GENERAL ASSEMBLY

MOBILE DEVELOPMENT

LESSON 05
HOLY SH*T

Arthur Ariel Sabintsev
Lead Mobile Architect, ID.me

READY?
SET?
GIT PULL!

LESSON 04 REVIEW

WHAT DID WE LEARN IN LESSON 04?

- Operators Continued
  - Unary
  - Binary
  - Ternary
- Optionals
  - Optional Binding
  - Optional Unwrapping
- Functions

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

LEARNING OBJECTIVES

LEARNING OBJECTIVES

- Object Oriented Principles
- Classes and Structs
- Anatomy of an Xcode Project
- Tying Interface Builder into Code (FINALLY!)

OBJECT ORIENTED PRINCIPLES

WHAT ARE THE PRINCIPLES?

- 4 Principles
  - Encapsulation
  - Abstraction
  - Inheritance
  - Polymorphism

ENCAPSULATION

- Image 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.

```
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)

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.

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!

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

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:

```
1 class Lamborghini {
2
3 }
4
```

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.
  - They’re the blueprints come to life!

STRUCTS

WHAT IS A STRUCT?

- Structures are blueprints of software constructs you want to build.
  - Typically, they are used when describing small things.
    - Example: Rectangle

WHAT'S INSIDE OF A STRUCT?

- 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**.
- Memberwise Initialization
- Instances of Structs are not called Objects. They're simply called Instances.

  - To Playgrounds for examples on Structs

EXAMPLE OF A STRUCT

```
struct Rectangle {
    var x: Float
    var y: Float
    var width: Float
    var height: Float
}

/*
    Square is an instance of the Rectangle struct
    Square is created via memberwise initialization of the Rectangle
        struct, which means all the constants and variables are set
        during initialization.
*/
let square = Rectangle(x: 0.0, y: 0.0, width: 10.0, height: 10.0)
```

CLASSES VS STRUCTS

CLASSES VS. STRUCTS

- Classes
  - Mutable
  - Pass by Reference
- Structs
  - Immutable
  - Pass by Value
  - Memberwise Initialization

- To Playgrounds!

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)

Work with the Sample Project, Facts Generator

TIMING

40 min 1. Code with partner

5 min 2. Debrief

DELIVERABLE

Work in groups - ask questions if you need help!