GENERAL ASSEMBLY

MOBILE DEVELOPMENT

LESSON 05
CLASSES AND TYING INTERFACE BUILDER TO CODE

Tedi Konda
VP, Engineering and Technology, RepEquity

GETTING STARTED

READY?
SET?
GIT PULL!

GETTING STARTED

LESSON 04 REVIEW

WHAT DID WE LEARN IN LESSON

- Operators Continued
  - Unary
  - Binary
  - Ternary
- Optionals
  - Optional Binding
  - Optional Unwrapping
- Functions
  - All different types!

QUESTIONS

- I will call on some of you to come to the front and show examples of:
  - a Unary operator
  - a Binary operator
  - a Ternary operator
  - Optionals and Optional Binding
  - a Function with one parameters and no return type
  - a Function with two parameters and a return type

GETTING STARTED

LEARNING OBJECTIVES

LEARNING OBJECTIVES

- Object Oriented Principles
- Classes
- Arrays (lightly touch on Arrays)
- Anatomy of an Xcode Project
- Tying Interface Builder into Code (FINALLY!)

GETTING STARTED

OBJECT ORIENTED PRINCIPLES

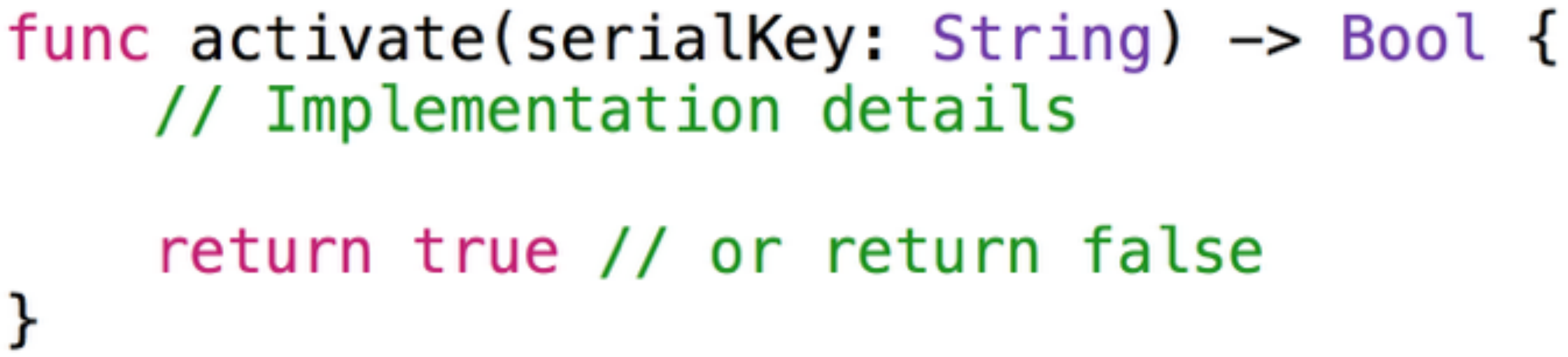
WHAT ARE THE PRINCIPLES?

- 4 Principles
  - Encapsulation
  - Abstraction
  - Inheritance
  - Polymorphism

OBJECT ORIENTED PRINCIPLES

ENCAPSULATION

- Imagine buying a piece of software (e.g., video game) that requires a serial key to be unlocked.
- When you pass in a serial key (e.g., String) to a validation function, that function runs a whole lot of code to check and see if the code is valid.
- The implementation details (e.g., guts) of the function are hidden.



```swift
func activate(serialKey: String) -> Bool {
    // Implementation details

    return true // or return false
}
```

ABSTRACTION (PT. 1)

- Let’s say you have a car - what properties does a typical car have?
  - Doors (Int)
  - Wheels (Int)
  - Spoiler (Bool)
  - Speed (Float)
  - Name (String)
  - Price (Float)

OBJECT ORIENTED PRINCIPLES

ABSTRACTION (PT. 2)

- What would make this car a Lamborghini?
  - Doors = 2
    - Door Type = Suicide
  - Wheels = 4
  - Spoiler = Maybe (true or false)
  - Top Speed = 349.0 km/h (218 mph)
  - Name = Lamborghini
  - Price = Expensive

ABSTRACTION (PT. 3)

- Abstraction in programming enables you, and enforces you to describe what you’re building as abstractly as possible.

- As programmers, we build on the abstract ideas by adding details to the abstract items.

INHERITANCE

- We said that a car is mainly defined by the following properties:
  - Doors (Int)
  - Wheels (Int)
  - Spoiler (Bool)
  - Speed (Float)
  - Name (String)
  - Price (Float)
  - Inheritance is the idea that stating that something is a type of a car gives it the properties of a car.

OBJECT ORIENTED PRINCIPLES

POLYMORPHISM

- Polymorphism := “One Name, Many Forms”
  - A polymorphic concept in Swift is Function Overloading, which is having a function with the same name, but different parameters and return types.
  - To Playgrounds!

GETTING STARTED

DATA STRUCTURES

WHAT IS A DATA STRUCTURE? (PT. 1)

- Data structures are groupings of variables (var), constants (let), and functions (func) that work together to ***describe itself*** and ***describe its purpose*** in the context of your program

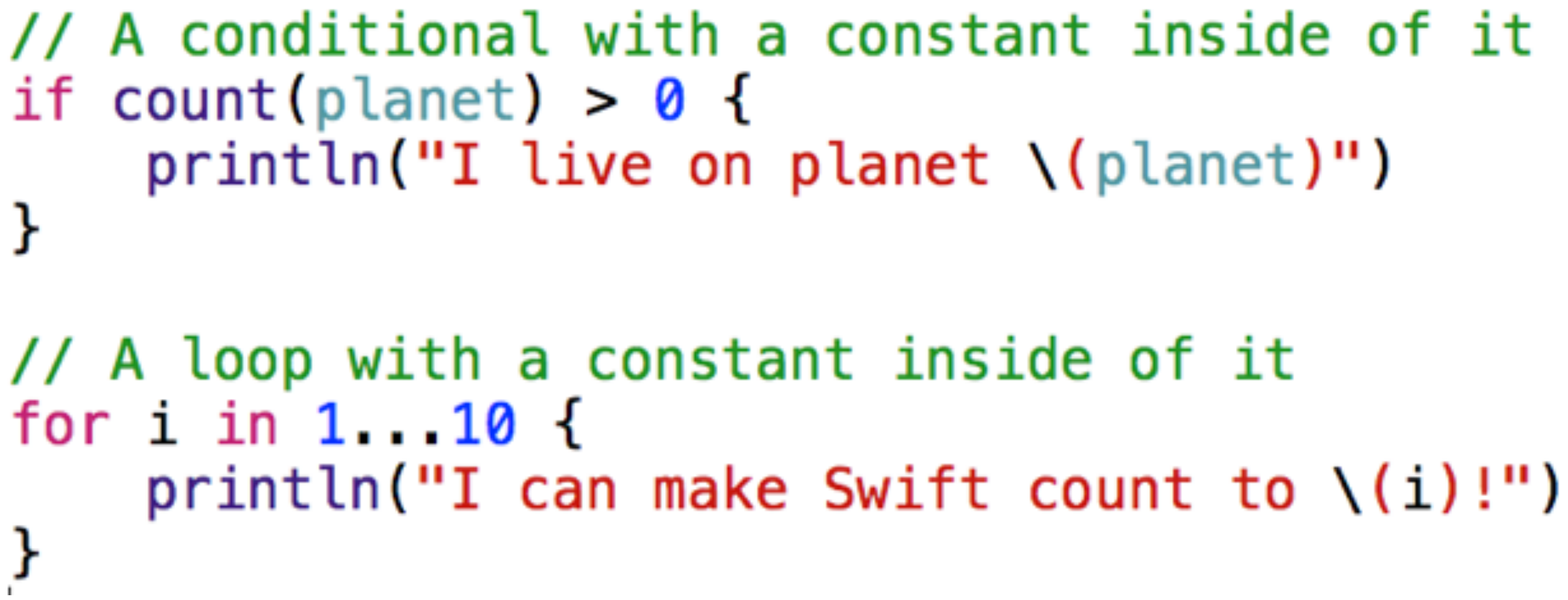
WHAT IS A DATA STRUCTURE? (PT. 2)

- We’ve learned about constants and variables by themselves.

```
// A constant
let planet = "Earth"
```

WHAT IS A DATA STRUCTURE? (PT. 3)

- We’ve been able to group them inside of conditionals (if-else), loops (while, for-in), and functions.

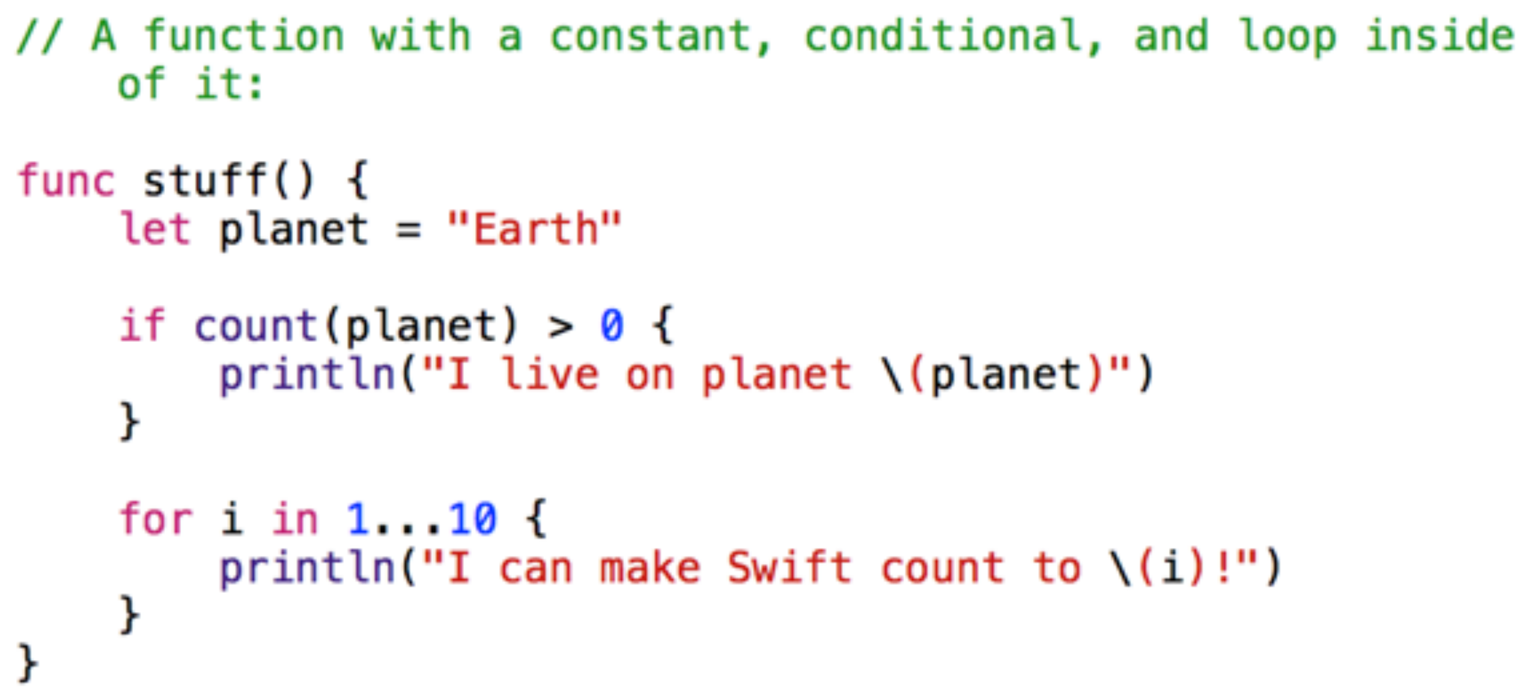


```
// A conditional with a constant inside of it
if count(planet) > 0 {
    println("I live on planet \(planet)")
}

// A loop with a constant inside of it
for i in 1...10 {
    println("I can make Swift count to \(i)!")
}
```

WHAT IS A DATA STRUCTURE? (PT. 4)

- We’ve been able to group constants, variables, loops, and conditionals inside of functions.



```
// A function with a constant, conditional, and loop inside
    of it:

func stuff() {
    let planet = "Earth"

    if count(planet) > 0 {
        println("I live on planet \(planet)")
    }

    for i in 1...10 {
        println("I can make Swift count to \(i)!")
    }
}
```

WHAT IS A DATA STRUCTURE? (PT. 5)

- The next step is two group functions inside of classes and structs!

- A class and a struct are groupings of variables (var), constants (let), and functions (func) that work together to ***describe itself*** and ***describe its purpose*** in the context of your program

GETTING STARTED

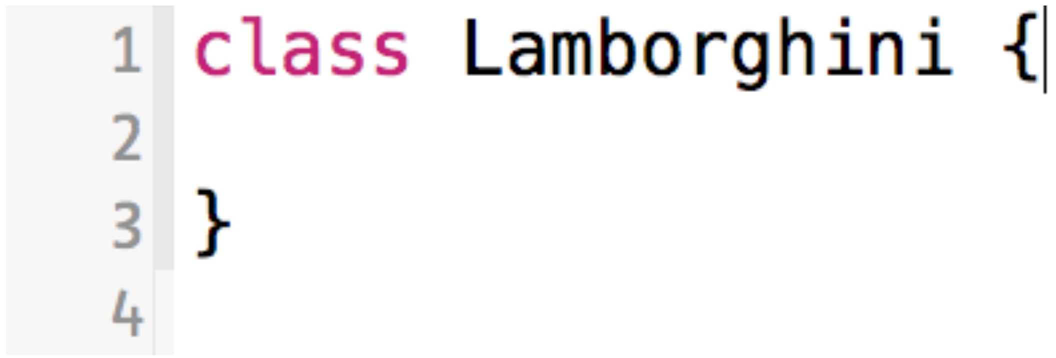
CLASSES

WHAT IS A CLASS?

- Classes are blueprints of software constructs you want to build.
    - A car can be considered to be a class.
    - A Lamborghini can be seen to be a type of car
    - A Diablo can be seen to be a type of Lamborghini, which is a type of car.
        - (Think Inheritance)

- You use classes to define an outline of what your software construct is and what it can do.

WHAT DOES A CLASS LOOK LIKE:



```
class Lamborghini {

}

```

WHAT'S INSIDE OF A CLASS?

- Properties
  - Constants and variables that describe the class
- Functions
  - Actions the function can perform with properties or other values.
  - Inside of Classes, functions are called methods.
    - Every class has one or more initialization methods that allow you to set some initial values.

  - To Playgrounds for examples on Classes and another new concept, Objects!

WHAT ARE OBJECTS?

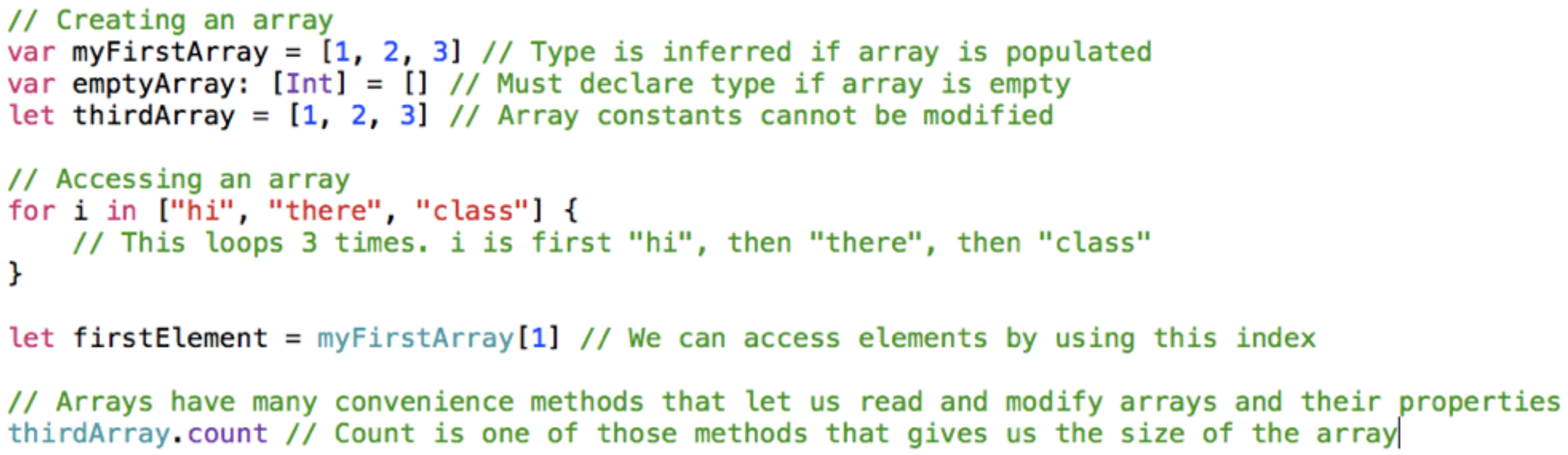
- Objects are instances of classes.
  - They allow us to take the blueprints and customize them to our needs:
    - An Aventador object is made from a Lamborghini class
      - The Lamborghini Aventador object then takes the various properties in the Lamborghini class and gives them values:
        - Yellow color
        - No spoiler
        - Black leather interior
        - etc.

ARRAYS

ARRAYS

- Arrays have a few interesting properties
  - They contain things (we’ll call them **elements**)
    - Arrays can also be empty
  - Each element has an index
    - Indexes start at 0
  - The array has a **count** of elements
  - Arrays have order, can be iterated over in order
  - Looking up element by index is **fast**

ARRAYS SYNTAX



```swift
// Creating an array
var myFirstArray = [1, 2, 3] // Type is inferred if array is populated
var emptyArray: [Int] = [] // Must declare type if array is empty
let thirdArray = [1, 2, 3] // Array constants cannot be modified

// Accessing an array
for i in ["hi", "there", "class"] {
    // This loops 3 times. i is first "hi", then "there", then "class"
}

let firstElement = myFirstArray[1] // We can access elements by using this index

// Arrays have many convenience methods that let us read and modify arrays and their properties
thirdArray.count // Count is one of those methods that gives us the size of the array
```

GETTING STARTED

INTERFACE BUILDER OUTLETS AND ACTIONS

WHAT IS A CLASS?

- Classes are blueprints of software constructs you want to build.
    - A car can be considered to be a class.
    - A Lamborghini can be seen to be a type of car
    - A Diablo can be seen to be a type of Lamborghini, which is a type of car.
        - (Think Inheritance)

- You use classes to define an outline of what your software construct is and what it can do.

GETTING STARTED

EXERCISE

KEY OBJECTIVE(S)

Follow directions in lesson plan view controllers to create a sprint match between two players.

TIMING

40-50 min	1. Code with partner
5 min	2. Debrief

DELIVERABLE

Work in groups - ask questions if you need help!