EE443 / EE593
Mobile Application Development

Prof. Phillip De Leon

Lecture 1:
Course Introduction
Introduction to Swift

Prof. Phillip De Leon

Outline

- Syllabus and Logistics
- Introduction to Swift and iOS App Development
- Variables, Constants, Numeric Types, Strings (Chapter 21)
- Collection Types: Arrays and Dictionaries (Chapter 21)
- Optionals (Chapter 21)
- Control Statements: if-else, switch-case, for (Chapter 21)

Syllabus

EE 443 / EE 593 Mobile Application Development (3 credits)
Spring 2018
Klipsch School of Electrical and Computer Engineering
College of Engineering
New Mexico State University

Instructor and Class Information
Instructor: Prof. Phillip De Leon, Goddard Hall 310A, (575) 646-DSP1 (3771), pdeleon@nmsu.edu
Class Days and Times: Tuesday and Thursday 1:10 – 2:25pm, T&B 303
Office Hours: Monday 1:30 – 2:30pm and Thursday 9:30 – 10.30am. Office hours may be held at the Engineering Learning Communities in Engineering Complex III Room 300.
Teaching Assistant (TA): Mr. Ty Vincent, tvincent@nmsu.edu
TA Office Hours: Monday, Wednesday 8:00 – 9:00am, T&B 176

Course Description from the Undergraduate Course Catalog
Introduction to mobile application development. Students will develop applications for iOS devices including iPhone and iPad. Topics include object-oriented programming using Swift, model-view-controller (MVC) pattern, view controllers including tables and navigation, graphical user interface (GUI) design, data persistence, GPS and mapping, camera, and cloud and web services.

Prerequisite
Any object-oriented programming course such as CS271 Object Oriented Programming, CS451 C++ Programming, CS452 Java Programming, or CS454 Python Programming II.

Textbook and Other Required Materials
Required (Print or Electronic): *iOS Apprentice Sixth Edition: Beginning iOS development with Swift 4* by M. Hollemans and F. Farook (ISBN-13: 978-1942878391)

Optional: *App Development with Swift*, Apple Inc. (free download)
https://itunes.apple.com/us/book/app-development-with-swift/id1219117996?mt=11

Students will be invited to join the iOS University Program free of charge. Joining the (regular) iOS Developer Program is $99/year and optional. You can only submit to the AppStore if you are a regular iOS Developer.

Laboratory Resources
It is expected that students have their own personal Macintosh computers with Xcode. There are a limited number of Macintosh computers with Xcode in T&B 203. If you need assistance, please contact the course TA or Mr. Jerry Navarro, navarro@nmsu.edu

Online Resources
EE 443 / EE 593 Web Page
http://wordpress.nmsu.edu/pdeleon/teaching/ee443

Course announcements and student grades will be posted on Canvas
http://learn.nmsu.edu

A great website to get answers to practical coding questions
http://stackoverflow.com

Note: Please do not email Prof. De Leon through Canvas--use the email address listed above.

Course Objectives
The objective of this course is to gain an understanding of mobile application development including:

- Swift
- Model-View-Controller (MVC) pattern
- Memory management
- View controllers
- Frameworks: Foundation, CoreGraphics, CoreLocation, MapKit, UIKit, WebKit

This objective is achieved through an undergraduate- and graduate-level treatment of mobile application development.

Contribution of EE 443 / EE 593 to Meeting the Professional Component
Mobile Application Development is an undergraduate EE elective (computers) within the Electrical Engineering curriculum. Students in EE 443 will apply techniques learned in class through assigned homework, software development projects, and in-class discussions. Techniques learned in this class will provide students with a broadening of their knowledge base through application of basic mathematics and engineering science techniques to mobile application development, preparation for capstone design project, and by providing a basis for career employment or graduate school. Discussion of design issues relate the class theory to practical societal issues. Class provides 3 credits of engineering science credit.

Relationship of the Course to Program Objectives
Mobile Application Development builds upon mathematics and engineering techniques learned in previous courses to provide an electrical engineering elective to give students

- an understanding of actual products (applications for mobile devices)
- a basis for capstone design classes
- a preparation for career employment or graduate school
- an opportunity to use computers in engineering problem solving

This will allow students to further explore their major specialty as well as seeing applications of basic techniques learned from computer programming, embedded systems, signals and systems, and other engineering classes.

Topics Covered / Course Schedule
The topics covered and course schedule are described in the Course Schedule section of this syllabus.

Grading
Homework - In each chapter, students will develop and submit the tutorial application. Homework will be worth 25% of the final grade. Late homework is not accepted except in the case of an absence due to a medical or other very serious reason.

Class Lectures - Each week, students will present course material in class (lecture). Student lectures will be worth 10% of the final grade.

Mid-Term App Proposal - The mid-term app proposal (3-5 pg) will describe the mid-term app including the purpose, similar applications (if any), description, operation, and user interface (including storyboard). The mid-term app proposal is worth 5% of the final grade.

Mid-Term App - The mid-term app, developed individually by the student, is worth 25% of the final grade.

Final App Proposal - The final app proposal (3-5 pg) will describe the final app including the purpose, similar applications (if any), description, operation, and user interface (including storyboard). The final app proposal is worth 5% of the final grade.

Final App - The final app, developed individually by the student, is worth 25% of the final grade.

Final Project Presentation - There will be a final project presentation where the student will present and demonstrate their application to the class. The final project presentation is worth 5% of the final grade.

Final Grade – Final grade will be assigned as follows.

A+	>100%	C+	79 – 76%
A	100 – 95%	C	75 – 73%
A–	94 – 90%	C–	72 – 70%
B+	89 – 86%	D+	69 – 66%
B	85 – 83%	D	65 – 63%
B–	82 – 80%	D–	62 – 60%

Policies
Disputes regarding grades must be submitted in writing to Prof. De Leon for review within 7 days after graded work has been returned or posted.

As a courtesy to the instructor and fellow students, please silence your cell phones. Any student who disrupts class due to the use of an unwelcomed electronic device will be asked to leave.

Academic Misconduct
The Student Code of Conduct defines academic misconduct, non-academic misconduct and the consequences or penalties for each offence. The Student Code of Conduct is available in the NMSU Student Handbook online:
http://studenthandbook.nmsu.edu/

Academic misconduct is explained here:
http://studenthandbook.nmsu.edu/student-code-of-conduct/academic-misconduct

The College of Engineering has additional language and policies related to academic misconduct that may be found here:
https://engr.nmsu.edu/academic-integrity

Discrimination and Disability Accommodation
Section 504 of the Rehabilitation Act of 1973 and the Americans with Disabilities Act (ADA) covers issues relating to disability and accommodations. If a student has questions or needs an accommodation in the classroom (all medical information is treated confidentially), contact:
Trudy Luken, Director
Student Accessibility Services (SAS) - Corbett Center, Rm. 208
Phone: (575) 646-6840 E-mail: sas@nmsu.edu
Website: http://sas.nmsu.edu/

NMSU policy prohibits discrimination on the basis of age, ancestry, color, disability, gender identity, genetic information, national origin, race, religion, retaliation, serious medical condition, sex, sexual orientation, spousal affiliation and protected veterans status. Furthermore, Title IX prohibits sex

discrimination to include sexual misconduct: sexual violence (sexual assault, rape), sexual harassment and retaliation. For more information on discrimination issues, Title IX, Campus SaVE Act, NMSU Policy Chapter 3.25, NMSU's complaint process, or to file a complaint contact:

Lauri Millot, Title IX Coordinator or Agustin Diaz, Title IX Deputy Coordinator
Office of Institutional Equity (OIE) - O'Loughlin House, 1130 University Avenue
Phone: (575) 646-3635 E-mail: equity@nmsu.edu
Website: http://oeo.nmsu.edu

Other NMSU Resources

NMSU Police Department:	(575) 646-3311	www.nmsupolice.com
NMSU Police Victim Services:	(575) 646-3424	
NMSU Counseling Center:	(575) 646-2731	
NMSU Dean of Students:	(575) 646-1722	
For Any On-campus Emergencies:	911	

Prepared
Phillip De Leon, 17 January 2018.

Course Outline

EE 443 / EE 593 Spring 2018 Course Schedule

This schedule is an estimate of the topics covered each week throughout the course. The following chapters are from *iOS Apprentice Sixth Edition: Beginning iOS development with Swift 4* by M. Hollemans and F. Farook (ISBN-13: 978-1942878391)

Week 1 **January 14, 2018**
Intro to Swift - (DeLeon)

Week 2 **January 21, 2018**
Intro to Swift, Ch 1-3 - (DeLeon)

Week 3 **January 28, 2018**
Ch 4-6 (DeLeon)

Week 4 **February 4, 2018**
Ch 9-10 (DeLeon)

Week 5 **February 11, 2018**
Ch 11-12 (DeLeon, Undergraduate #1)

Week 6 **February 18, 2018**
Ch 13-14 (Undergraduate #2, #3)

Week 7 **February 25, 2018**
Ch 15-16 (Undergraduate #4, #5)
Mid-Term App proposal due March 1

Week 8 **March 4, 2018**
Ch 17-18 (Undergraduate #6, #7)

Week 9 **March 11, 2018**
No class (mid-term app development)
Mid-Term App due 5:00pm March 16

Week 10 ***March*** **18, 2018**
Spring Break!

Week 11 **March 25, 2018**
Ch 22-23 - (Undergraduate #8, #9)

Week 12 **April 1, 2018**
Ch 25-26 - (Graduate #1, #2)

Week 13 **April 8, 2018**
Ch 27-28 - (Graduate #3, #4)

Week 14 **April 15, 2018**
Ch 29-30 - (Graduate #5, #6)
Final App Proposal due April 19

Week 15 **April 22, 2018**
Ch 34-36 (Graduate #7, #8)

Week 16 **April 29, 2018**
No class (final app development)
Final App due 5:00pm May 4

Week 17 **May 6, 2018**
Final App Presentations
May 8, 1:00pm – 3:00pm, T&B 303
May 8, 3:30pm – 5:30pm, T&B 303
May 9, 3:30pm – 5:30pm T&B 303

Course Logistics

- Personal Mac laptops are recommended; iMacs in T&B 203 (lock code 20302)
- Student Macs: Update to OS X 10.13 “High Sierra" and install Xcode 9.2
- Accept invite to NMSU-DeLeon Apple iOS Developer "Team"
- No need at this time to join Standard Developer program $99
- Email UUID to pdeleon@nmsu.edu to provision devices

Outline

- Syllabus and Logistics
- Introduction to Swift and iOS App Development
- Variables, Constants, Numeric Types, Strings
- Collection Types: Arrays and Dictionaries
- Optionals
- Control Statements: if-else, switch-case, for

Introduction to Swift

- Swift introduced at WWDC 2014 and Swift 4 at WWDC 2017
- Swift is the result of the latest research on programming languages and strives to present a simpler syntax; takes language ideas from Objective-C, Python, Ruby, etc.
- Swift is intended to be more resilient to erroneous code, "safer" (easier to catch software bugs) and also more concise
- Built with the LLVM compiler framework included in Xcode, and uses the Objective-C runtime, allowing C, Objective-C, C++ and Swift code to run within a single program

https://developer.apple.com/swift/

Introduction to iOS

- iOS is the mobile OS created and developed by Apple Inc. and distributed exclusively for Apple's iPhone and iPad
- Q4 2017 46.7M, 10.3M iPhones, iPads sold
- Over 1 billion active iOS devices (Jan. 2016)
- USA smartphone subscribers Android 53.3% vs. iOS 44.9% (Jun. 2017)[1]
- Apple's App Store contains more than 2.2 million iOS applications and 25+ billion downloads (2016)[2]

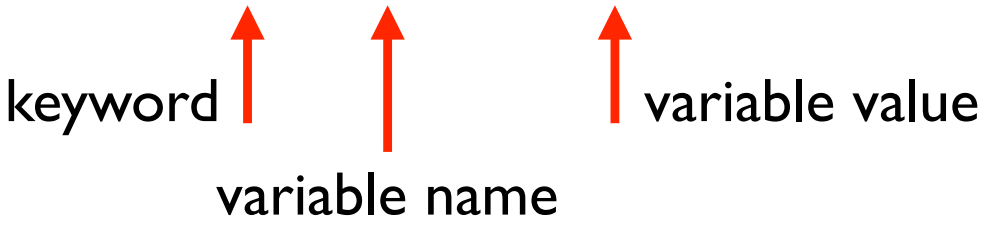
(1) https://www.statista.com/statistics/266572/market-share-held-by-smartphone-platforms-in-the-united-states/
(2) http://www.businessofapps.com/data/app-statistics/

Outline

- Syllabus and Logistics
- Introduction to Swift and iOS App Development
- Variables, Constants, Numeric Types, Strings
- Collection Types: Arrays and Dictionaries
- Optionals
- Control Statements: if-else, switch-case, for

Variables

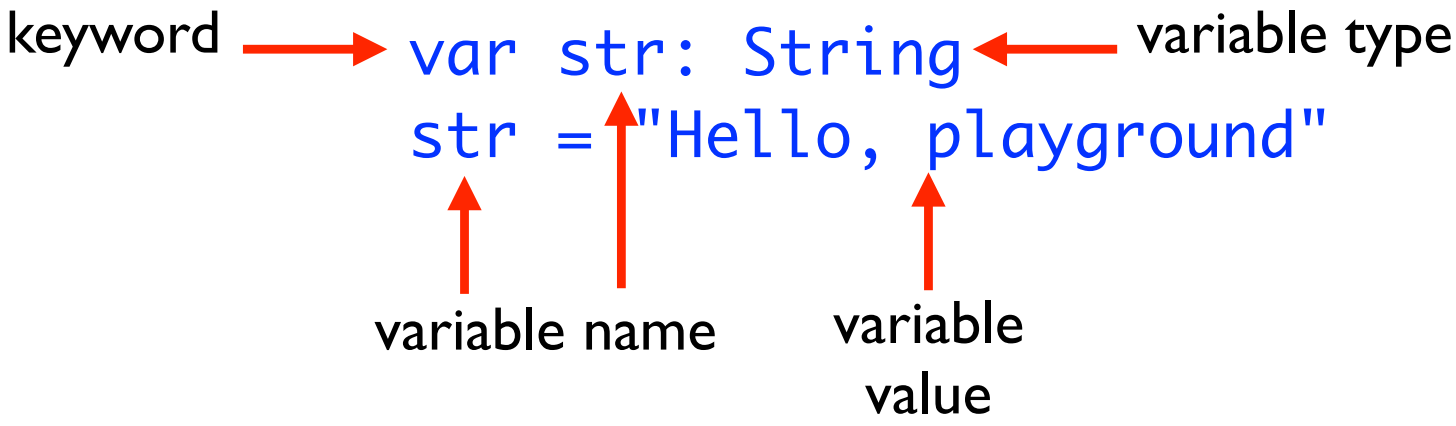
```
var str = "Hello, playground"
```



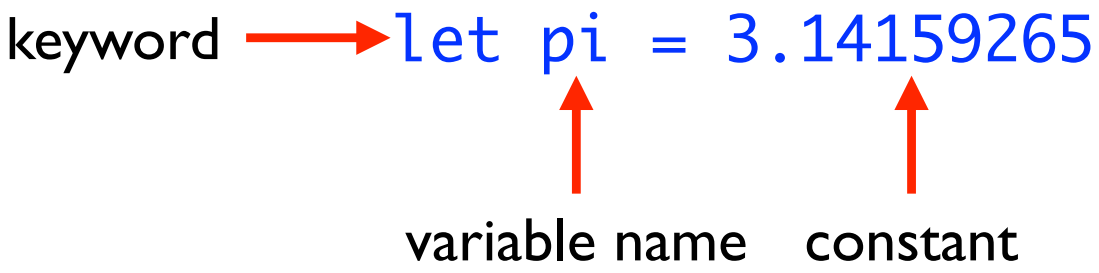
- Swift infers the variable type to be a String because of initialization
- No semicolon ";" needed to end the line

Variables (cont.)

- If we do not initialize variable, must declare variable type
- Swift objects can never be nil



Constants



- Swift infers type of constant
- Variables are mutable and constants are immutable

Numeric Types

- Numeric Types: Int, Uint, Float, and Double

Table A-1. Integer Types

Type	Size in Bits	Maximum Value	Minimum Value
`Int`	32 or 64	As `Int32` or `Int64`	As `Int32` or `Int64`
`UInt`	32 or 64	As `UInt32` or `UInt64`	0
`Int64`	64	9,223,372,036,854,775,807	–9,223,372,036,854,775,808
`UInt64`	64	18,446,744,073,709,551,615	0
`Int32`	32	2,147,483,647	–2,147,483,648
`UInt32`	32	4,294,967,295	0
`Int16`	16	32767	–32768
`Uint16`	16	65535	0
`Int8`	8	127	–128
`UInt8`	8	255	0

Numeric Types (cont.)

```
let a = 123
let b = 0.456
let c = Double(a) + b
```

- Swift does not provide implicit type conversion when you assign a variable (or expression) of one numeric type to a variable of another numeric type
- The expression Double(a) invokes an initializer of the Double type with an integer argument

Numeric Types (cont.)

Table A-2. Predefined Binary Arithmetic Operators

Operator	Meaning
<<	Bitwise left shift
>>	Bitwise right shift
, &	Multiplication
/, &/	Division
%, &%	Remainder
&	Bitwise AND
+, &+	Addition
-, &-	Subtraction
|	Bitwise OR
^	Bitwise XOR

Table A-3. Predefined Unary Arithmetic Operators

Operator	Meaning
++	Pre- or post-increment
--	Pre- or post-decrement
+	Unary plus
-	Unary minus
~	Bitwise NOT

- Integer overflows are trapped in Swift
- Programmers can choose to allow overflows by using "&"

Strings

- Any Unicode character can be embedded in a string

```
let atSigns = "@\u{40}" // "@@"
```

- Strings have the ability to interpolate expressions enclosed in the escape sequence \()

```
let r = 3.0
print("Area of a circle of radius \(r) is \(M_PI * r * r)")
```

- The "+" operator can be used to concatenate strings

```
let s = "That's one small step for man, " + "one giant leap for mankind"
```

Outline

- Syllabus and Logistics
- Introduction to Swift and iOS App Development
- Variables, Constants, Numeric Types, Strings
- Collection Types: Arrays and Dictionaries
- Optionals
- Control Statements: if-else, switch-case, for

Arrays

- Ordered collection of items, zero-indexed, and typed
- If an array is created using a let statement, neither it nor its contents can be modified in any way (immutable)
- Examples of array creation and initialization

```
var integers = [1, 2, 3]
var integers: [Int]; integers = [1, 2, 3]
var days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
            "Saturday"]
let weekendDays: [String] = ["Saturday", "Sunday"]
let weekDays = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"]
let allDays = weekendDays + weekDays
var batterySizeNames: [String] = []
batterySizeNames += ["AA", "AAA"]
```

Arrays (cont.)

- Examples of array element access, new value assignment, and element removal

```
var favoriteBooks = ["Learn C in 10,000 Hours",
   "Javascript: The Mediocre Parts",
   "Haskell for Dummies"]
favoriteBooks.count
favoriteBooks.first
favoriteBooks.last
favoriteBooks[1]
```

```
integers[0]
integers[2]
integers[1..<3]
days[3]
days[3] = "WEDNESDAY"
integers.append(4)
integers.insert(-1, at: 0)
days.remove(at: 3)
days.removeSubrange(0..<4)
```

Dictionaries

- Data structure mapping instances of a key (usually a string) to a corresponding value (usually an object)
- Elements are not ordered and keys must be unique

```
var dict = ["Red": 0, "Green": 1, "Blue": 2]
let value = dict["Green"]
dict["Yellow"] = 3
dict.removeValue(forKey: "Red")
dict.count
```

Outline

- Syllabus and Logistics
- Introduction to Swift and iOS App Development
- Variables, Constants, Numeric Types, Strings
- Collection Types: Arrays and Dictionaries
- Optionals
- Control Statements: if-else, switch-case, for

Optionals

- By default Swift objects can never be nil (no value)

```
var x: Int
var y = x + 1 // compiler error!
```

- What happens when you try to use an empty reference? ... crash! ... throw an exception (probably crash)! ... silently do nothing at all!
- Sometimes it's useful to have a variable that can have no value, in which case you need to declare it as an optional

How Swift Deals with nil

- Optionals "?" allow variables to be empty (nil)

```
var x: Int? // x is an "Optional Int" that contains nothing
x = 1       // x contains the value 1
x = nil     // x contains nothing
```

- An optional "?" can be made of any Swift type:

```
var i: Int?
var s: String?
var f: Float?
```

- An optional is like a box that might hold a value and Swift syntax forces you to safely deal with nil

Forced Unwrapping

- You can forcibly unwrap optionals with the "!" operator

```
var x: Int? = 3
let y: Int = x! // y now has the value 3
```

- "Forcibly": the app will crash if the optional didn't have a value

```
var x: Int? = nil
let z: Int = x! // this will crash!
```

- If you are 100% sure an optional has a value, you can forcibly unwrap
- How to safely deal with nil?

Determining if an Optional Contains a Value

- Compare optionals with nil then if they have a value, unwrap

```
// assuming optionalInt is an Int?
if optionalInt != nil {
    // it's safe to unwrap
    let x: Int = optionalInt!
    print("optionalInt has a value, and that value is \(x)")
} else {
    print("optionalInt does not have a value")
}
```

Optional Binding (Preferred)

- Optional Binding is a shortcut to check and unwrap an optional

```
// assuming optionalInt is an Int?
if let x: Int = optionalInt {
    print("optionalInt has a value, and that value is \(x)")
} else {
    print("optionalInt does not have a value")
}
```

See also "Optional Chaining"

Nil Coalescence

- The nil-coalescing operator "??" unwraps its left operand, if it's not nil and returns its value; otherwise it returns its second operand

```
let defaultColorName = "red"
var userDefinedColorName: String? // defaults to nil

...
// maybe userDefinedColorName gets a value, maybe not
...

var colorNameToUse = userDefinedColorName ?? defaultColorName
```

Outline

- Syllabus and Logistics
- Introduction to Swift and iOS App Development
- Variables, Constants, Numeric Types, Strings
- Collection Types: Arrays and Dictionaries
- Optionals
- Control Statements: if-else, switch-case, for

if-else

```
var count = 0
var text: String
if count == 0 {
    text = "No Items"
} else if count == 1 {
    text = "1 Item"
} else {
    text = "\(count) Items"
}
```

Switch

- Use the switch statement to select a code path based on one of several possible values of a variable or expression

```
var value: UInt = 11
switch value {
case 2, 3, 5, 7, 11, 13, 17, 19:
    print("Count is prime and less than 20")
case 20...30:
    print("Count is between 20 and 30")
default:
    print("Greater than 30")
}
```

- A case can have multiple possible values
- Case value can be a range
- Control does not pass from one case into another. That means that the example will execute only one case and print only once
- Switch expression does not have to be numeric, i.e. could switch on the value of a string

for

```
for i in 0..<10 {
    print(i)
}
```

Iteration over a sequence of values, i.e. range or collection

```
for i in stride(from: 10, through: 0, by: -2) {
    print(i)
}
```

More general ranges with stride

```
for i in stride(from: 10, to: 0, by: -2) {
    print(i)
}
```

for

```
let base = 3
let power = 10
var answer = 1
for _ in 1...power {
    answer *= base
}
print("\(base) to the power of \(power) is \(answer)")
```

If you don't need the loop value, you can ignore by using an underscore in place of a loop variable

for

```
let names = ["Anna", "Alex", "Brian", "Jack"]
for name in names {
    print("Hello, \(name)!")
}

let numberOfLegs = ["spider": 8, "ant": 6, "cat": 4]
for (animalName, legCount) in numberOfLegs {
    print("\(animalName)s have \(legCount) legs")
}

var dict = ["Red": 0, "Green": 1, "Blue": 2]
for (key, value) in dict {
    print("\(key) -> \(value)")
}
```

- Iterate over keys of a dictionary by using its keys property,
- Iteration order is undefined since dictionaries do not provide any ordering

while & repeat-while

- A while loop starts by evaluating a single condition. If the condition is true, a set of statements is repeated until the condition becomes false.

```
while condition {
    statements
}
```

- A repeat-while loop, performs a single pass through the loop block first, before considering the loop's condition. It then continues to repeat the loop until the condition is false.

```
repeat {
    statements
} while condition
```

Control Transfer Statements

(study outside of class)

- continue
- break
- fallthrough
- return
- throw
- guard