Swift 4 Intro

Cody Henrichsen
September 2017

Playgrounds

- XCode's built in demo mode for Swift programming.
- Good
  - Easy to demo most aspects of Swift code
    - GUI
    - Objects
    - Methods
    - Output
- Bad
  - No keyboard input – readLine() always returns nil

Variable Basics

- Playground Level Structure
  - constancy name :Type
  - constancy name = value
  - constancy name :Type = value
- Class Structure
  - Starts with visibility level
- Basic style is REQUIRED!
  - Implicit or Explicit typing

Value Constancy

- Variable
  - Can be changed as needed
  - Keyword var
    - Regular variables
    - General programming use
- Constant
  - Only one assignment
  - Keyword let
    - Great for calculated values
    - Lambdas
    - Closures

Types

- Int
  - Integers AKA Counting numbers
  - ± 2 Billion
  - Type Constants
    - Int.max
    - Int.min
  - Type Methods
    - abs(someValue)
- Bool
  - Boolean values
- Double
  - Floating point
  - Real numbers
  - $\pm\sim 2 * 10^{308}$
  - 15 digits of precision
  - Type Constants
    - Double.pi
    - Double.infinity
    - Double.nan
  - Type Methods
    - Double.squareRoot(someValue)
    - Double.round(someValue)
    - Double.abs(someValue)

String

- New in Swift 4
  - Multiline literals with a set of triple """
    - Includes new lines, tabs, etc.
  - variableName.count for characters involved instead of variableName.characters.count
    - Café and cafe have the same length (4)
    - Even if you build the String by adding the accent with a += to the variable
    - Korean and other languages the same
  - String IS a Collection type
    - Iterable!
      - for eachLetter in someString {…}

Combining Strings

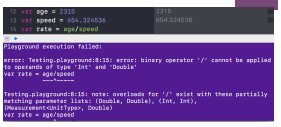
- Concatenation with + and += for other strings
  - Outside of ""
- Interpolation
  - Inside the ""
  - \(maybeNotAStringValue)
    - Variables
    - Numeric values
    - Literals
    - Method calls
    - Strings

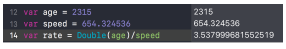
String Comparison

- Equality in composition with ==
  - Matches each character exactly
    - Includes graphemes as a single character or built character
  - Can force ignore case
    - first.caseInsensitiveCompare(second) == .orderedSame
    - But this does not work with special cases eg ß
- Alphabetic comparison with < and >

Casting

- Swift treats Int and Double as first class objects and expects the operations on them to match their Type
- So if you want to do math with mixed Type values you will need to cast them as is appropriate





Classes

- Used to build objects like Java does
- Swift's equivalent of a constructor is the initializer: init()
  - ALL data members must be initialized or set as lazy in the init or the class WILL NOT compile!
  - Overloaded init's are common
- Designated Initializer
  - Explicitly initializes all data members, usually via parameter list
  - Must have at least 1
- Convenience Initializer
  - "Helper constructor"
  - Useful for simplification patterns
  - Keyword convenience must precede init
  - MUST call a designated init

Class Initializers

```
init(named :String)
{
    self.other = named
}

convenience init()
{
    self.init(named: "adsad")
}
```

```
var instanceOfTest = Test()
var otherInstance = Test(named: "Sample init")
instanceOfTest.doesNothing()
    other "adsad"
otherInstance.doesNothing()
    other "Sample init"
```

Visibility

- Swift refers to this as Access Control
  - public
    - Like Java's public
    - Can be seen outside of the file and accessed by all instances
  - internal
    - Like Java's protected
    - Visible within the framework
    - This is the default (blank)
  - fileprivate
    - Skip me
  - private
    - Like Java's private
    - Useable only within the file

Methods

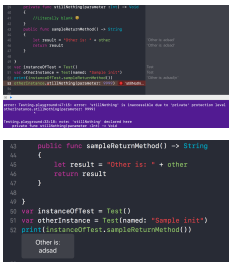
- Different from Java
  - visibility func nameOfMethod(parameter list) -> ReturnType {…}
- Methods have keywords!
  - func
    - After the visibility modifier
    - REQUIRED!!
  - ->
    - After the parameter list
    - Identifies the Type returned from the method

Method: Parameters

- Parameter list
  - Parameters are separated by commas
  - First parameter MUST have a name
  - The listing can have two forms
    - name :Type
    - externalCallName internalName : Type
  - Names in parameter list are used in the call of the method
    - Long lists look nice separated by line breaks

```
myThing.doesStuff( firstParam : 9
                   second : "sdasdas"
                   third : 32.232
                   …)
```

Methods in action



```
public func sampleReturnMethod() -> String
{
    let result = "Other is: " + other
    return result
}
}
var instanceOfTest = Test()
var otherInstance = Test(named: "Sample init")
print(instanceOfTest.sampleReturnMethod())
```

```
Other is:
adsad
```