ORACLE

Academy

Java Foundations

9-2
Colors and Shapes

ORACLE
Academy

Guess What, Boys and Girls?!

- Today we're going to learn about colors and shapes!

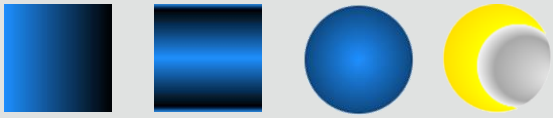
Objectives

- This lesson covers the following objectives:
  - Create and use custom colors
  - Create shapes and explain their properties and behaviors
  - Reference the JavaFX API Documentation

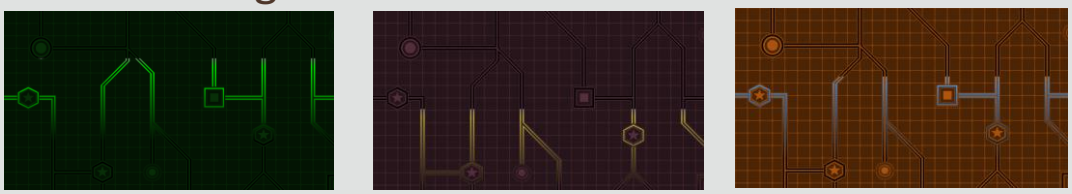
What Can I Do with Colors in JavaFX?

- Color shapes

- Create gradients



- Colorize images



JavaFX Contains a Color Class

- Colors can be stored as variables:

```
Color color = Color.BLUE;
```

- Colors can be passed in methods:

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

  - This example makes the scene's background black

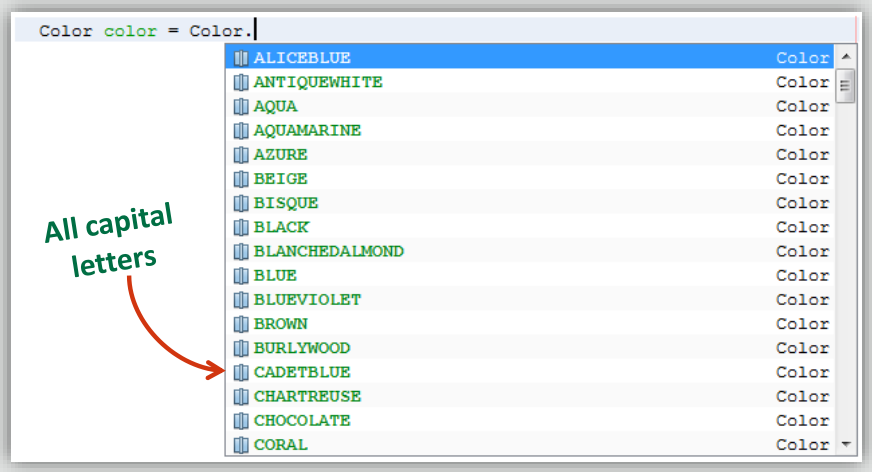
- But before using any Color …
  - You'll first need to make the following import:

```
import javafx.scene.paint.Color;
```

  - Ignore your IDE's other Color import suggestions

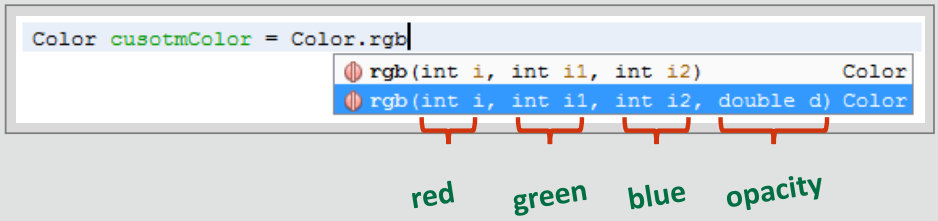
Referencing a Color

- There are many colors in JavaFX
- Typing `Color.` in your IDE reveals the entire list of possible colors



Customizing a Color

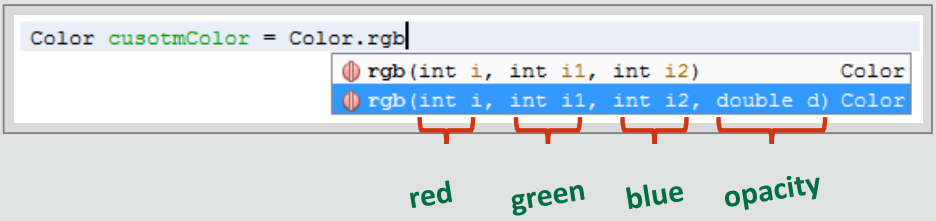
- If you're unhappy with the colors that JavaFX provides, there are ways to customize your own color
- The Color class contains methods to do this:



  - Customize a color by mixing red, green, and blue components
  - Opacity can also be controlled

JavaFX provides other methods for creating colors. You're welcome to use them in your programs, but we'll cover only the red-green-blue (RGB) method in this lesson.

The Range of Color Components



Component	Range of values
Red	0–255
Green	0–255
Blue	0–255
Opacity	0.0–1.0

Color Example

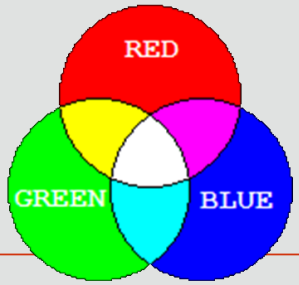
- In this example, the resulting color contains …

```
Color color = Color.rgb(255, 255, 20);
```

  - As much Red as possible
  - As much Green as possible
  - Only a little Blue

- The resulting color is very close to yellow
  - But how do we know this?
  - For the most part, finding the perfect color is “guess and check,” but there are guiding principles

Rules of Additive Color Mixing



Examples:

Code	Color	
`Color.rgb(255, 0, 0);`	red	Pure red
`Color.rgb(0, 255, 0);`	green	Pure green
`Color.rgb(0, 0, 255);`	blue	Pure blue
`Color.rgb(255, 255, 0);`	yellow	No blue
`Color.rgb(0, 0, 0);`	black	No color
`Color.rgb(255, 255, 255);`	white	All color

Colors are created according to the rules of additive color mixing.

Exercise 1

- Create a new JavaFX project using `JavaFXMainEx1.java`
  - `JavaFXMainEX1.java` is a copy of `JavaFXMain.java`
  - Change the Root Node to a Group type
  - Remove the button and any other unnecessary code relating to the button
- Experiment with customizing colors
  - Create a few custom colors
  - Admire your custom colors through the scene's background by providing a Color argument when the Scene is instantiated

This Is a Rectangle

- This is how to instantiate a JavaFX Rectangle:

```
Rectangle rect = new Rectangle(20, 20, 100, 200);
```

x-position, y-position, width, height

- You'll first need to make the following import:

```
import javafx.scene.shape.Rectangle;
```

  - Ignore your IDE's other Rectangle import suggestions

Important Methods for Rectangles

- We can get a sense of a Rectangle's properties from the constructor and the following methods:
  - setX(double d)
  - setY(double d)
  - setWidth(double d)
  - setHeight(double d)
  - setFill(Paint paint)
  - setStroke(Paint paint)
  - setStrokeWidth(double d)
    - (There are many more Rectangle methods besides these seven)

These can accept a color as an argument

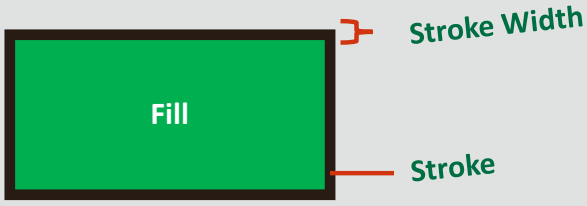
- But what exactly will these methods do?

Exercise 2

- Continue editing the JavaFX project that you created in the previous exercise
- Create a Rectangle and add it to the Root Node
- Call each method outlined in the previous slide
- Can you figure out what each method does?

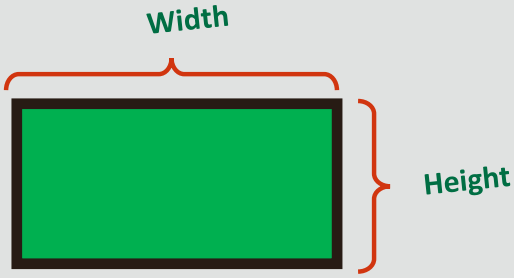
Method Descriptions, Part 1

- setFill(Paint paint)
  - Sets the color of the Rectangle
- setStroke(Paint paint)
  - Sets the color of the Rectangle's outline
- setStrokeWidth(double d)
  - Sets the width of the Rectangle's outline



Method Descriptions, Part 2

- setX(double d)
- setY(double d)
  - Sets the x or y position of the Rectangle
- setWidth(double d)
- setHeight(double d)
  - Sets the width or height of the Rectangle



Changing a Node's Position

- We've seen a couple ways to change a node's position … but which way is preferable?

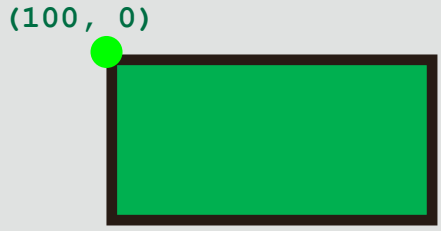
- setX(double d)
- setY(double d)
  - These are preferable in most cases

- setLayoutX(double d)
- setLayoutY(double d)
  - Use these if your Node is locked in a Layout pane, such as a FlowPane
  - Or if setX() is unavailable, which is the case with UI elements, such as Buttons

setX() definitely won't work in this case

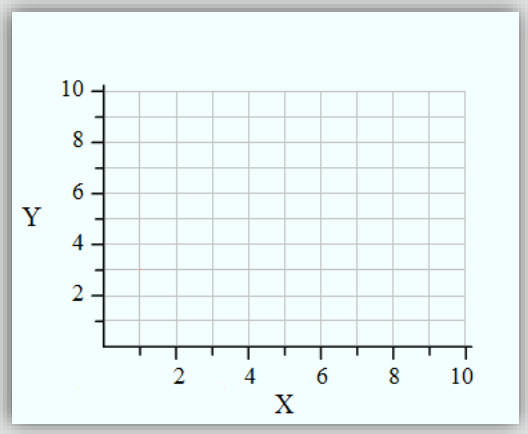
Positioning a Node

- Most Nodes are positioned with respect to their top-left corner
  - And not with respect to their geographic center
- If you call setX(100) on a Node …
  - The x-position of the Node's top-left corner is set to 100



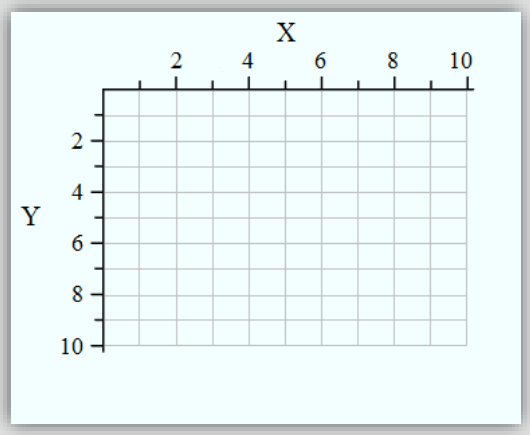
A JavaFX Circle is an exception to this rule. JavaFX Circles are positioned with respect to their center.

Coordinate Systems



Mathematical Coordinate System

- The origin is located at the bottom-left corner

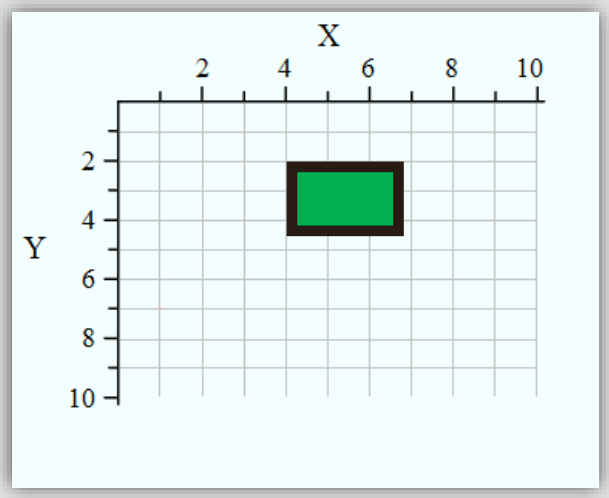


JavaFX Coordinate System

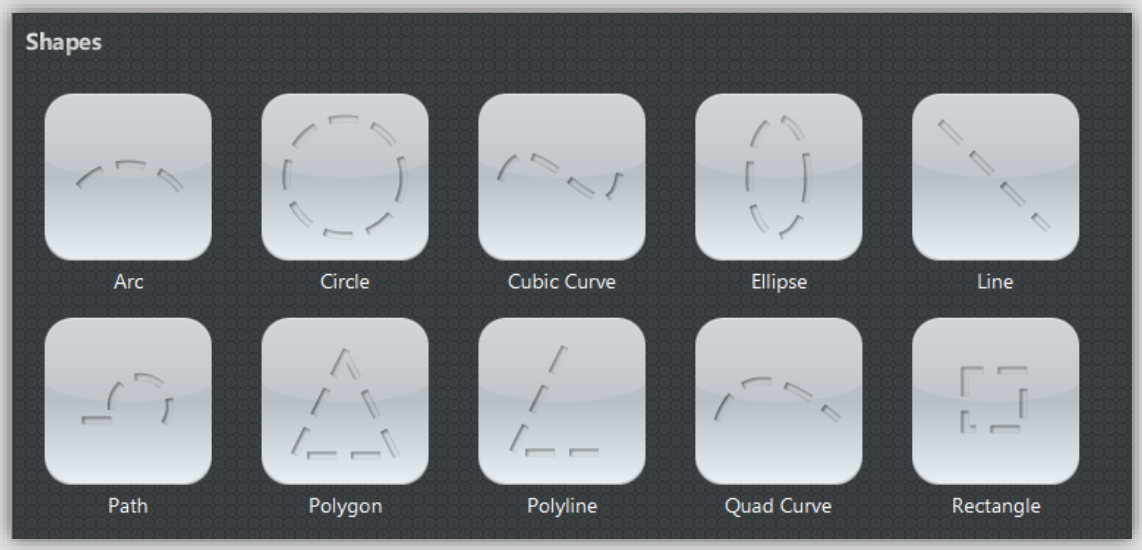
- The origin is located at the top-left corner
- The y-axis is backward

Positioning Example

- This Rectangle is positioned at (4,2) by calling:
  - `setX(4);`
  - `setY(2);`



Many Shapes Are Available in JavaFX



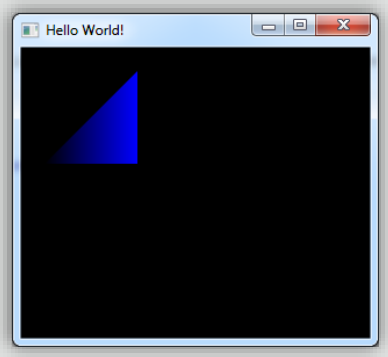
The JavaFX API Documentation

- This contains class information and code examples of JavaFX features
- Go to https://openjfx.io/javadoc/17/index.html
- The Graphics module is a helpful starting point
- There is a search feature to allow you to locate specific classes, or you can browse the packages for ideas

We were learning JavaFX at the same time we were developing the game.

Exercise 3

- Explore the JavaFX API Documentation
- Can you figure out how to create a right triangle with a gradient coloring using the JavaFX project that you created in the previous exercise?
- Hint: use the search box to first search for gradient, then search for Polygon



Exploring the API Documentation: Linear Gradient Example

- The Linear Gradient example shows us …
  - How to create a gradient:


```
//create simple linear gradient
LinearGradient gradient1 = new LinearGradient(0, 0, 1, 0, true,
CycleMethod.NO_CYCLE, new Stop[] {
        new Stop(0, Color.DODGERBLUE),
        new Stop(1, Color.BLACK)
});
```

  - How to color a shape with a gradient:

```
//First rectangle
Rectangle rect1 = new Rectangle(0,0,80,80);

//set rectangle fill
rect1.setFill(gradient1);
```

  - Remember to make the proper imports

Exploring the API Documentation: Polygon Example

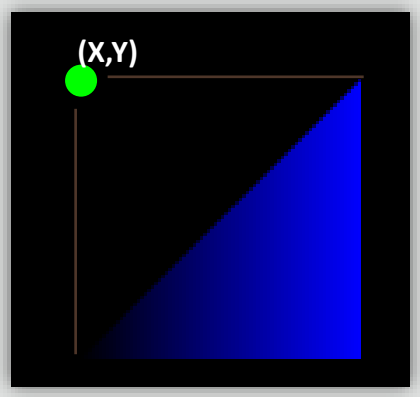
- Type polygon in the API search box - the Polygon example shows us …
  - How to create a polygon from an array of points:

```
//Simple triangle
Polygon polygon1 = new Polygon(new double[]{
        80.0, 10.0,
        80.0, 80.0,
        10.0, 80.0
});
```

- Combine this with the gradient example, and you'll have your solution
  - But even better, you'll understand how the API Documentation is a valuable resource
  - This could prove very useful when you do the problem set

The Polygon

- The Polygon has similar methods as a Rectangle
  - Nodes share the same methods
- If you experiment with setLayoutX()…
  - You'll notice that the Polygon is positioned with respect to where its top-left corner would be



Secrets about Java Puzzle Ball

- We drew lines and polygons for collision detection
  - But these lines are hidden in the latest version

- We also drew two octagons around each bumper
  - An inner octagon handles collision detection
  - An outer octagon detects if the ball is far enough away for the bumper to rotate
- We had to do extra work to position and rotate Nodes the way we wanted

Summary

- In this lesson, you should have learned how to:
  - Create and use custom colors
  - Create shapes and explain their properties and behaviors
  - Reference the JavaFX API Documentation


ORACLE
Academy