Classes

by Ash Dreyer & Donald Pinckney

Table of Contents

Object-Oriented Programming

What is an Object?

- Name some objects in the room
- Real-world objects have a couple of characteristics
  - State
    - Ex. Dogs - breed, color, size, name, hungry
  - Behaviors
    - Ex. Dogs - barking, fetching, wagging tail
- Identify characteristics of some objects in the room

Objects in Programming

- Very conceptually similar to real-world objects
- Consist of state and behaviors
- Object stores its state in variables / constants & behaviors in functions

Classes

Relating Objects to Classes

- In the real world, there are many individual objects all of the same kind
  - A mountain bike is a bike just like a cruiser
    - Made of a lot of similar components
- In object-oriented terms, your bike is an *instance (or object)* of the *class of objects* known as bicycles
- **A *class* is the blueprint from which individual objects are created**

Creating a Class

```
class Dog {
  let name: String = “Cool dog”
  let owner: String = “Ash”
  var age: Int = 5
  var hungry: Bool = true

  func eatUntilFull() {
    hungry = false
  }

  func bark(loud: Bool = true) {
   if loud {
     print(“BARK”)
   } else {
     print(“bark”)
   }
 }

  func readTag() -> String {
    return “\(name)\nOwner: \(owner)”
  }
}
```

How to Use a Class

```
var myDog = Dog()

myDog.readTag()

if myDog.hungry {
    myDog.eatUntilFull()
}

myDog.bark(loud: myDog.age < 3)

myDog.age += 1
```

Output:
```
Cool dog
Owner: Ash
bark
```

Instances

- Concrete occurrence of any class
- A class is just a blueprint for what objects you can make with that class
  - Have to create an *instance* of the class in order to use / customize that blueprint for an individual object
- *Dot syntax* is used to access properties and functions of a class
  - instanceName.property
  - instanceName.function()

Properties

Properties

- Properties associate values with a particular class
- Properties, like any other constant or variable, must be initialized when they are created
  - 2 ways of doing this: default property values & initializers

Default Property Values

```
class Dog {
  let name = “Cool dog”
  let owner = “Ash”
  var age = 5
  var hungry = true
  // ...
}
```

- Every Dog instance will start off with these values
- It’s good to have name and owner as constants, but not every dog should be owned by Ash
  - How do we give a constant property a unique value?

Class Initialization

```
class Dog {
  let name: String
  let owner: String
  var age: Int
  var hungry: Bool

  init(dogName: String, dogOwner: String,
       dogAge: Int, isDogHungry: Bool) {
    name = dogName
    owner = dogOwner
    age = dogAge
    hungry = isDogHungry
  }

  // ...
}
```

Class Initialization - WRONG

```
class Dog {
  let name: String
  let owner: String
  var age: Int
  var hungry: Bool

  init(name: String, owner: String,
       age: Int, hungry: Bool) {
    name = name
    owner = owner
    age = age
    hungry = hungry
  }

  // ...
}
```

Class Initialization - FIXED

```
class Dog {
  let name: String
  let owner: String
  var age: Int
  var hungry: Bool

  init(name: String, owner: String,
       age: Int, hungry: Bool) {
    self.name = name
    self.owner = owner
    self.age = age
    self.hungry = hungry
  }

  // ...
}
```

Calling a Class Initializer

- When an instance of the class Dog is created, the initializer is immediately called
- As shown before, `var myDog = Dog()` creates a new instance of Dog
  - Calls an initializer, even if you don't explicitly write one
  - Only if all properties have default values!
- Initialization parameters are put in the parentheses
  - In the last case you would write…

```
var myDog = Dog(name: “Buddy”, owner: “Ash”,
                age: 7, hungry: true)
```

Default Properties vs. Initializers

- Default properties are good for properties that always start out the same or are the same for their entire use
- Initializers are good for value-dependent properties
- Ok to mix them!

Methods

Methods

- Like regular functions, except only accessible on an instance
- As said before, accessed by dot syntax
  - myDog.bark(loud: true)
  - classInstanceName.function(argumentLabel: input)
- Can automatically access instance variables