CSC 581: Mobile App Development

Spring 2018

Unit 3: Optionals & segues

- Swift
  - optionals, guards
  - scope, enumerations
- UIKit
  - segues
  - navigation controllers

Optionals

recall: Swift uses optionals when a value may or may not be present

```
var ages = ["Pat": 18, "Chris": 20, "Kelly": 19]
print(ages["Chris"])              → Optional(20)

outputLabel.text = "Hello world"
print(outputLabel.text)           → Optional("Hello world")
```

- you can unwrap an optional using ! or (more elegantly) if-let

```
print(ages["Chris"]!)             → 20

if let msg = outputLabel.text {
    print(msg)                    → Hello world
}
```

every type in Swift has a corresponding optional type (with ? at end)

```
var age: Integer? = ages["Chris"]

var msg: String? = outputLabel.text
```

Intentional optionals

structs/classes can have optional fields; functions can return optionals

```
struct Profile {
    var name: String
    var age: Int?
    var favoriteMovie: String?

    func getAge() -> Int? {
        return age
    }
}

var person = Profile(name: "Sean", age: 22, favoriteMovie: "Life of Pi")
if let age = person.getAge() {
    print("\(person.name) is \(age) years old.")
}
else {
    print("\(person.name)'s age is unknown")
}

person = Profile(name: "Dave", age: nil,
                 favoriteMovie: "Big Trouble in Little China")
if let age = person.getAge() {
    print("\(person.name) is \(age) years old.")
}
else {
    print("\(person.name)'s age is unknown")
}
```

3

Guards

when defining functions, will often have an initial validity test or tests

```
func average(_ nums: [Double]) -> Double {
    if nums.count > 0 {
        var sum = 0.0
        for n in nums {
            sum += n
        }
        return sum/nums.count
    }
    else {
        return 0.0
    }
}


func display(for person: Profile) {
    if let age = person.age {
        if let fav = person.favoriteMovie {
            print("\(person.name) (age \(age)) loves \(fav)")
        }
    }
}
```

4

Guards

a guard is a cleaner notation for handling validity tests

```
guard condition else {  }
```

```
func average(_ nums: [Double]) -> Double {
    guard nums.count > 0 else { return 0.0 }

    var sum = 0.0
    for n in nums {
        sum += n
    }
    return sum/nums.count
    }
}
```

```
func display(for person: Profile) {
    guard let age = person.age, let fav = person.favoriteMovie else { return }

    print("\(person.name) (age \(age)) loves \(fav)")
}
```

5

Variable scope

as in most languages, a block (code enclosed in { }) defines a new scope for variables

- note: a function defines a new scope as well as control statements

```
var x = 100
if true {
    var y = 3
    print("\(x) \(y)")      → 100 3
}
print(x)                    → 100
print(y)                    → ERROR: Use of unresolved identifier 'y'
```

variable shadowing

- unlike Java, you can override an existing variable inside a new scope

```
var x = 100
if true {
    var x = 3
    print(x)                → 3
}
print(x)                    → 100
```

6

Enumerations

as in Java, you can define a new type by enumerating all of the possible values of that type

- useful when you want a limited range of values to be assignable

```
enum Answer {
    case yes, no, maybe
}

enum DayOfWeek {
    case Monday, Tuesday, Wednesday, Thursday,
         Friday, Saturday, Sunday
}

var response = Answer.yes

if (response == .yes || response == .no) {    // Swift infers the
    print("That's a decisive answer")         // enum type
}

var today = DayOfWeek.Tuesday
print(today)
```

7

Segues

a view controller manages a screen within an app

- if an app is to have multiple screens
  - ✓ need a view controller for each screen
  - ✓ need to define how to transition from one view controller to another

a segue defines a transition between view controllers

- segues are defined in Interface Builder by connecting the controllers
- can select the presentation method for the transition
- navigation controllers allow you to go back or jump to other screens

complete Lesson 3.6

- create an app with multiple views, define segues between them
- final version has two destination buttons, an enable/disable switch

HW3a: complete the lab at the end of 3.6

- create an app with a login screen

8