GUI Output

Adapted from slides by Michelle Strout with some slides from Rick Mercer

CSc 210
GUI (Graphical User Interface)

- We all use GUI’s every day
- Text interfaces great for testing and debugging
- Infants know how to use GUIs on phones
- What GUI features ...
  - Do you like most?
  - Have Opportunities For Improvement (OFI)?
AWT then Swing

- Java's first GUI support (1995) was called the Abstract Window Toolkit (AWT)
- AWT had some limitations, and some particular problems with how it was implemented on some platforms
- AWT was replaced by a new library called Swing
  - more versatile, more robust, and more flexible
- Swing was designed primarily for use in desktop applications
  - Could do some web-based things with it though Applets
GUI I/O Implemented with Libraries

• Some popular GUI Libraries and App Frameworks
  • Python has tKinter
  • Java has Swing
  • C++ has QT
  • Unity
  • .NET on Microsoft Windows Platforms

• Java FX
  • Improving on Swing
  • Will be used in 335
Today’s Goals

• Draw some shapes with JavaFX
  • Basics for making a window
  • Coordinate system used in JavaFX
  • Interface for drawing basic shapes
Today we will learn to draw
JavaFX Layout

A JavaFX Application has 3 major components:

1. Stage
2. Scene
3. Nodes
Stage

• A stage (window) contains the JavaFX objects

• The primary stage is created by the platform

• There are five types of stages available

• You must call the show() method to display the contents of the stage.
Scene

• A scene represents the physical contents of the application.

• Like a play, you can use different scenes to show different displays

• When you create a scene you can opt for the display size and give it's root node.
Scene Graph

- A Scene Graph (tree) represents the objects in a scene. It consists of nodes (Node class)

- There are three types of nodes:
  1. Root Node - the root of the tree
  2. Parent Node
  3. Leaf Node
Parent Node

- The root node as well as all parent nodes must be a child of the abstract class Parent. It's children are:
  - **Group** - a collective node that contains a list of children nodes
  - **Region** - the base class of all JavaFX Node based UI controls
  - **WebView** - manages the web engine
Basics for making a window (slide 1)

- Classes you will need to import

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
```

- Inherit from Application class and call launch()

```java
public class Drawing extends Application {
    public static void main(String[] args) {
        launch(args);
    }
    ...
}
```
**Basics for making a window (slide 2)**

- `launch()` (or something in its call tree) calls `start()` and passes in a Stage
- **Stage will contain Scene will contain RootPane**

```java
public class Drawing extends Application {
    public static void main(String[] args) {
        launch(args);
    }
    @Override
    public void start(Stage stage) {
        // create the root pane and then the scene
        BorderPane root_pane = new BorderPane();
        Scene scene = new Scene( root_pane, 400, 300 );

        //===== Drawing stuff goes here =====
        //===================================
        // Set the scene on the stage and show the stage.
        stage.setScene( scene );
        stage.show();
    }
}
```
Drawing on a Canvas

- Stage will contain Scene will contain RootPane
  - Will contain a Canvas

```java
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;

... //===== Drawing stuff goes here =====
// Construct a Canvas object and put it in
// the root pane.
Canvas canvas = new Canvas( 300, 200 );
rootPane.getChildren().add(canvas);
// Drawing on the canvas, getting a graphics context
// and calling drawing methods on it.
GraphicsContext gc = canvas.getGraphicsContext2D();
gc.fillText("Text in a Canvas at 30,40", 30, 40);
gc.fillText("Text in a Canvas at 30,50", 30, 50);
//==================================
```
JavaFX Coordinate System

- A simple two-dimensional coordinate system exists for each graphics context, or **drawing surface**
- Each point on the coordinate system represents a pixel
- Top left corner of the area is coordinate <0, 0>

```java
// This string will be drawn 20 pixels right,
// 40 pixels down as the lower left corner.
// All other shapes point is the upper left
gc.fillText("I'm in a Canvas", 20, 40);
```

- A drawing surface has a width and height
- Anything drawn outside of that area is not visible
The Coordinate System

\(<0, 0>\)

\(x\)

\(y\)

\(<x, y>\)

\(<\text{width}-1, \text{height}-1>\)

\(X\)

\(Y\)
Draw Common Shapes

• What does this do?

    gc.strokeOval(20, 20, 40, 40);
A couple method headings

- Lines are drawn with **stroke** (lines) and **fill** (solid)
- Oracle has an API online for JavaFX just like for Java:  [https://docs.oracle.com/javase/8/javafx/api/toc.htm](https://docs.oracle.com/javase/8/javafx/api/toc.htm)

```java
public void fillOval(double x, double y, double w, double h)
```

Fills an oval using the current fill paint.

**Parameters:**
- **x** - the X coordinate of the upper left bound of the oval.
- **y** - the Y coordinate of the upper left bound of the oval.
- **w** - the width at the center of the oval.
- **h** - the height at the center of the oval.
Color

• The `Color` class is used to define and manage the color in which shapes are drawn.

• Can set the color for stroke and fill with `setFill(Color)` and `setStroke(Color)`.

• `javafx.scene.paint.Color;` has many, many colors from `Color.AliceBlue` to `Color.YellowGreen`. 
Color

- Colors can also be defined with an *RGB value*, to set the relative contribution of the primary colors red, green, blue.

```java
Color color = Color.rgb(80, 210, 110);
gc.setFill(color);
gc.setStroke(color);
gc.setLineWidth(4);  // width now 4 pixels

gc.strokeOval(20, 20, 40, 40);
gc.fillOval(70, 20, 40, 40);
```
Clearing GraphicsContext

public void clearRect(double x, double y, double w, double h)

Clears a portion of the canvas with a transparent color value.

Parameters:

- **x** - X position of the upper left corner of the rectangle.
- **y** - Y position of the upper left corner of the rectangle.
- **w** - width of the rectangle.
- **h** - height of the rectangle.
Clearing GraphicsContext

Example Code:

```java
Canvas canvas = new Canvas(300, 300);
GraphicsContext gc = canvas.getGraphicsContext2D();
gc.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());
```
Interacting with Command Line

• Suppose we want to type our drawing commands through the keyboard.
• Can we just add input commands to our start method?
• No. JavaFX will not display the stage until the start method returns. 😞
• To actually take commands we need something running while JavaFX displays the stage.
Threads

• A program is a set of instructions.

• A process is a running program, it has instructions, but also data values, and control location (what in the program is currently being executed)

• Multi-processing is when more than one process is being executed at once.
Threads

• **Multi-threading** is when more than one control location is being executed in the same process.

• A **thread** is a sequential flow of control through a program
Threads

• One way to launch a thread in Java is to create a class that extends Thread.

• This class should include a public void method run(). This is the method that will run when the thread is started.

```java
public class Worker extends Thread {
    @Override public void run() {
```
Threads

• You can then instantiate an object of the created class.

• Calling the `start()` method (a method of the Thread class) will create a new thread and start it executing in the classes `run()` method.

```java
Worker w1 = new Worker(gc);
w1.start();  //starts thread
```