Java Foundations

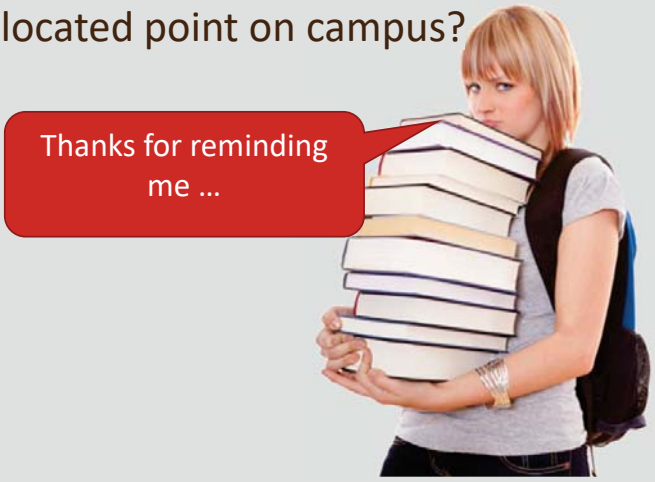
Introduction to JavaFX

Objectives

- This lesson covers the following objectives:
  - Create a JavaFX project
  - Explain the components of the default JavaFX project
  - Describe different types of Nodes and Panes
  - Explain the Scene Graph, Root Node, Scenes, and Stages

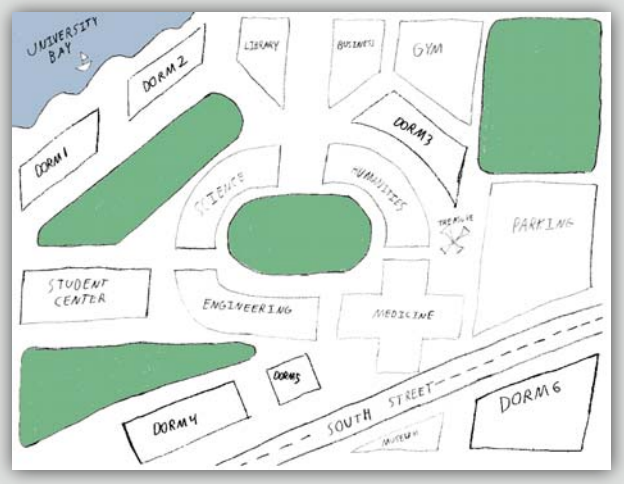
It's Almost Time for Final Exams!

- It's important to study
- Do you like to study with friends?
  - But do your friends live in other dorms?
  - Where is the best place to meet your friends?
  - What is the most centrally located point on campus?



JavaFX Can Help

- JavaFX is used to create GUI applications
- GUI: Graphical user interface
- A GUI application allows us to see the answer on a map



Exercise 1

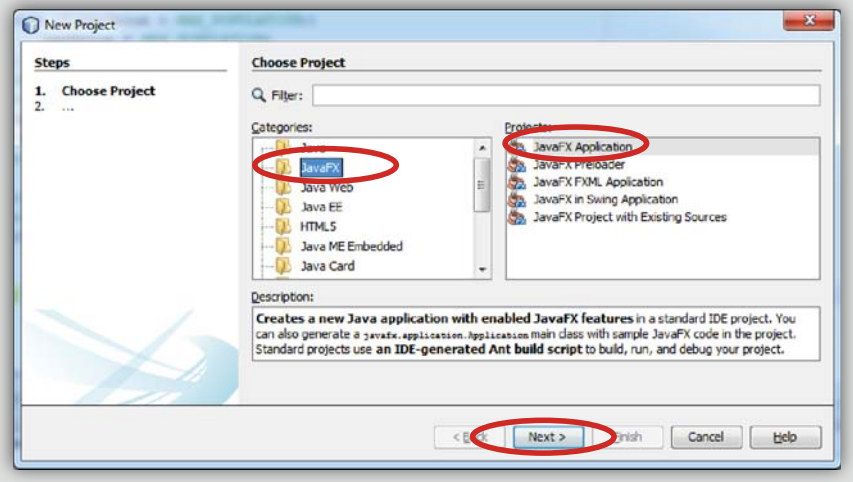
- Run `CampusMap.jar`
- Align each square with the correct dorm on the map
- Estimate and adjust each dorm's population
  - Click and drag the text below each square
- Observe changes in the following center points:
  - All students in all dorms
  - A study group of three friends living in Dorms 1, 2, and 4

But That's Not my Campus!

- You're right
- It would be better if the program used your school's …
  - Map of campus
  - Dorm names
  - Dorm populations
  - And your group of friends
- That's this section's problem set
- Section 9 discusses everything you'll need to re-create the program

How to Create a JavaFX Program

- In NetBeans, click the New Project button ()
- Select JavaFX for Category and JavaFX Application for Project, and then click Next
- Continue like you're creating any other Java project



Exercise 2

- Create a JavaFX project
  - Java should provide you with a default program
- Experiment with the program
- Can you make these changes?
  - Change the button's label
  - Change what's printed when the button is clicked
  - Create another button and display both buttons
  - Change the default size of the application's window

The Default JavaFX Project

```
public class JavaFXTest extends Application {

    @Override
    public void start(Stage primaryStage) {
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                System.out.println("Hello World!");
            }
        });

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

Continued on next slide..

The Default JavaFX Project

… continued

```
        Scene scene = new Scene(root, 300, 250);

        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }//end method start


    public static void main(String[] args) {
        launch(args);
    }//end method main
}//end class JavaFXTest
```

Two Methods: start() and main()

- start() is the entry point for all JavaFX applications
  - Think of it as the main method for JavaFX

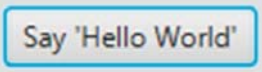
```
public void start(Stage primaryStage) {
    …
}//end method start
```

- main() is still required in your programs
  - It launches the JavaFX application

```
public static void main(String[] args) {
    launch(args);
}//end method main
```

Buttons Are Objects

- Buttons are like any other object
  - They can be instantiated
  - They contain fields
  - They contain methods




```
public void start(Stage primaryStage) {
    Button btn = new Button();
    btn.setText("Say 'Hello World'");
    …
}//end method start
```

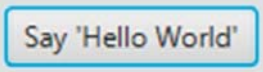
- From this code, we can tell …
  - Buttons contain a text field
  - Buttons contain a method for changing the text field

Buttons Are Nodes

- Some of these fields and methods are designed to store and manipulate visual properties:
  - btn.getText()
  - btn.setMinHeight()
  - btn.setLayoutX() **//set x position**
  - btn.setLayoutY() **//set y position**
  - btn.isPressed() **//is it pressed?**
- Objects like this are called JavaFX Nodes

Nodes

- There are many types of JavaFX Nodes:



Button

Rectangle

PieChart

ScrollBar

Text

ImageView

- Visual objects you'll create will most likely …
  - Be a Node, or
  - Include a Node as a field

Node Interaction

- The following helps handle Button interaction:

```
public void start(Stage primaryStage) {
   …
   btn.setOnAction(new EventHandler<ActionEvent>() {
      @Override
      public void handle(ActionEvent event) {
         System.out.println("Hello World!");
      }//end method handle
   });//end setOnAction
   …
}//end method start
```

- This is called an "**anonymous inner class**"
  - Doesn't the syntax look messy?
  - Java SE 8 Lambda expressions are an elegant alternative
  - We'll discuss Lambda expressions later in this section

Creating Nodes

- Nodes are instantiated like any other Java object:

```
public void start(Stage primaryStage) {
   Button btn1 = new Button();
   Button btn2 = new Button();
   btn1.setText("Say 'Hello World'");
   btn2.setText("222");
   …
}//end method start
```

- After you instantiate a Node:
  - It exists and memory is allocated to store the object
  - Its fields can be manipulated, and methods can be called
  - But it might not be displayed …

At least not yet …

Displaying Nodes

- There are a few steps to displaying a node

```
public void start(Stage primaryStage) {
   Button btn1 = new Button();
   Button btn2 = new Button();
   btn.setText("Say 'Hello World'");
   btn.setText("222");
   StackPane root = new StackPane();
   root.getChildren().add(btn1);
   root.getChildren().add(btn2);
   …
}//end method start
```

- First, add each Node to the Root Node
  - It's usually named root
  - It's very much like an ArrayList of all Nodes

Adding Nodes to the Root Node

- You could add each Node separately:

```
root.getChildren().add(btn1);
root.getChildren().add(btn2);
root.getChildren().add(btn3);
```

- Or you could add many Nodes at once:

```
root.getChildren().addAll(btn1, btn2, btn3);
```

Adding Nodes to the Root Node

- But don't add the same Node more than once
  - This causes a compiler error:

```
root.getChildren().add(btn1);
root.getChildren().add(btn1);
```

StackPane Root Node

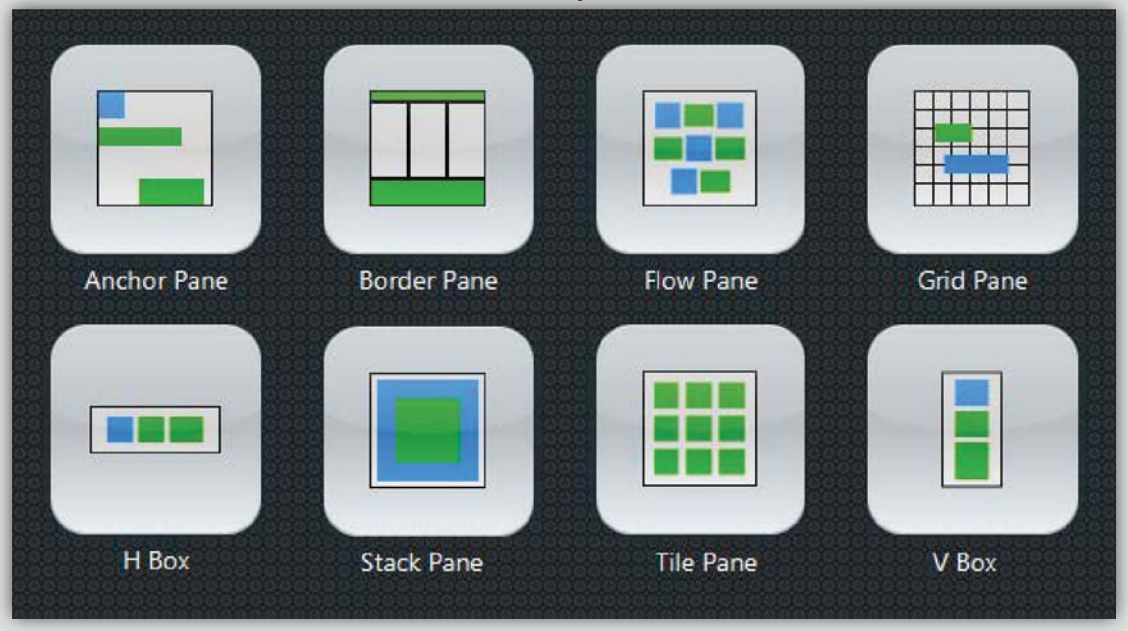
- The Root Node in this example is a StackPane

```
StackPane root = new StackPane();
root.getChildren().addAll(btn1, btn2);
```

  - The StackPane stacks Nodes on top of each other
  - But small buttons could become buried and unreachable

Panes as Root Nodes

- Each Pane determines the layout of Nodes



Programming Different Panes as Root Nodes

- It's easy to design the root node as a different pane
- Just specify a different reference type and object type

Change this

And this

```
StackPane root = new StackPane();
root.getChildren().addAll(btn1, btn2);
```

```
TilePane root = new TilePane();
root.getChildren().addAll(btn1, btn2);
```

```
VBox root = new VBox();
root.getChildren().addAll(btn1, btn2);
```

Exercise 3

- Edit your current JavaFX project
  - We're going to do a little experimenting
- After adding a button to the Root Node, try to change its position
  - `btn1.setLayoutY(100);`
- Will a button's position change if the Root Node wasn't a StackPane?
- Try these alternatives:
  - TilePane
  - VBox
  - Group

Group Root Node

- A Group allows you to place Nodes anywhere

```
Group root = new Group();
root.getChildren().addAll(btn1, btn2);
btn1.setLayoutY(100);
```

- A pane may restrict where Nodes are placed
  - You couldn't move them even if you wanted to
  - You couldn't click and drag a node that's locked in a pane

```
StackPane root = new StackPane();
root.getChildren().addAll(btn1, btn2);
btn1.setLayoutY(100);                //Has no effect
```

A Group Can Contain a Pane

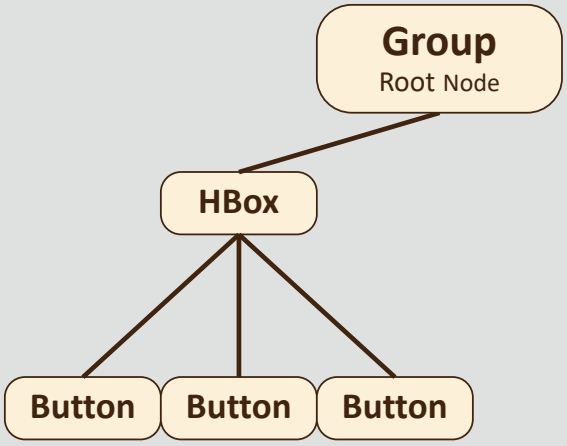
- Panes are also Nodes
  - Any node can be added to the Root Node
- A Pane may be a good option for storing buttons, text input dialog boxes, and other GUI elements
  - You can't quite move individual Nodes in a Pane
  - But you can move the entire Pane in a Group
  - Move the Pane like you would any other Node

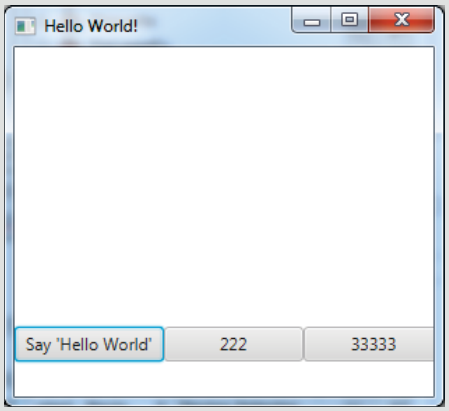
Exercise 4

- Edit your current JavaFX project
  - It's time for more experimenting
- Can you figure out how to do the following?
  - Create an HBox pane and add several buttons to it
  - Add the HBox pane to a Group Root Node
  - Position the HBox near the bottom of the window

The JavaFX Scene Graph

- How you decide to add nodes can be drawn as a Scene Graph
  - The Root Node contains an Hbox
  - The HBox acts as a container for buttons

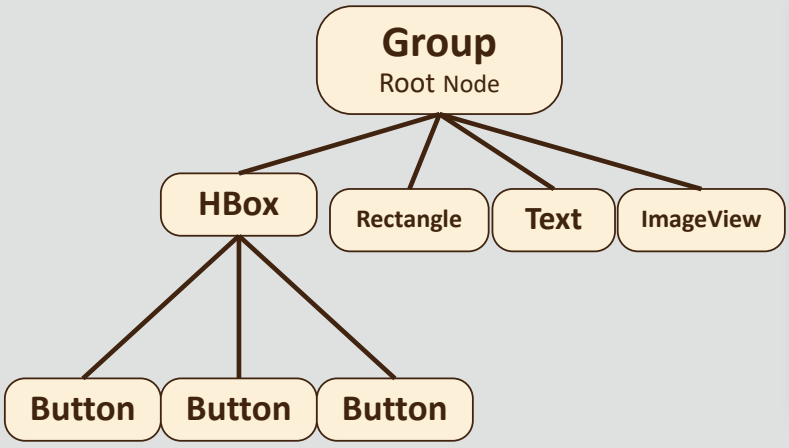


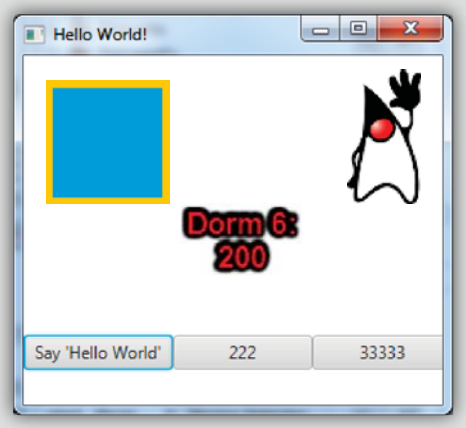


The Scene Graph

- The HBox keeps the GUI organized and conveniently located

The rest of the window could be used for other Nodes





The Scene and Stage

- If we look at the rest of the default JavaFX program, we notice two more things:
  - A Scene (which contains the Root Node)
  - A Stage (which contains the Scene)

```
public void start(Stage primaryStage) {
   …
   Scene scene = new Scene(root, 300, 250);

   primaryStage.setTitle("Hello World!");
   primaryStage.setScene(scene);
   primaryStage.show();
}//end method start
```

What Is the Scene?

- There are a few notable properties that describe a Scene:
- Scene Graph
  - The Scene is the container for all content in the JavaFX Scene Graph
- Size
  - The width and height of the Scene can be set
- Background
  - The background can be set as a Color or BackgroundImage
- Cursor Information
  - The Scene can detect mouse events and handles cursor properties

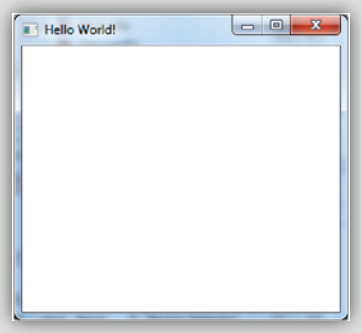
Root Node width height background

```
Scene scene = new Scene(root, 300, 250, Color.BLACK);
```

What Is the Stage?

- Think of the Stage as the application window

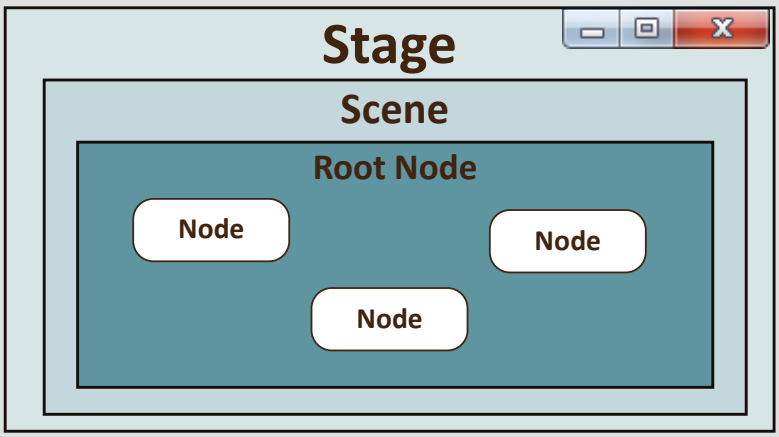
- Here are two notable Stage properties:
- Title
  - The title of the Stage can be set
- Scene
  - The Stage contains a Scene




```
primaryStage.setTitle("Hello World!");
primaryStage.setScene(scene);
primaryStage.show();
```

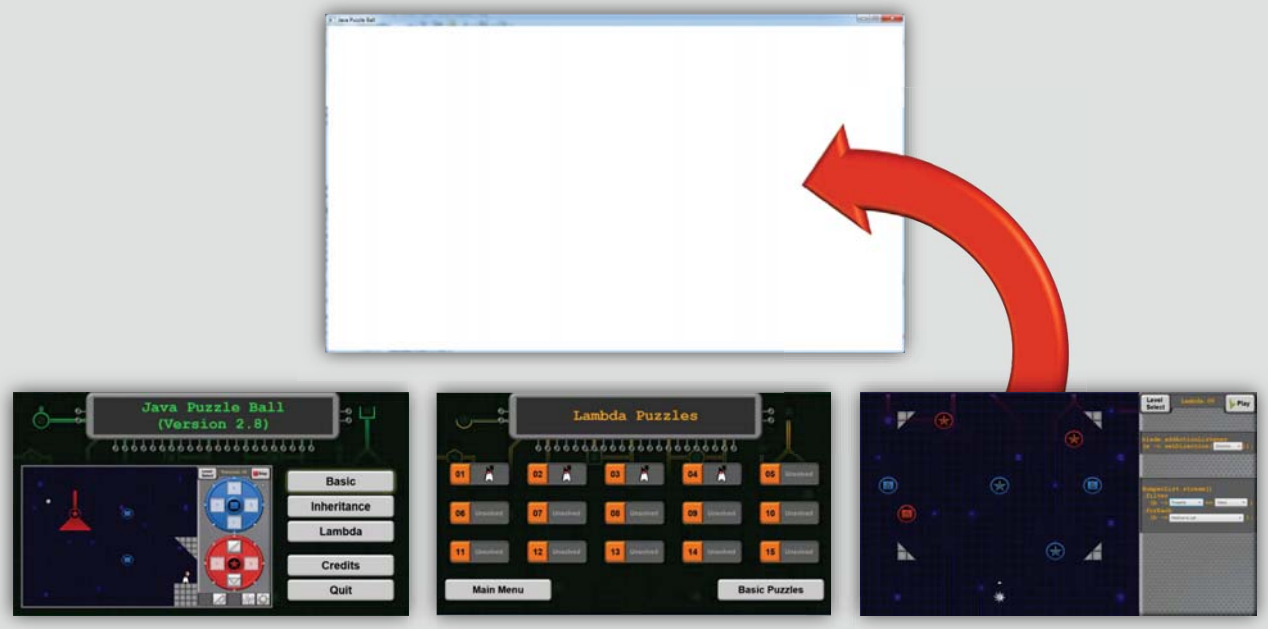
Hierarchy Animation

- A Stage is the top-level container
- A Stage contains a Scene
- A Scene contains a Root Node
- The Root Node contains other Nodes



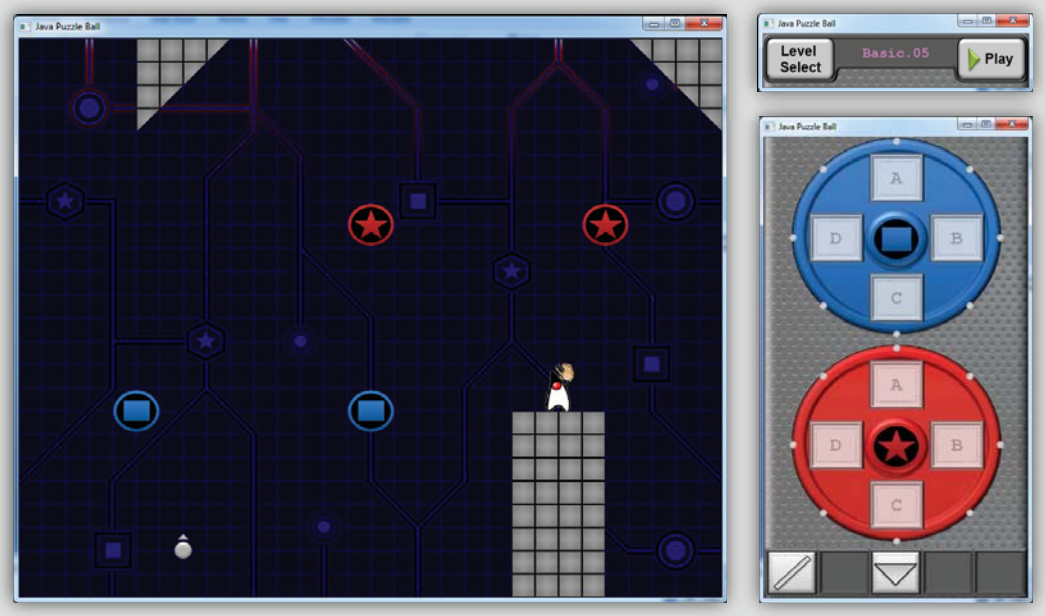
Many Scenes, One Stage

- It's possible to swap any scene into a single Stage



Many Scenes, Many Stages

- It's also possible to create many Stages



Summary

- This lesson covers the following objectives:
  - Create a JavaFX project
  - Explain the components of the default JavaFX project
  - Describe different types of Nodes and Panes
  - Explain the Scene Graph, Root Node, Scenes, and Stages