L02 - FRIDAY, AUGUST 23

CPSC 4150/6150 - FALL 2019

ANONYMOUS CLASSES

L02 - DATE

CPSC 4150/6150 - FALL 2019

ANONYMOUS CLASSES

checkin.cs.clemson.edu

code: PYVW1

CLEMSON SCHOOL OF COMPUTING

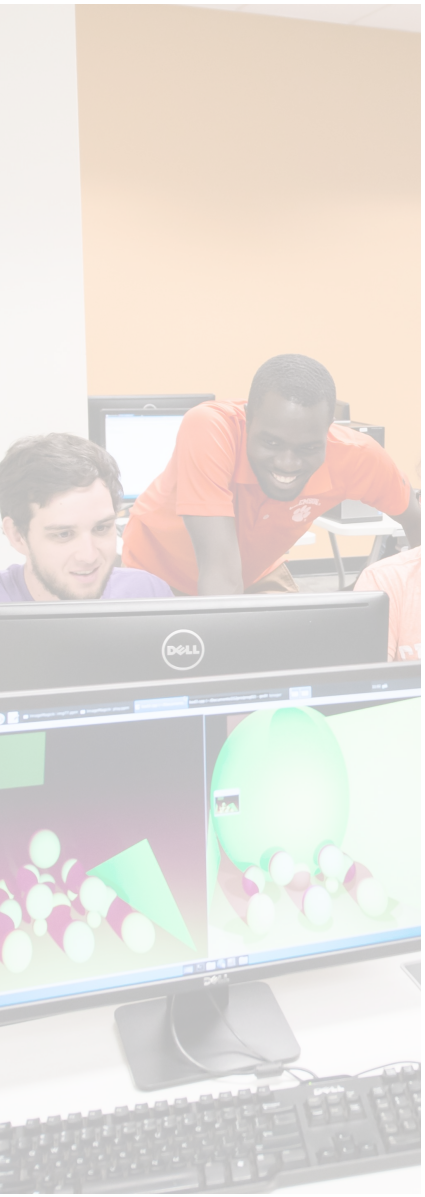
LECTURE OBJECTIVES

BY THE END OF THIS CLASS, YOU SHOULD BE ABLE TO:

Explain why anonymous classes are used

Implement an anonymous class and access local variables

WHY DO YOU USE ANONYMOUS CLASSES IN JAVA?



CODE DEMO

https://docs.oracle.com/javase/tutorial/java/javaOO/anonymousclasses.html

WHY ANONYMOUS CLASSES

More concise code (e.g., you only need to use a local class once)

Are expressions

Heavily used in user interface design in Java (e.g., Swing, JavaFX, Android)

Ideal for implementing an interface that contains two or more methods

You can declare the following within an anonymous class: fields, extra methods, instance initializers, local classes but not constructors

ACCESSING LOCAL VARIABLES

Anonymous classes have the same access to local variables of the enclosing scope

Anonymous classes cannot access local variables in its enclosing scope that are not declared final

Declaration of a type such a variable in an anonymous class shadows any other declarations in the enclosing scope with the same name

You cannot declare static initializers or member interfaces in an anonymous class

You can have static members if declared constant

JAVAFX EXAMPLE

```
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;

public class HelloWorld extends Application {
    public static void main(String[] args) {
        launch(args);
    }

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Hello World!");
        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);
        primaryStage.setScene(new Scene(root, 300, 250));
        primaryStage.show();
    }
```

@override

Recommended annotation to inform the compiler that a method is meant to override an element declared in a super class

FIN