



ORACLE

Academy



Java Foundations

9-3

Graphics, Audio, and MouseEvents

ORACLE
Academy



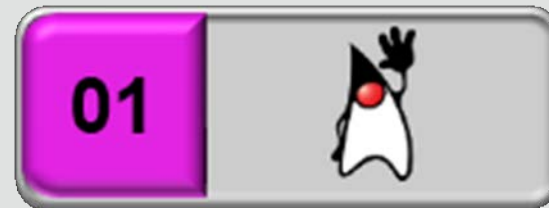
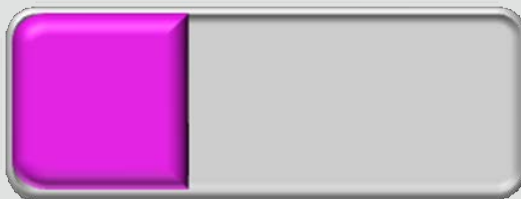
Objectives

- This lesson covers the following objectives:
 - Create and use a JavaFX image and ImageView
 - Create and use JavaFX audio
 - Create and use MouseEvents
 - Understand Lambda expressions in GUI applications



Using Your Own Graphics

- JavaFX can provide UI elements, shapes, and text
 - But if you have a talent for art, you can use your own graphics in place of those that JavaFX provides
- For example:



- The art for the level-select button wasn't created by JavaFX
- But we used JavaFX to procedurally add level numbers, text, and the graphic of Duke



A JavaFX Image and ImageView

- An Image is an object that describes the location of a graphics file (.png, .jpg, .gif ...)

```
Image image;  
String imagePath = "Images/Fan1.png";  
image = new Image(getClass().getResource(imagePath).toString);
```

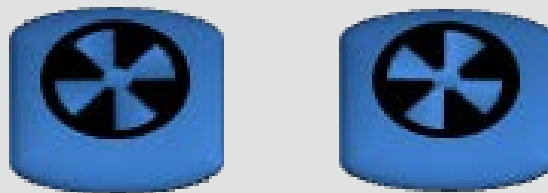
- An ImageView is the actual Node
 - Calling its constructor requires an Image argument

```
ImageView imageView = new ImageView(image);
```

- An ImageView also contains the same properties as any other node: x-position, y-position, width, height ...

Why Have Both an Image and ImageView?

- One big advantage is animation
 - Images can be swapped in and out of the same ImageView
- The Fan in Java Puzzle Ball takes advantage of this
 - The fan cycles through 2 images when it's blowing



- Custom buttons also benefit
 - You could use different images for buttons depending on their state:
 - Is the mouse hovering over the button?
 - Is the user clicking the button?

ImageView Hints

- How to create Images:

```
Image image1 = new  
Image(getClass().getResource("Images/fan1.png").toString());  
Image image2 = new  
Image(getClass().getResource("Images/fan2.png").toString());
```

- How to create an ImageView:

```
ImageView imageView = new ImageView(image1);
```

- How to swap an Image into an ImageView:

```
imageView.setImage(image2);
```

– imageView retains its properties, such as positioning

**Remember to import
javafx.scene.image.Image; and
javafx.scene.image.ImageView;**

Creating Objects with Node Properties

- So far, we've written all JavaFX code in the `start()` method
 - This is similar to the beginning of the course, where most code was written in the `main()` method
- Object-oriented code shouldn't be written this way
 - Instead, objects should have Node fields
- The `start()` and `main()` methods are intended to be drivers

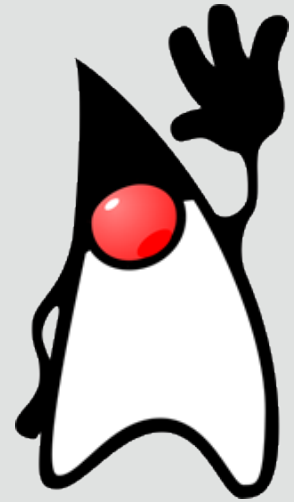
Example: The Goal Class

- Fields

- `private Image dukeImage;`
 - `private ImageView dukeImageView;`

- Constructor

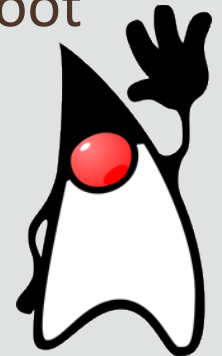
- Takes arguments for x and y positions
 - Assigns the image to its respective ImageView
 - Positions dukeImageView according to the x and y arguments





Exercise 1

- Import and open the GoalTest project
- Notice that ...
 - The Root Node is publically available
 - There's a package with several graphic files
 - The Goal class is an ordinary Java class file type
- Write the Goal class according to the specifications on the previous slide
 - You'll also need to add this class's ImageView to the Root Node
- Instantiate a few Goal objects from the start() method



File Locations

- Make sure files are in the correct location

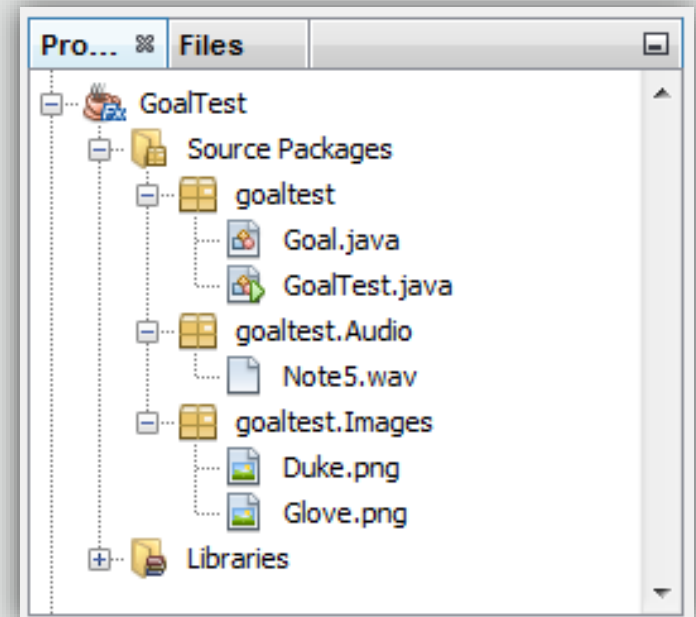
```
Image image = new Image(getClass().getResource("Images/Duke.png"));
```

- Images/Duke.png refers to a folder within the GoalTest folder

– ... \GoalTest \src \goaltest \Images

Project Folder Source Primary Package Another Package

– Or a package within a package

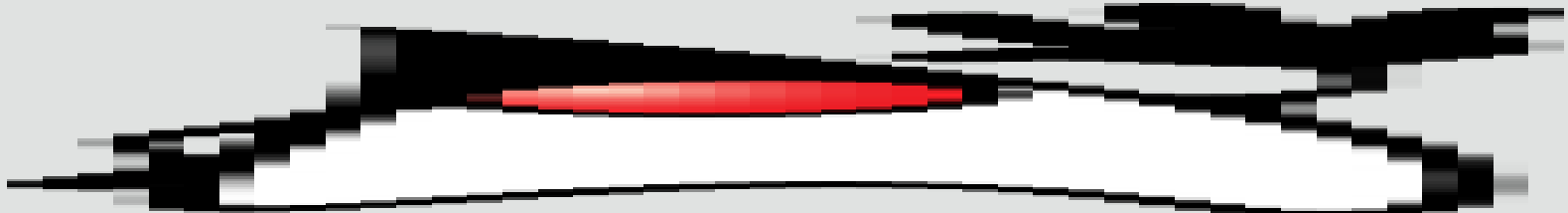


Scaling a Node

- It's very easy to make a rectangle wider:



- But if you try the same thing with an ImageView ...
 - It might look awful!



Scaling a Node the Right Way

- JavaFX is very good at scaling graphics
 - The quality of the image is less likely to deteriorate
- You have the option to preserve the aspect ratio of an ImageView
 - An ImageView's width and height scale together
 - This avoids distortion

```
imageView.setPreserveRatio(true);  
imageView.setFitWidth(25);
```

Ordering Nodes

- Sometimes, testers of Java Puzzle Ball didn't realize that their goal was to get the ball to Duke
- We thought adding a baseball glove would help solve the problem
- Duke and the glove are two separate ImageViews
 - These needed to be ordered properly so that the glove doesn't display behind the hand



Correct



Incorrect

Ordering Nodes the Right Way

- The order that Nodes are added to the Root Node determines the order that they are displayed
- Nodes added early are buried under nodes added later

```
root.getChildren().addAll(gloveImageView, dukeImageView);
```

- To fix this you could ...
 - Change the order that Nodes are added to the Root Node
 - Bring an ImageView to the front or back

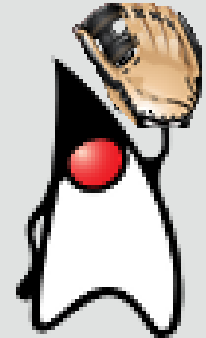
```
gloveImageView.toFront();    //Either one of these  
dukeImageView.toBack();     //will solve the problem
```



The Goal Class

- Fields

- private Image dukeImage;
 - private ImageView dukeImageView;
 - private Image gloveImage;
 - private ImageView gloveImageView;



- Constructor

- Takes arguments for x and y positions
 - Assigns each Image to its respective ImageView
 - Positions dukeImageView according to the x and y arguments
 - Positions and scales gloveImageView relative to dukeImageView



Exercise 2

- Continue editing the GoalTest project
- Write the Goal class according to the specifications on the previous slide
 - The constructor should still take only two arguments
 - A glove should appear on top of Duke's hand
- **Hint:** Nodes, including ImageViews, have getter and setter methods for properties like position



Image and Audio Similarities

- Creating a JavaFX Image object ...

```
Image image = new  
Image(getClass().getResource("Images/fan1.png").toString());
```

- Is very similar to creating a JavaFX Audio object

```
Audio audio = new  
Audio(getClass().getResource("Audio/Note5.wav").toString());
```

- It's common to store images and audio in their own packages/folders

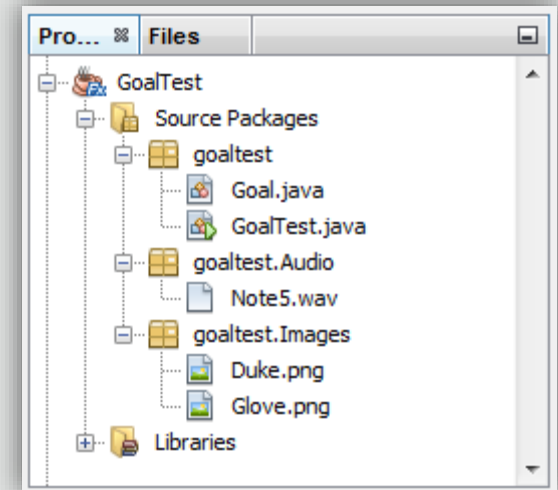


Image and Audio Differences

- An Audio object describes the location of an audio file (.wav, .mp3 ...)

```
Audio audio = new  
    Audio(getClass().getResource("Audio/Note5.wav").toString());
```

- And unlike an Image ...
 - There is no Audio equivalent of an ImageView
 - Audio can be played by referencing the Audio object directly

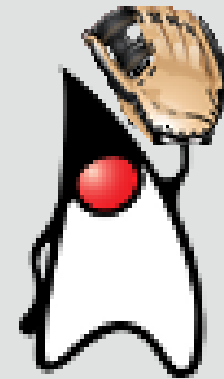
```
audio.play();
```

- There are many other Audio methods you can call

The Goal Class

- Fields

```
-private Image dukeImage;  
-private ImageView dukeImageView;  
-private Image gloveImage;  
-private ImageView gloveImageView;  
-private Audio tone;
```



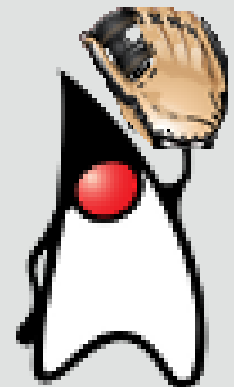
- The Goal class contains an Audio object as a field
 - tone plays when the mouse is pressed on Duke
 - We'll see how to implement this feature in the next part of this lesson



Exercise 3

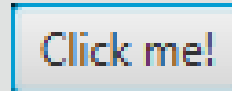
- Continue editing the GoalTest project
- Declare an Audio object as a field
- Instantiate the Audio object
 - Use the .wav file in the project directory

Remember to import
`javafx.scene.media.AudioClip;`



Mouse and Keyboard Events

- Nodes can detect mouse and keyboard events
 - This is true about ImageViews, too!
 - You aren't limited to buttons and other GUI components.
- Helpful methods to make this happen include:
 - `setOnMouseClicked()`
 - `setOnMouseDragged()`
 - `setOnMouseEntered()`
 - `setOnMouseExited()`
 - `setOnMouseMoved()`
 - `setOnMousePressed()`
 - `setOnMouseReleased()`



*Remember to import
`javafx.scene.input.MouseEvent`*

Lambda Expressions

- These methods use a special argument, called a Lambda expression:

```
imageView.setOnMousePressed( /*Lambda Expression*/ );
```

- Lambda expressions use special syntax:

```
(MouseEvent me) -> System.out.println("Pressed")
```

No semicolon

- Curley braces allow Lambda expressions to contain multiple statements:

```
(MouseEvent me) -> {  
    System.out.println("Statement 1");  
    System.out.println("Statement 2");  
} //end MouseEvent
```

semicolons

Lambda Expressions as Arguments

- When these are combined, we get the following:

```
imageView.setOnMousePressed( (MouseEvent me) -> {  
    System.out.println("Statement 1");  
    System.out.println("Statement 2");  
} );
```

- What this code does:
 - Allows imageView to detect a mouse press at any time
 - If that occurs, the two print statements are executed
 - Otherwise, this code is ignored

MouseEvent

- A MouseEvent object exists only within the scope of the Lambda expression
- It contains many useful properties and methods:

```
imageView.setOnMousePressed( (MouseEvent me) -> {  
    System.out.println(me.getSceneX());  
    System.out.println(me.getSceneY());  
} );
```

- In this example:
 - me is the MouseEvent object
 - me is accessed to print the x and y positions of the mouse cursor when imageView is pressed



MouseEvent Methods

- `getSceneX ()`
- `getSceneY ()`
 - Returns a double
 - Returns the position of the cursor within the JavaFX Scene
 - The top-left corner of the Scene is position (0,0)
- `getScreenX ()`
- `getScreenY ()`
 - Returns a double
 - Returns the position of the cursor on your computer's screen
 - The top-left corner of your computer's screen is (0,0)

Event Listening

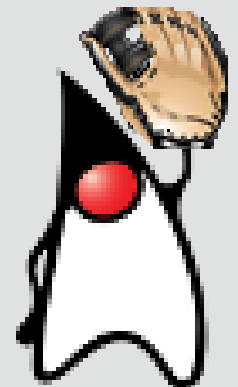
- When you write code for MouseEvents
 - You're telling a Node to listen for a particular event
 - But the events don't actually have to occur
- As long as the Node is listening ...
 - It can detect any event, at any time
- A Node can listen for many events

```
imageView.setOnMousePressed( /*Lambda Expression*/ );  
imageView.setOnMouseDragged( /*Lambda Expression*/ );  
imageView.setOnMouseReleased( /*Lambda Expression*/ );
```



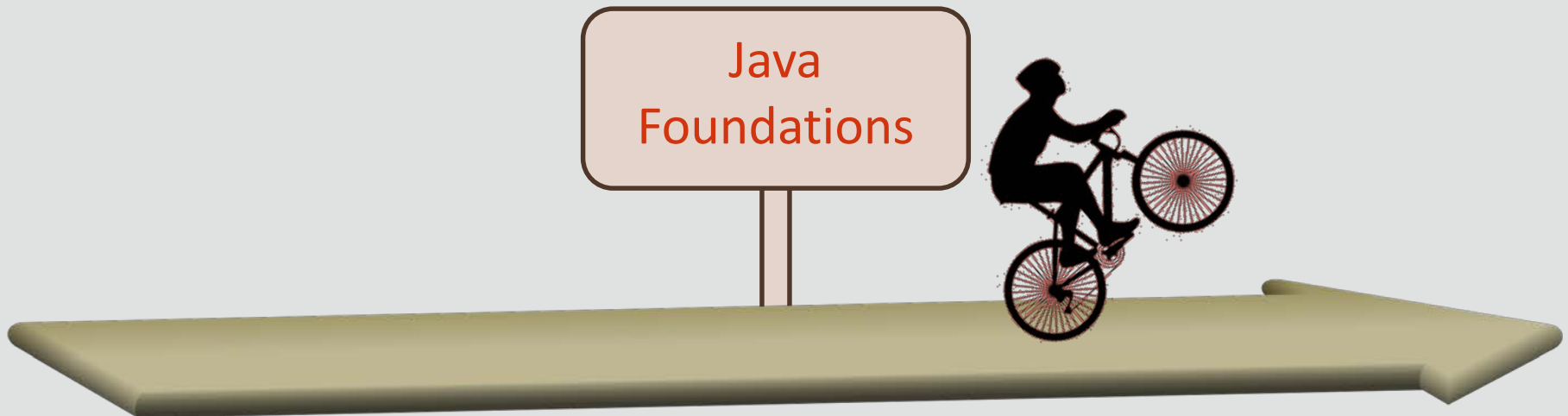
Exercise 4

- Continue editing the `GoalTest` project
- Complete the `interactions()` method so that ...
 - Duke listens for a mouse press and mouse drag
 - Play a sound when the mouse is pressed
 - Print the x and y positions of the mouse dragged event
 - This will be helpful for the problem set
- What if `interactions()` is never called?
 - Comment out this method call in the constructor



Summary

- In this lesson, you should have learned how to:
 - Create and use a JavaFX image and ImageView
 - Create and use JavaFX audio
 - Create and use MouseEvents
 - Understand Lambda expressions in GUI applications





ORACLE

Academy

