GENERAL ASSEMBLY

IOS DEVELOPMENT WITH SWIFT

Giorgio Natili @giorgionatili, Lead UI Engineer / Mobile Developer / Agile Coach

CLASSES

AGENDA

- What is a `class`
- Functions as class `methods`
- `Stored` properties and `computed` properties
- Classes, methods and properties visibility (i.e. `internal`, `public` and `private` access modifiers)
- Pairing session
- Questions and Answers

LEARNING OBJECTIVES

- What is an object and why is relevant to development
- Define a class, its properties and methods
- Use a class in an iOS app
- Modularize the code base

FUNCTIONS REVIEW

A Function is a combination of instructions coupled together to achieve some result, it may take arguments and return a result.

- The keyword `func` is followed by the `identifier`
- The `identifier` is followed by an optional parameter list enclosed in parentheses
- The `->` sign specifies the return type of the function, if there is one (`Void` is assumed to be the default return value)

WHAT IS AN OBJECT

- An object is a location in memory having a value
- An object refers to a particular instance of a class where it can be a combination of variables, functions, and data structures

WHAT IS A CLASS

- A class is an extensible program-code-template for creating objects,
- A class provides initial values for state (member variables) and implementations of behavior (member functions, methods)
- A class represents a reusable part of code of a software

CLASSES AS REFERENCE TYPES

A reference is usually associated with a pointer, meaning that the memory address where your variable “resides” is actually holding *another* memory address which points to the actual object in a different memory location.

`class` is a reference type, meaning that if you assign an instance of the class to another variable, it will hold only the reference to the instance, not a copy.

`structs` are value types, meaning that if you assign an instance of a structure to another variable, it is actually copied to it

CLASSES

- The syntax is similar to the one of the most common languages
- A class can contains methods and `stored` or `computed` properties

```
class Person{
    var name:String = "Giorgio"
    var surname:String = "Natili"

    init() {

    }
}
```

PROPERTIES AND METHODS

- Methods ("member functions") are similar to functions, they belongs to classes or objects and usually expresses the verbs of the objects/class
- Properties are as in everyday language and technically are fields of objects/ classes with optional dedicated getter/setter routines

CLASSES – METHODS

- A method uses classes stored values to accomplish a task
- A method can optionally return a value

```
class Teacher{
    var name:String = "Giorgio", surname:String = "Natili"

    func sayHello() {

      println("Hello \(name) \(surname)")

    }
}
```

CLASSES – COMPUTED PROPERTIES

- Do not actually store a value
- Provide a getter + an optional setter to retrieve and set other properties indirectly

```
class Teacher{
    var name:String = "Giorgio", surname:String = "Natili"
    var fullname:String {
        get { return name + " " + surname}
        set(value){
            var data = split(value) {$0 == " "}
            name = data[0]
            surname = data[1]
        }
    }
}
```

CLASSES – INITIALIZERS

- Each class must have an initializer
- Strictly speaking the initializer is where the class member will be initialized *(not always true)*
- An initializer accepts parameters like a function
- A class can have multiple initializers that differ in their signature

CLASSES – MULTIPLE INITIALIZERS

```
class Animal{
    init(){ // Some code }
}

class Animal{
    init(specie:String){
        println("I am a \(specie)")
    }
    init(specie:String, gender:String){
        println("I am a \(specie), my gender is \(gender)")
    }
}
```

CLASSES – MEMBERS SIGNATURE

- By default all the members have an `internal` access level
- Other optional access levels are `public`, `private`, `final` and `static`
- The access level of a custom type affects all members of that type

THE INTERNAL LEVEL
This means internal to a product. A product is an app, framework, or any distributable compiled item.

PROTOCOLS

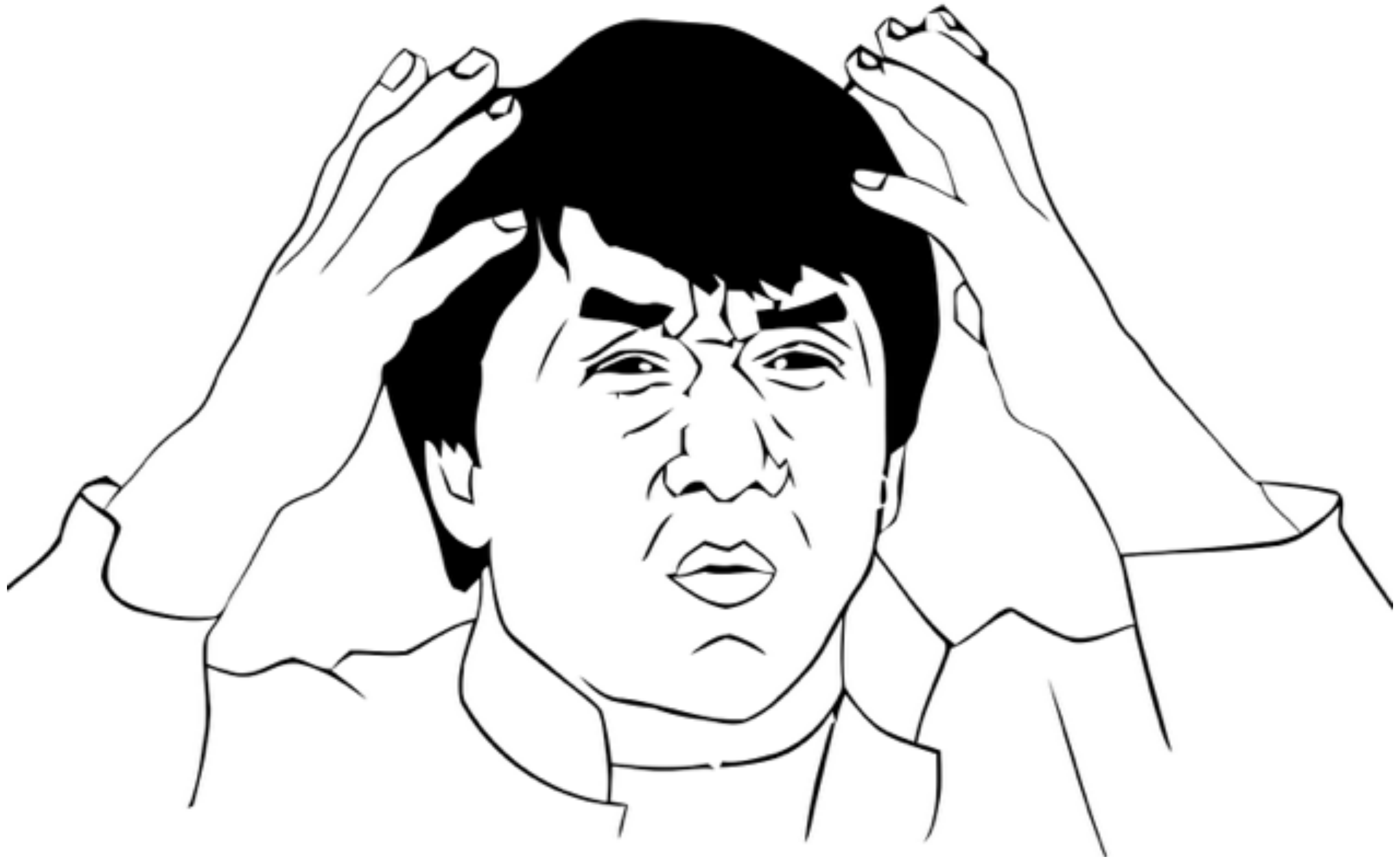
- A *protocol* is a specification that list the properties and methods an implementer has to support
- A property is specified by declaring a variable followed with a type and then either `{get}` or `{get set}`
- A method is specified using the `func` keyword followed by the function signature

```
protocol DailyGreting{
    var userName:String {get set}
    func welcome() -> String
}
```

PAIRING LAB

Imagine you work in the finance industry. Define a class to reuse across your app to add two numbers, multiply two numbers, and subtract two numbers. The class should be able to store the result of each of these operations.

HOW TO USE A CLASS IN PRACTICE



Q&A