Lecture 2: Create login form using Swift

ADVANCED MOBILE DEVELOPMENT (iOS)

Lecturer: Christine Jiang

Senior Mobile Application Developer at IBM

Content

- Objective C syntax
- Xcode structure
- Xcode shortcut
- Create a login page using Swift
- Swift syntax

Method Syntax

```
-(void)entrolWithCourse:(NSString *)courseName
```

- for instance method.

Call syntax: [self entrolWithCourse:@"iOS"];

```
+(CGFloat)RedValue;
```

+ for class method.

Call syntax: [PaintView RedValue];

Method Syntax

```
-(void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event
```

Nothing return

```
- (NSString*)searchLocation:(NSString *)keyword;
```

The return type is NSString

Method Syntax

```
-(void)enrolWithCourseName:(NSString*)name courseCode:(NSString*)courseCode
```

1st part of the method name

2nd part of the method name

Full name is enrolWithCourseName:courseCode:

Method Syntax

name of 1st argument

```
-(void)registerWithName:(NSString *)name withDate:(NSDate*)date
```

name of 2nd argument

Method Syntax

Line up colons when there are lots of arguments

```
-(void)registerWithName:(NSString *)Name
            withStartDate:(NSDate*)startDate
              withEndDate:(NSDate*)endDate
        withCampusLocation:(NSString*)location
```

Always make sure your code is readable :-)

Instance Methods

- Starts with a dash

```
-(BOOL)registerWithName:(NSString *)name
          withCourse:(NSString*)course
      withCampus:(NSString*)campus
{
                              …
}
```

- Example calling syntax:

```
[self registerWithName:@"Charles"
            withCourse:@“iOS programming"
                withCampus:@"Unitec"];
```

Method Syntax

- Use IBAction to alert Interface Builder of an action.

```
- (IBAction)redPressed:(id)sender;          //We normally use this format.
- (IBAction)redPressed:(UIButton*)sender;   //But we also can try this.
- (IBAction)redPressed:sender;              //same as (id) sender version.
- (IBAction)redPressed;                     //You can ignore sender as well.
```

Scope

By default, the instance variable are @protected

@protected: only the class and subclasses can access

@private: only the class/yourself can access

@public: anyone can access

Scope Example

```
@interface MyProfile : NSObject
{
    NSDate *enrolmentDate;

    @public
    NSString *name;

    @private
    NSString *address;

    @protected
    NSArray *courses;

    @public
    NSString *hobby;
}
```

Question:
How many public variables, private variables and protected variables?

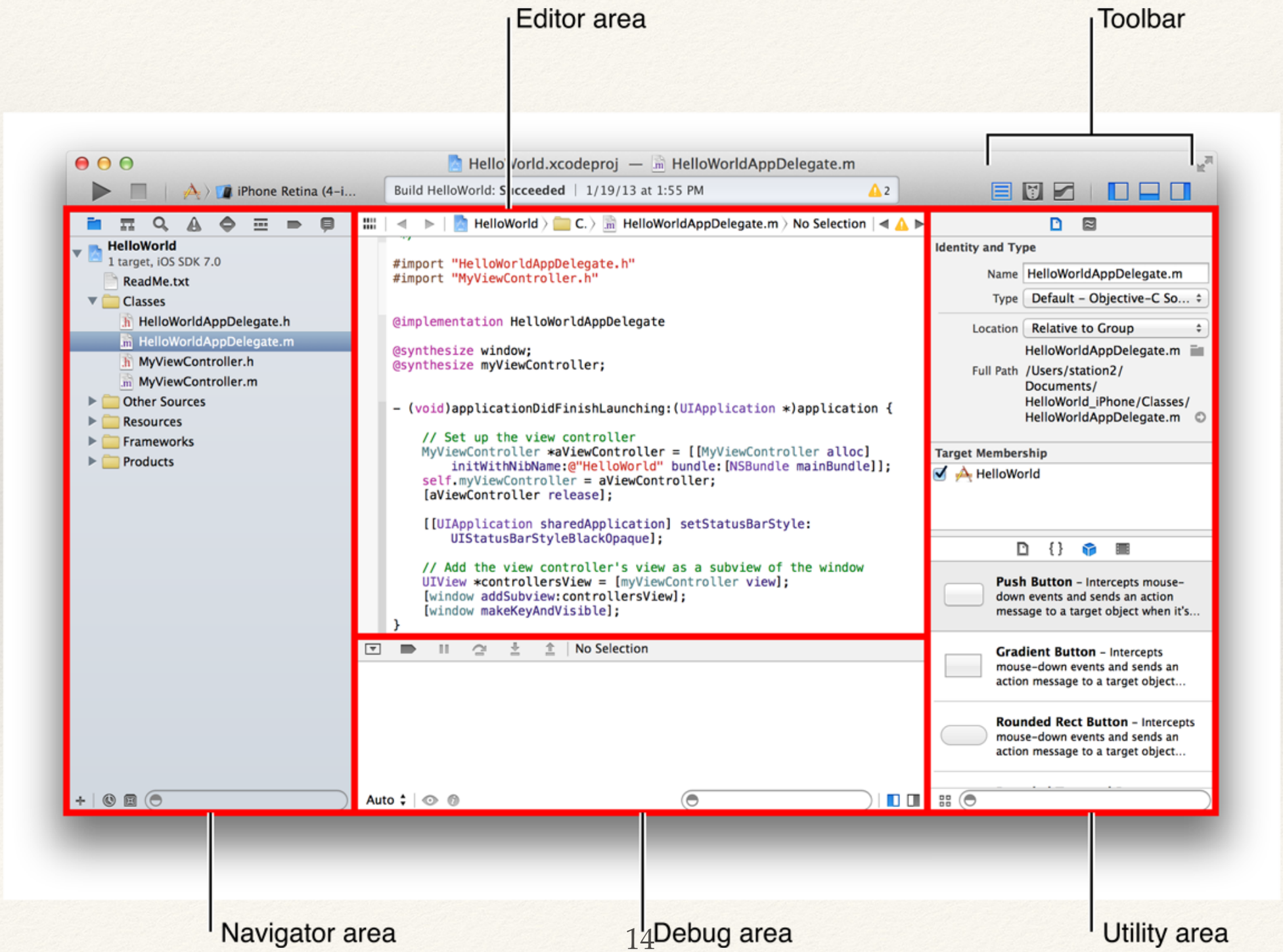
Scope Example

Answer:
Protected: enrolmentDate & courses
Private: address
Public: name & hobby

Xcode

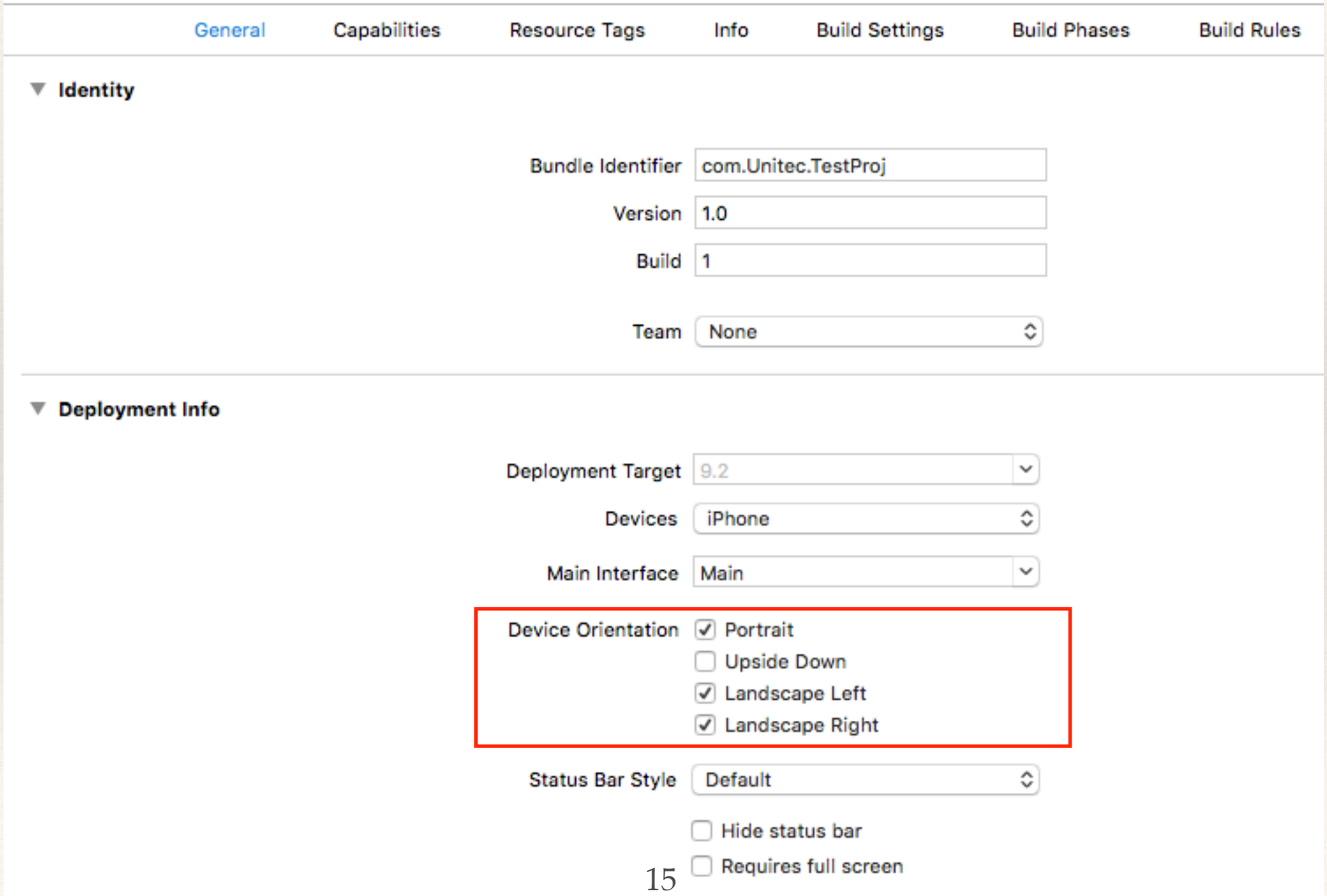
- Getting around Xcode's Workspace Windows
- Xcode breaks up its main workspace window into several sub-areas. When describing keyboard shortcuts, it is informative to keep these areas in mind:
  - The Editor – where most coding work is done.
  - The Navigator – where a developer can navigate through tests, project files, compiler errors, breakpoints, etc.
  - The Debug Area – used during debugging.
  - The Utility Area – quick help, file and data descriptions, etc.

Xcode


Editor area
Toolbar
HelloWorld.xcodeproj — HelloWorldAppDelegate.m
iPhone Retina (4-i...
Build HelloWorld: Succeeded | 1/19/13 at 1:55 PM
HelloWorld
1 target, iOS SDK 7.0
ReadMe.txt
Classes
HelloWorldAppDelegate.h
HelloWorldAppDelegate.m
MyViewController.h
MyViewController.m
Other Sources
Resources
Frameworks
Products
HelloWorld 〉 C. 〉 HelloWorldAppDelegate.m 〉 No Selection
#import "HelloWorldAppDelegate.h"
#import "MyViewController.h"
@implementation HelloWorldAppDelegate
@synthesize window;
@synthesize myViewController;
- (void)applicationDidFinishLaunching:(UIApplication *)application {
// Set up the view controller
MyViewController *aViewController = [[MyViewController alloc]
initWithNibName:@"HelloWorld" bundle:[NSBundle mainBundle]];
self.myViewController = aViewController;
[aViewController release];
[[UIApplication sharedApplication] setStatusBarStyle:
UIStatusBarStyleBlackOpaque];
// Add the view controller's view as a subview of the window
UIView *controllersView = [myViewController view];
[window addSubview:controllersView];
[window makeKeyAndVisible];
}
No Selection
Auto
Identity and Type
Name HelloWorldAppDelegate.m
Type Default – Objective-C So...
Location Relative to Group
HelloWorldAppDelegate.m
Full Path /Users/station2/Documents/HelloWorld_iPhone/Classes/HelloWorldAppDelegate.m
Target Membership
HelloWorld
Push Button – Intercepts mouse-down events and sends an action message to a target object when it's...
Gradient Button – Intercepts mouse-down events and sends an action message to a target object...
Rounded Rect Button – Intercepts mouse-down events and sends an action message to a target object...
Navigator area
Debug area
Utility area

Device Orientation

- Select your project file -> Select your target -> Select General tab



Xcode Shortcut

⌘ = Command

⇧ = Shift

⌥ = Option / Alt

^ = Control

←··→ = Left / Right Arrow Keys

↑↓ = Up / Down Arrow Keys

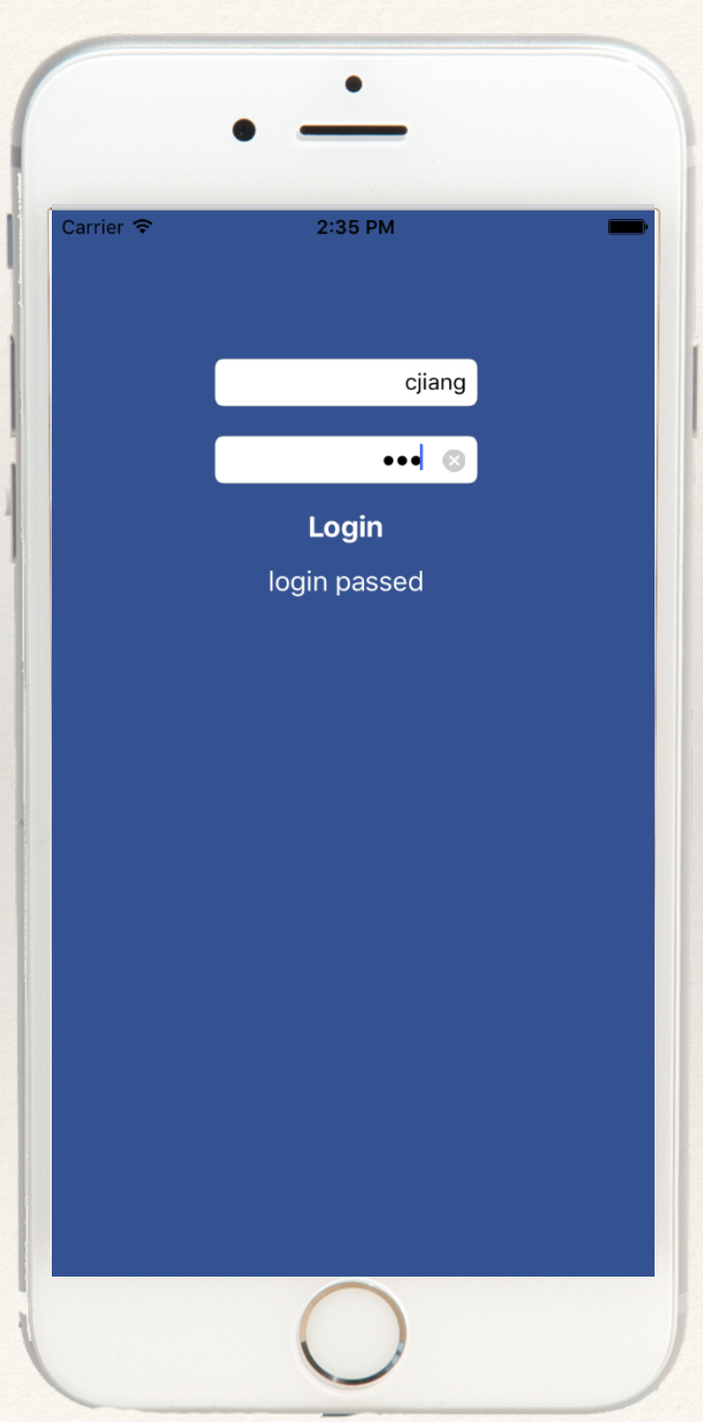
Xcode Shortcut

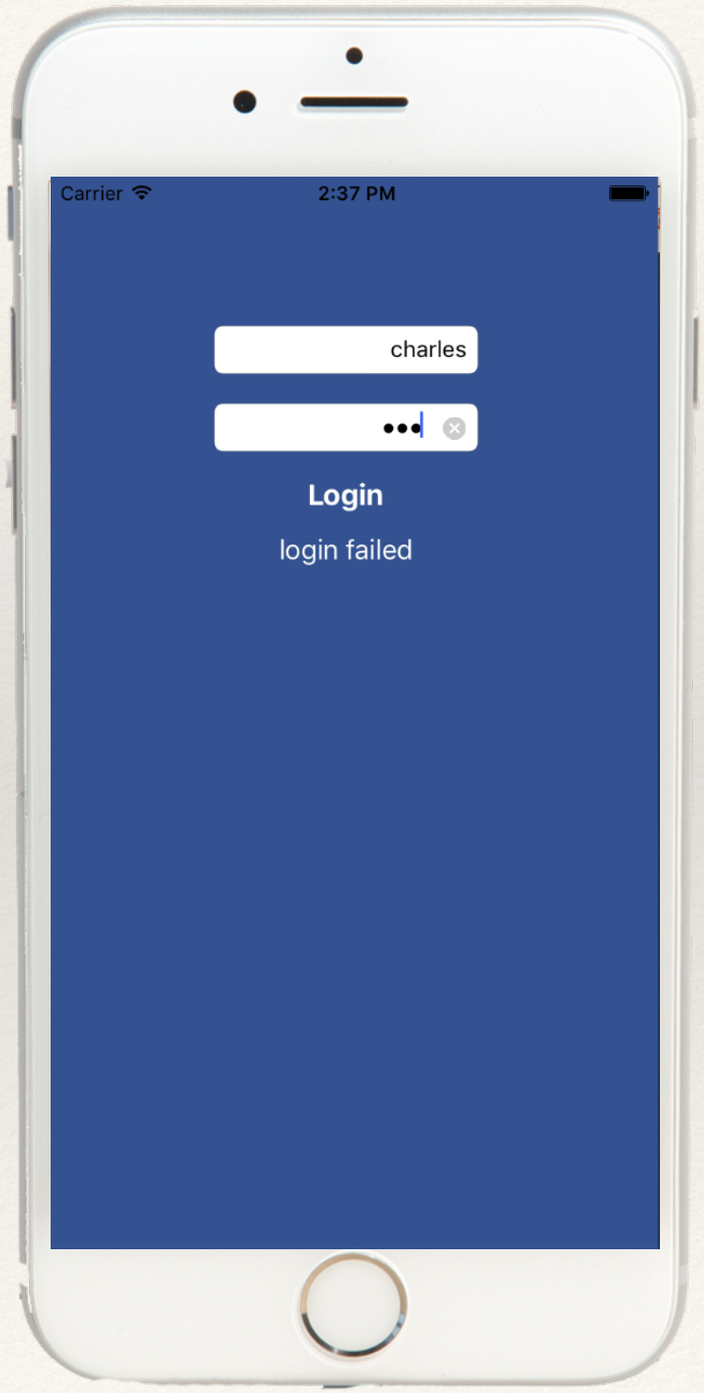
http://www.1729.us/xcode/Xcode%20Shortcuts.png

Task

- Create login application using Swift
- Authentication succeeds if you enter your user name, otherwise it fails.

Login Page


Carrier
2:35 PM
cjiang
Login
login passed


Carrier
2:37 PM
charles
Login
login failed

Swift

```
@IBOutlet weak var userNameTextField: UITextField!
@IBOutlet weak var passwordTextField: UITextField!
@IBOutlet weak var resultLabel: UILabel!

  @IBAction func buttonDidTap(_ sender: Any)
  {
    let userName = self.userNameTextField.text?.uppercased()
    let password = self.passwordTextField.text

    let isEqualName = (userName == "CJIANG")
    let isEqualPwd = (password == "IBM")

    if isEqualName && isEqualPwd
    {
      self.resultLabel.text = "login succeed"
    }
    else
    {
      self.resultLabel.text = "login failed"
    }
}
```

Compare Strings

Review:
How to do this using Objective-C?

Compare Strings

Question:
How to do this using Swift?

Swift

```
let x = "hello"
let y = “hello world"
let isEqual = (x == y)

Use == operator
```

Swift

- Let: defines a constant
- Var: defines a variable

```
let maximumNumberOfLoginAttempts = 10

var currentLoginAttempt = 0
```

Swift

```
var variableString = "Apple"

variableString += " and Banana"

// variableString is now "Apple and Banana"
```

```
let constantString = "Apple"

constantString += " and another Banana"

// this reports a compile-time error - a constant string cannot be modified
```

Swift

- You can declare multiple constants or multiple variables on a single line, separated by commas:

```
var x = 0.0, y = 0.0, z = 0.0
```

- If a stored value in your code is not going to change, always declare it as a constant with the let keyword.
- Use variables only for storing values that need to be able to change.

Type Annotations

```
var welcomeMessage:String

welcomeMessage = “Hello”

var red, green, blue: Double
```

Naming Constants and Variables

- Constant and variable names can contain almost any character, including Unicode characters:

```
let π = 3.14159
let 你好 = "你好世界"
let 🐶🐮 = "dogcow"
```

Printing Constants and Variables

```
print(friendlyWelcome)

print("The current value of friendlyWelcome is \(friendlyWelcome)")
```

Semicolons

- Unlike many other languages, Swift does not require you to write a semicolon (;) after each statement in your code, although you can do so if you wish. However, semicolons *are* required if you want to write multiple separate statements on a single line:

```
let cat = "🐱"; print(cat)
// Prints "🐱"
```

Extension

```
extension String {
  var banana : String {
    let shortName = String(characters.dropFirst(1))
    return "\(self) \(self) Bo B\(shortName) Banana Fana Fo F\(shortName)"
  }
}

let bananaName = "Jimmy".banana
print(bananaName)
```

* What's output?

Swift ? !

- When working with optional values, you can write ? before operations like methods, properties. If the value before the ? is nil, everything after the ? is ignored and the value of the whole expression is nil. Otherwise, the optional value is unwrapped, and everything after the ? acts on the unwrapped value. In both cases, the value of the whole expression is an optional value.
- Once you're sure that the optional does contain a value, you can access its underlying value by adding an exclamation mark (!) to the end of the optional's name. The exclamation mark effectively says, "I know that this optional definitely has a value; please use it." This is known as forced unwrapping of the optional's value (...)