Announcements

- Try to get some help from me and tutors
- Reading assignment for this slide set: Chapter 14

Motivations

- JavaFX is a new framework for developing Java GUI programs.
- The JavaFX API is an excellent example of how the object-oriented principle is applied.
- Two topics: the basics of JavaFX programming; using JavaFX to demonstrate OOP.
- Specifically, we introduce the framework of JavaFX and discuss JavaFX GUI components and their relationships.

JavaFX vs. Swing and AWT

- Swing and AWT are replaced by the JavaFX platform for developing rich Internet applications.
- Early on the GUI classes were bundled in a library called Abstract Windows Toolkit (AWT), which was fine for simple GUIs.
- AWT user interface components were replaced by a more robust, versatile, and flexible library called Swing components.
  - Swing components are painted directly on canvases using Java code.
  - Swing components depend less on the target platform and use less of the native GUI resource.
- In Java 8, Swing is replaced by a new GUI platform known as JavaFX.
Basic structure of JavaFX

- `javafx.application.Application`
- Override the `start (Stage)` method
- Stage, Scene, and Nodes

Examples:
- See MyJavaFX.java
- See MultipleStageDemo.java

Display a Shape

This example displays a circle in the center of the pane.

- See ButtonInPane.java
- See ShowCircle.java

Binding properties

- JavaFX introduces a new concept called binding property that enables a target object to be bound to a source object.
  - Example: `target.bind(source)`
- If the value in the source object changes, the target property is also changed automatically.
- The target object is simply called a binding object or a binding property.

Panes, UI Controls, and Shapes

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

UI controls such as Label, TextField, Button, CheckBox, RadioButton, and TextArea are subclasses of Control.
Binding property:
getter, setter, and property getter

```java
public class SomeClass {
    private PropertyType x;
    //** Value getter method */
    public PropertyType get() { ... }
    //** Value setter method */
    public void set(PropertyType value) { ... }
    //** Property getter method */
    public PropertyType x(Property) { ... }
}
```

Uni/Bidirectional binding
- See BindingDemo.java

Common properties and methods for Nodes
- style: set a JavaFX CSS style
- rotate: Rotate a node
- See NodeStyleRotateDemo.java

The Color class
- red: double
- green: double
- blue: double
- opacity: double
- Color newColor(double r, double g, double b);
- brighter(): Color
- darker(): Color
- color(int r, int g, int b): Color
- alpha(double a): Color
- alpha(double opacity): Color
- Create a Color with the specified red, green, blue, and opacity values.
- Create a Color that is a brighter version of this Color.
- Create a Color that is a darker version of this Color.
- Create a Color with the specified red, green, and blue values.
- Create a Color with the specified red, green, blue, and opacity values.
- Create a Color with the specified red, green, and blue values in the range from 0 to 255.
- Create a Color with the specified red, green, and blue values in the range from 0 to 255 and a given opacity.
The Font class

```
javaFX.scene.text.Font
- size: double
- name: String
- family: String

+font(size: double)
+font(name: String, size: double)
+font(name: String, size: double, family: String)
+font(name: String, size: double, family: String, weight: FontWeight)
+getFamilies(): List<String>
+getFontNames(): List<String>
```

See FontDemo.java

The Image class

```
javaFX.scene.image.Image
- error: ReadOnlyBooleanProperty
- width: ReadOnlyBooleanProperty
- height: ReadOnlyBooleanProperty
- progress: ReadOnlyBooleanProperty
- image(filenameOrURL: String)
```

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

Indicates whether the image is loaded correctly.
The height of the image.
The width of the image.
The approximate percentage of image’s loading that is completed.
Creates an Image with contents loaded from a file or a URL.

The ImageView class

```
javaFX.scene.image.ImageView
- fitHeight: DoubleProperty
- fitWidth: DoubleProperty
- x: DoubleProperty
- y: DoubleProperty
- image: ObjectProperty<Image>
+imageView(): ImageView
+imageView(image: Image)
+imageView(filenameOrURL: String)
```

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

The height of the bounding box within which the image is resized to fit.
The width of the bounding box within which the image is resized to fit.
The x-coordinate of the ImageView origin.
The y-coordinate of the ImageView origin.
The image to be displayed in the image view.
Creates an ImageView.
Creates an ImageView with the specified Image.
Creates an ImageView with image loaded from the specified file or URL.

See ShowImage.java

Layout Panes

JavaFX provides many types of panes for organizing nodes in a container.

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

-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)

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.

See ShowFlowPane.java

GridPane

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.

-GridPane()
-GridPane(ConstraintColumns columnConstraints, int...)
-ColumnConstraints(): void
-RowConstraints(): void

GridPane(grid: List<List<Node>>)
GridPane(grid: GridPane, row, column constraints)
GridPane(grid: GridPane, row, column, horizontal, vertical)

Sets the alignment of the parent.
Sets a column to a new column. This method removes the node.
Sets a node to a new row. This method repopulates the node.
Sets the horizontal alignment for the child in the cell.
Sets the vertical alignment for the child in the cell.

See ShowGridPane.java

BorderPane

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.

-BorderPane()
-BorderPane(child: Node, pos: Pos)

See ShowBorderPane.java

HBox

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.

-HBox()
-HBox(spacing: double)

See ShowHBox.java
VBox

- `alignment`: ObjectProperty<Box>
- `fillWidth`: BooleanProperty
- `spacing`: DoubleProperty

VBox()  // Creates a default VBox.

setter(node: Node, value: Insets): void

- `setMargin(node: Node, value: Insets): void`

Shapes

JavaFX provides many shape classes for drawing texts, lines, circles, rectangles, ellipses, arcs, polygons, and polylines.

Text

- `text`: StringProperty
- `x`: DoubleProperty
- `y`: DoubleProperty
- `underline`: BooleanProperty
- `font`: ObjectProperty<Font>

+Text()
+Text(x: double, y: double, text: String)

Text example

(a) Text(x, y, text)
(b) Three Text objects are displayed

- `getTextX()`: number
- `getTextY()`: number
- `getTextWidth()`: number
- `getTextHeight()`: number

- `setX(x: number)`: void
- `setY(y: number)`: void
- `setFont(font: Font)`: void

- `setUnderline(underline: boolean)`: void

+Text(text: String)
+Text(x: double, y: double, text: String)

See ShowText.java
Line

```java
class Line {
    -startX: DoubleProperty
    -startY: DoubleProperty
    -endX: DoubleProperty
    -endY: DoubleProperty

    Line() {
        startX, startY, endX, endY = 0;
    }
}
```

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.

See ShowLine.java

Rectangle

```java
class Rectangle {
    -x: DoubleProperty
    -y: DoubleProperty
    -width: DoubleProperty
    -height: DoubleProperty
    -arcWidth: DoubleProperty
    -arcHeight: DoubleProperty

    Rectangle() {
        x, y, width, height, arcWidth, arcHeight = 0;
    }
}
```

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.

See ShowRectangle.java

Rectangle example

```
(x, y) \rightarrow \frac{aw}{2}
\frac{ah}{2}
```

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

Circle

```java
class Circle {
    -centerX: DoubleProperty
    -centerY: DoubleProperty
    -radius: DoubleProperty

    Circle() {
        centerX, centerY, radius = 0;
    }
    Circle(x: double, y: double) {
        centerX, centerY = x, y;
    }
    Circle(x: double, y: double, radius: double) {
        centerX, centerY, radius = x, y, radius;
    }
}
```

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.

The x-coordinate of the center of the circle (default 0), The y-coordinate of the center of the circle (default 0), The radius of the circle (default: 0).

See ShowCircle.java
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.

- centerX: DoubleProperty
- centerY: DoubleProperty
- radiusX: DoubleProperty
- radiusY: DoubleProperty

- Ellipse()
- Ellipse(x: double, y: double)
- Ellipse(x: double, y: double, radiusX: double, radiusY: double)

See ShowEllipse.java

Arc

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.

- centerX: DoubleProperty
- centerY: DoubleProperty
- radiusX: DoubleProperty
- radiusY: DoubleProperty
- startAngle: DoubleProperty
- length: DoubleProperty
- type: ObjectProperty={ArcType=OPEN, ArcType=CHORD, ArcType=ROUND}

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

See ShowArc.java

Arc examples

(a) Negative starting angle –30° and negative spanning angle –20°
(b) Negative starting angle –50° and positive spanning angle 20°

See ShowArc.java

Polygon and Polyline

(a) Polygon
(b) Polyline
Polygon

java.x.scene.shape.Polygon

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

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.

Create an empty polygon.
Create a polygon with the given points.
Returns a list of double values as x- and y-coordinates of the points.

See ShowPolygon.java

Case study: the ClockPane class

This case study develops a class that displays a clock on a pane.

java.x.scene.layout.Panell

+getToken()
+setToken()

The getter and setter methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

The hour in the clock.
The minute in the clock.
The second in the clock.
Constructs a clock with the specified time.

See ClockPane.java
See DisplayClock.java