Introduction to Swift

PROG31975 – Week 1 Part 2

Jawaad Sheikh

Jawaad.Sheikh@sheridancollege.ca

Outline

- Introduction
- File Structure
- Classes
- Variables
- Control Statements
- Optionals
- Other Language Quirks
- Exercises

Introduction to Swift

- Apple introduced the Swift programming language to make creating iOS apps simpler.
- It is an evolving language, the current version is 4.0
- Note that the transition from version 2.2 to 3.0 is a huge step.
  - Many changes were made to the language.

Introduction to Swift

- The huge changes also spilled over into Objective-C in what Apple called “The Grand Renaming”
- The Grand Renaming is an undertaking from Apple to rename most / all methods and objects to “make more sense”.
- Therefore you may run into issues with rebuilding existing Objective-C and Swift projects with iOS 9.

Introduction to Swift

- This course will jump straight into Swift 4.0.
- There is a free reference book on iBooks for you to read “The Swift Programming Language (Swift 4)”.
- Now lets begin.

File Structure

- The first thing you will notice with Swift is that you are dealing with 1 source file per object.
- No header files unlike Objective-C.
- Files are a .swift file extension.
- You will find your Java & Javascript knowledge somewhat useful here.

Class Definition

- Since we have 1 file to work with, we can simplify the definition to “class”

```
class MyClass : NSObject { … }
```

Variable Definitions

- Takes a similar approach to Javascript.
- There are two keywords for variables:
  - var – is used to define variables.
  - let – is used to define constants.

```
let myConstant = 45
var myVariable = 34.4
```

Variable Definitions

- There is no need, most of the time, to define the variable type.
- But sometimes there is a need.
- You can typecast as needed.

```
let myDouble : Double = 74.3
```

Variable Definitions

- Sometimes you just need cast

```
let myLabel = “The bank balance is: “
let balance = 595

let totalMsg = myLabel + String(balance)
```

Variable Definitions

- Another way (using back slash)

```
let balance = 595

let totalMsg = “my balance is \(balance) dollars”
```

Variable Definitions

- Arrays & dictionaries are defined with square brackets

```
var nums = [ 595 , 445, 554 ]
let num = nums[1]

var occupations = [ “Jim Kirk” : “Captain”,
                    “Mr. Spock” : “First Officer” ]
occupations[“McKoy”] = “Doctor”
```

Variable Definitions

- Empty Arrays & dictionaries can be defined as:

```
let emptyArray = [String]()
let emptyDictionary = [String : Float] ()
```

Or

```
let emptyArray = []
let emptyDictionary = [:]
```

→ what just happened????

Variable Definitions

- You are going to find that there is a lot of inference of types in this language.
- Also you may have noticed that the language doesn’t need semi-colons
  - You can still use semi-colons at the end of your lines of code though.

Control Statements

- Yes, if’s and switch’s still exist in the language for conditional statements.
- For’s while’s also, but we now have for-in and repeat-while.

Control Statements - if

- No need for round brackets anymore.

```
if score > 50 {
    myScore += 1
} else {
    myScore -= 1
}
```

Control Statements - if

- Must be Boolean expression – the following will give an error:

```
if score  {
        myScore += 1
}
```

Control Statements – for-in

- Different approach to for loop

```
var total = 0
for i in 0..<4 {
    total += i
}
```

Control Statements - if

- You can combine if and let into 1 statement

```
var optionName : String? = “John Appleseed”
var greeting = “Hello!”

if let name = optionName {
    greeting = “Hello, \(name)”
```

?????

If optionName is not nil then this code will run.

Control Statements – for-in

- Useful for looping through arrays & dictionaries:

```
let numbers = [ 2, 3, 4, 5, 6 ]
var largest = 0
for number in numbers {
    if number > largest
        largest = number
}
```

Control Statements – for-in using tuples

```
let intNums = [ “Prime” : [2,3,5,7,11, 13],
        “Fibonacci” : [1, 1, 2, 3, 5, 8],
        “Square” : [1, 4, 9, 16, 25] ]
var largest = 0
for (kind, numbers) in intNums{
        for number in numbers {
                if number > largest
                        largest = number
        } } }
```

Control Statements – repeat-while

- In Java we had do-while, in Swift we have repeat-while

```
var m = 2
repeat{
    m = m * 2
} while m < 100
```

Control Statements – methods (functions)

- Similar to Javascript, we now use the ‘func’ keyword to define methods.

```
func increment() {
        x += 10
}
// called as “increment()”
```

Control Statements – methods (functions)

- Methods that return values

```
func increment() -> Int{
        x += 10
        return x
}
// called as “var y = increment()”
```

Control Statements – methods (functions)

- Methods taking in variables take a similar approach to Objective-C using labels between arguments

```
func multiply(first x:Int, second y:Int) -> Int{
        let z = x * y
        return z
}
// called as "var b = multiply(first: 5, second: 6)"
```

Part of "Grand Renaming" first arguments now have labels even in Objective-C too!

Optionals

- The ? Operator is used to represent variables where the value may be nil sometimes.
- If it isn’t used the an error will be generated when the value is nil.
- Xcode 8 makes it really easy to know when to use this operator

```
var myName : String? // nil currently
var myName : String? = “Jim Kirk”
```

Optionals

- There are 3 optional operators:
  - ? – optional, there may or may not be value
  - ! – guaranteed to be a value
  - ?? – if nil then this is the value to use.

Optionals

- Examples
  - ? – @IBOutlet weak var switch : UISwitch?
  - ! – @IBOutlet var lblName : UILabel!
  - ?? – let x = lblHelloWorld.Text ?? “hello world”

Other Language Quirks

- nil - Null is not used in this language, nil is.
- Any - This object is used to reference objects where you don’t know the type until run time.
- @IBOutlet – used to tell IB to make this variable available to connect to an object.
- @IBAction – used to tell IB to make this event handler available to connect to an object.

Exercise 1

- Create a Swift Object called Triangle containing:
  - Three lengths – a, b and c.
  - A method for calculating area and perimeter.
  - A constructor.

Exercise 2

- Create a Swift Object called Employee containing:
  - Name, age, occupation
  - A method for printing out all three to the console.
  - A constructor.

Exercise 3

- Assuming you are new to macOS:
- After exploring your new Mac’s environment try to do the following:
  - Open the console and try a few commands from your old Linux course.
  - Try to open the system settings and look at your screen resolution.
  - Try to also find out how to partition your disk (but do not attempt to do so)
  - Find out how to launch Xcode
  - Find out how to change the app store account ID to your ID so that you can download apps to your Mac

Exercise 4

- Launch Xcode and create your first iPhone app.
  - In the app, you will add a button and a label to your home page.
  - Setup an event handler to change the text of the label after clicking on the button.

Exercise 5

- Launch Xcode and create your second iPhone app.
  - In the app, you will create a simple calculator to do add, subtract, multiply and divide.
  - User alert boxes to prevent a divide by zero