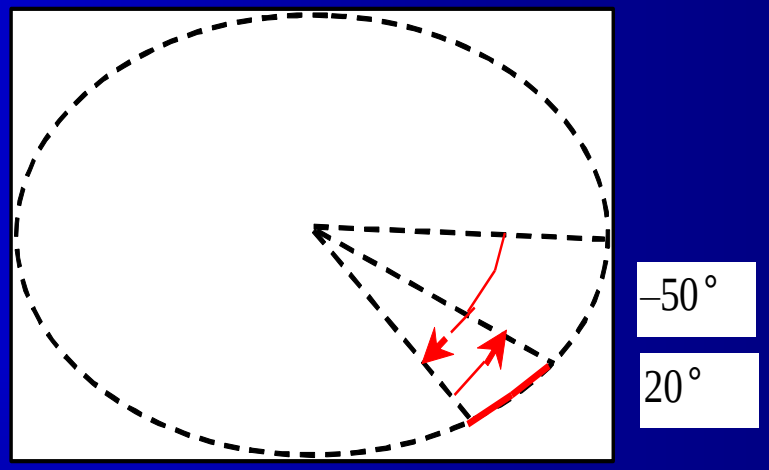
```
new Arc(x, y, radiusX, radiusY, -30, -20);
new Arc(x, y, radiusX, radiusY, -50, 20);
```



```
new Arc(x, y, radiusX, radiusY, -30, -20);
new Arc(x, y, radiusX, radiusY, -50, 20);
```



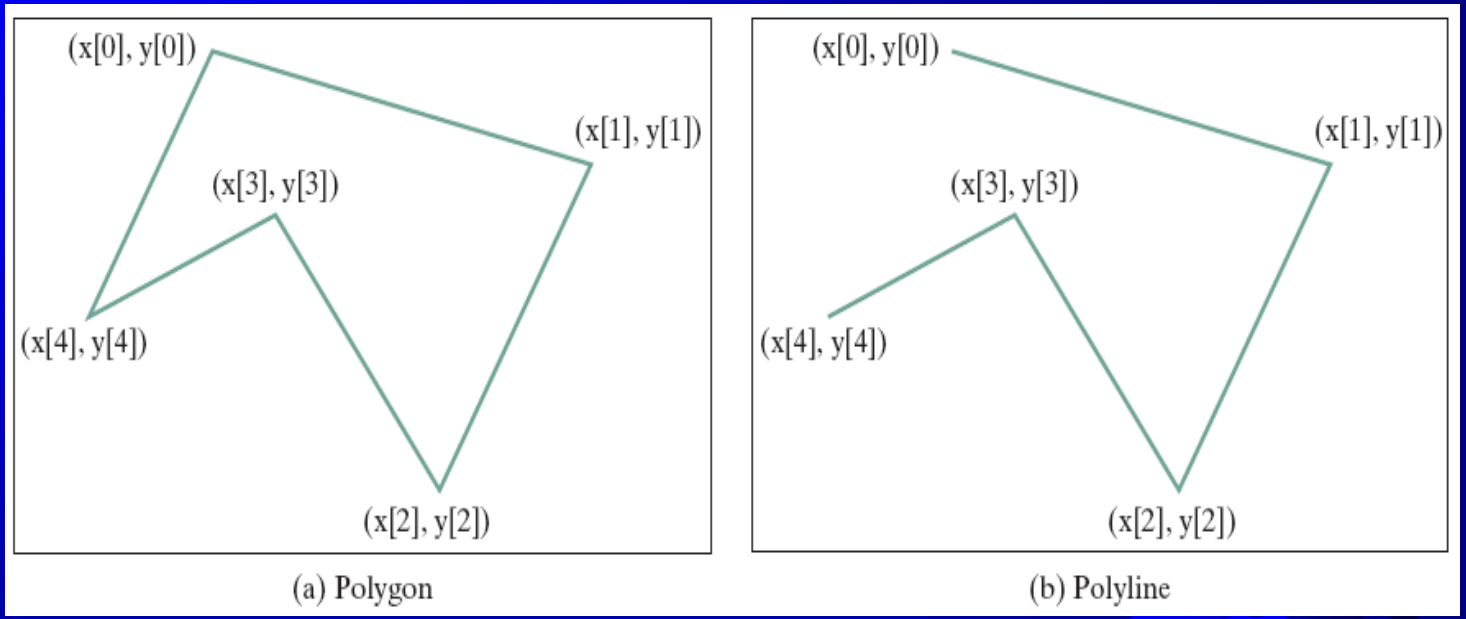
(a) Negative starting angle –30 ° and negative spanning angle –20 °



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

Polygon and Polyline

- connecting a sequence of points
  - Polygon: closed
  - Polyline: not closed



javafx.scene.shape.Polygon

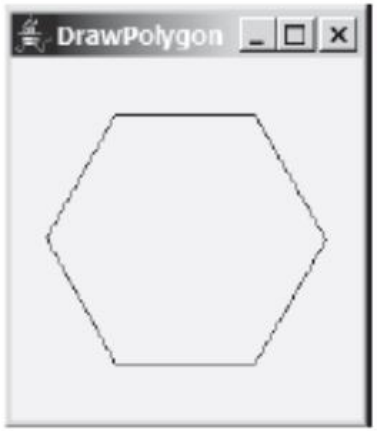
```
+Polygon()
+Polygon(double... points)
+getPoints():
   ObservableList<Double>
```

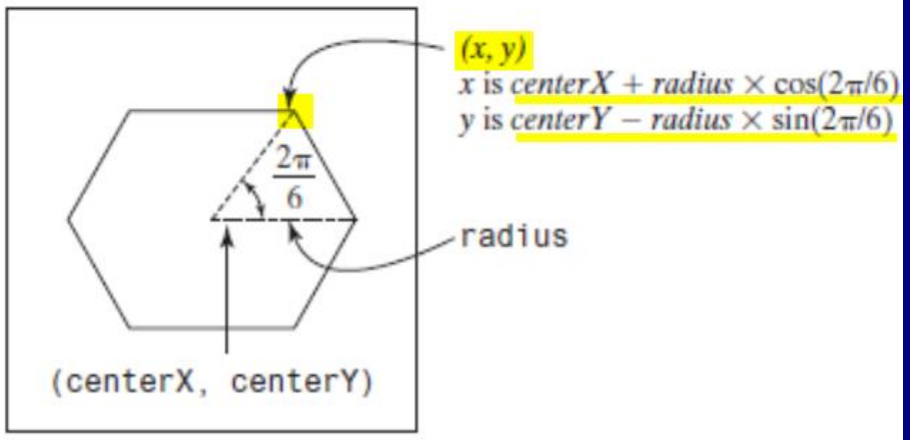
Creates an empty Polygon.

Creates a Polygon with the given points.

Returns a list of double values as *x*- and *y*-coordinates of the points.

ShowPolygon.java





(a)

```
22      // Create a polygon
23      Polygon polygon = new Polygon();
24      polygon.setFill(Color.WHITE);
25      polygon.setStroke(Color.BLACK);
26      ObservableList<Double> list = polygon.getPoints();
27
28      double centerX = getWidth() / 2, centerY = getHeight() / 2;
29      double radius = Math.min(getWidth(), getHeight()) * 0.4;
30
31      // Add points to the polygon list
32      for (int i = 0; i < 6; i++) {
33        list.add(centerX + radius * Math.cos(2 * i * Math.PI / 6));
34        list.add(centerY - radius * Math.sin(2 * i * Math.PI / 6));
35      }
```

Case Study: **The** ClockPane **Class**

a class that displays a clock on a pane.

javafx.scene.layout.Pane

ClockPane

-hour: int
-minute: int
-second: int

+ClockPane()
+ClockPane(hour: int, minute: int, second: int)
+setCurrentTime(): void
+setWidth(width: double): void
+setHeight(height: double): void

The getter and setter
these data fields are provic
but omitted in the UML di

The hour in the clock.
The minute in the clock.
The second in the clock.

Constructs a default clock for the current time.
Constructs a clock with the specified time.

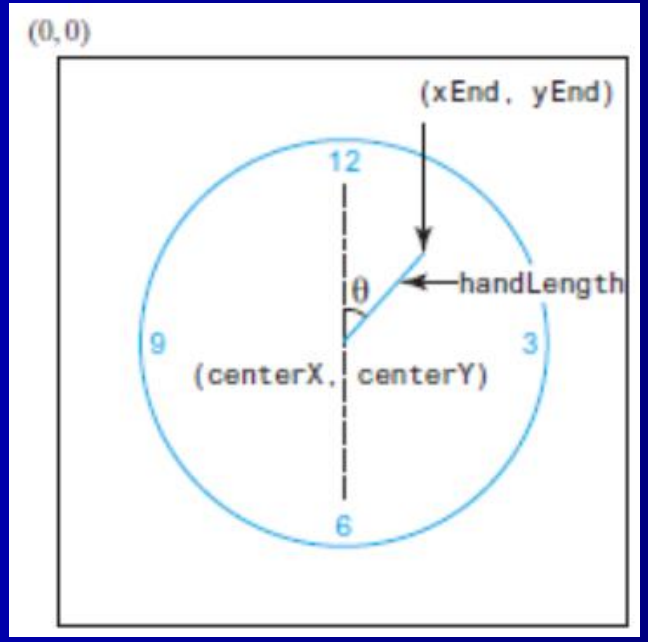
Sets hour, minute, and second for current time.
Sets clock pane's width and repaint the clock.
Sets clock pane's height and repaint the clock.

LISTING 14.21 ClockPane.java

```
1  import java.util.Calendar;
2  import java.util.GregorianCalendar;
```

```
59   /* Set the current time for the clock */
60   public void setCurrentTime() {
61     // Construct a calendar for the current date and time
62     Calendar calendar = new GregorianCalendar();
63
64     // Set current hour, minute and second
65     this.hour = calendar.get(Calendar.HOUR_OF_DAY);
66     this.minute = calendar.get(Calendar.MINUTE);
67     this.second = calendar.get(Calendar.SECOND);
68
69     paintClock(); // Repaint the clock
70   }
```


```
72    /** Paint the clock */
73    private void paintClock() {                                    paint clock
74      // Initialize clock parameters
75      double clockRadius =                                         get radius
76        Math.min(getWidth(), getHeight()) * 0.8 * 0.5;
77      double centerX = getWidth() /2;                              get center
78      double centerY = getHeight() /2;
79
80      // Draw circle
81      Circle circle = new Circle(centerX, centerY, clockRadius);   create a circle
82      circle.setFill(Color.WHITE);
83      circle.setStroke(Color.BLACK);
84      Text t1 = new Text(centerX - 5, centerY - clockRadius + 12, "12");   create texts
85      Text t2 = new Text(centerX - clockRadius + 3, centerY + 5, "9");
86      Text t3 = new Text(centerX + clockRadius - 10, centerY + 3, "3");
87      Text t4 = new Text(centerX - 3, centerY + clockRadius - 3, "6");
```




```
89      // Draw second hand
90      double sLength = clockRadius * 0.8;
91      double secondX = centerX + sLength *
92        Math.sin(second * (2 * Math.PI / 60));
93      double secondY = centerY - sLength *
94        Math.cos(second * (2 * Math.PI / 60));
95      Line sLine = new Line(centerX, centerY, secondX, secondY);
96      sLine.setStroke(Color.RED);
```

```
107       // Draw hour hand
108       double hLength = clockRadius *  0.5;
109       double hourX = centerX + hLength *
110         Math.sin((hour % 12 + minute / 60.0) * (2 * Math.PI / 12));
111       double hourY = centerY - hLength *
112         Math.cos((hour % 12 + minute / 60.0) * (2 * Math.PI / 12));
113       Line hLine = new Line(centerX, centerY, hourX, hourY);
114       hLine.setStroke(Color.GREEN);
```

Splash Screen

- ***Splash screen***: an image that is displayed while the application is starting up.
  - If your program takes a long time to load, you may display a splash screen to alert the user.

- e.g. the following command:

```
java –splash:image/us.gif TestImage
```

  - displays the image us.gif while the program TestImage is being loaded.