



UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad
Homework Guide

Students should be careful to preserve their project at each step along the way by always making a copy of their current project before starting the next basic assignment and before attempting the challenges. Your file system folder structure for this class may look something like:

- Intro to iOS Programming
 - Week 1
 - Quiz.xcodeproj
 - Week2
 - SwiftConvrxns_Wk2.xcodeproj
 - SwiftConvrxns_Wk2Challenge1.xcodeproj
 - Week 3
 - SwiftConvrxns_Wk3.xcodeproj

Savvy students will program in an extensible manner, but since we use the same essential sample application throughout the five weeks, implementing the challenges or new assignments on the same codebase may result in extra work since I will ask you to dramatically change the architecture of your project at different points along the way. It is also good to have “versions” of your project at each step along the way so that you have something to refer to while you are still acclimating yourself to iOS Development.

HOMEWORK SUBMISSION:

You must make at least two forum entries per week.

Although I do assign homework each week, I will not collect it every week. UCI requires that I have 2 concrete grading criteria and so you must submit screenshots for two (2) of the assignments no later than one week after the last lecture. I will notify you which assignments must be submitted

You may choose which challenges to do for credit, but I suggest that you try all of them and do at least the basic assignment each week for your own good and not fall behind as we build upon each previous week's homework project.

**PLEASE SUBMIT ALL HOMEWORK TO IPHONE@SMILEFISH.COM
AND INCLUDE YOUR NAME AND THE ASSIGNMENT IN THE SUBJECT
LINE OF YOUR EMAIL.**



Copyright © 2014 Smilefish Corporation. All Rights Reserved.
This material was written and developed by Justin-Nicholas Toyama.



UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad
Homework Guide

Week 2: The View Hierarchy

- **Topics covered:**

- Model-View-Controller (MVC) Architecture (this week we focus on the “V” and the “C” – the model is really integrated into the Controller in this example)
- **In-class:** I will show the Homepwner application that I have written in Swift as an example to learn the following topics. I will also show the C-F converter so that students can use it to kickstart their homework project.
 - Use UITabBarController to present different views over the same data or different functions of the app
 - Create a view controller hierarchy (i.e. navigation stack) using UINavigationController
 - Create a view hierarchy with Interface Builder (IB)
 - Create a view hierarchy programmatically – add the image view programmatically
 - Implementation preparation
 - the tab bar in Homepwner will contain three table view controllers:
 - 1) default table view cells with a text only list of items
 - 2) custom table view cells containing the images of the items only and a navigation controller to view a larger version of the image in a UIScrollView
 - 3) the combined custom table view cell that comes from a separate NIB including the text and pictures and a navigation controller to show the item details with a custom-drawn “tally” of how many times the item has



Copyright © 2014 Smilefish Corporation. All Rights Reserved.
This material was written and developed by Justin-Nicholas Toyama.



UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad Homework Guide

been opened (up to 20 tally marks, then will just show numbers since tallies are too hard to count)

Homework:

- **Reading (optional): Big Nerd Ranch Ch. 4, 5, 6**
- **Coding:** Students will create a converter application that contains a C-F converter (given), distance converter (mi <-> km), and a volumetric converter (liter <-> gallon), which will allow students to understand UITabBarController, UINavigationController and UIView
- **Implementation Instructions:**
 - Create a new application and name it SwiftConvrxns (read as “Swift Conversions”)
 - Implement a UITabBarController to with three tabs, each containing a view controller with one of the following three converters (use NIBs to create your views – do not use a storyboard!)
 - **°C - °F converter** - This will be provided to you in Swift (download from the class homepage under Week 2). Your assignment is to place this View Controller inside a Navigation Controller and have a second view controller in the stack that shows an image of the Celsius and Fahrenheit curves).
 - **Distance converter** (mi <-> km) - You should be able to tweak the code in the C-F converter to do a different conversion calculation.
 - **Hint:** 1 km = 0.62137 miles (or visit <http://www.mile-to-km.com> for the algebra if needed)
 - **Volumetric converter** (liter <-> gallon)
 - **How to accomplish this (next page):**





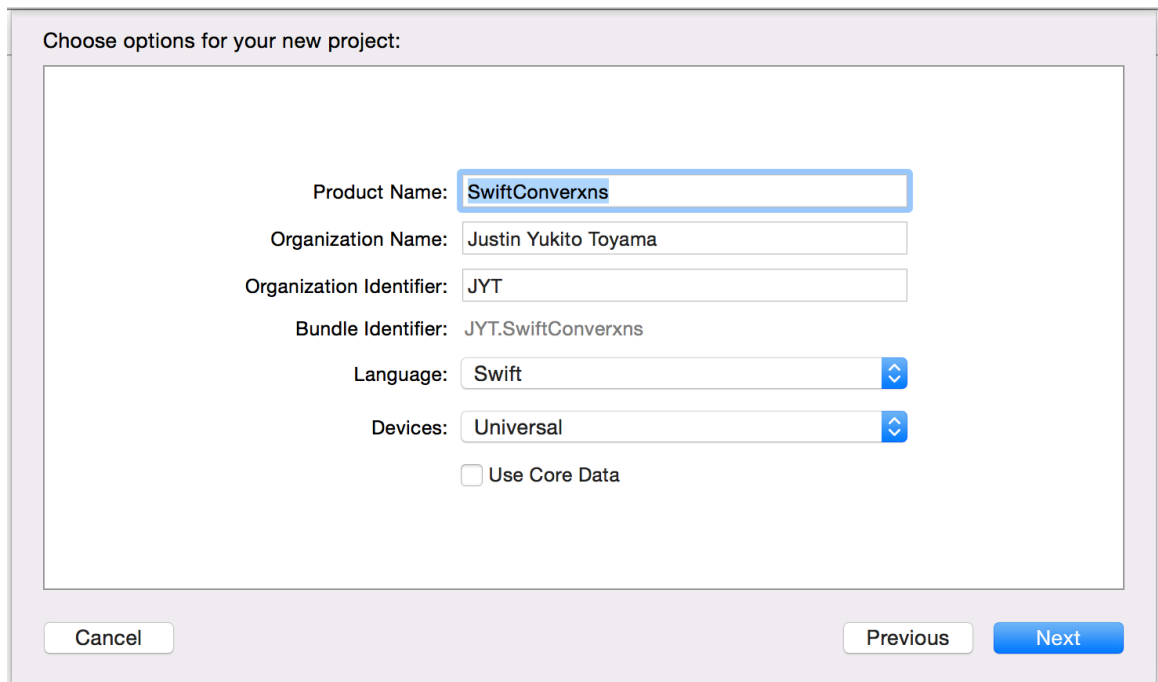
UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad
Homework Guide

1. Create a new project.

- Open Xcode.
- From the dialog that appears, select 'Create a new Xcode project'.
- From the dialog that follows, select the 'Single View Application' template and click 'Next'.
- Name your application as follows and click 'Next' (use your name and initials). Make sure to select Swift as the programming language.



The image shows the 'Choose options for your new project' dialog box in Xcode. The dialog has a title bar that says 'Choose options for your new project:'. Inside, there are several input fields and dropdown menus. The 'Product Name' field is highlighted with a blue border and contains the text 'SwiftConverxns'. Below it, the 'Organization Name' field contains 'Justin Yukito Toyama', and the 'Organization Identifier' field contains 'JYT'. The 'Bundle Identifier' field contains 'JYT.SwiftConverxns'. The 'Language' dropdown menu is set to 'Swift', and the 'Devices' dropdown menu is set to 'Universal'. At the bottom, there is a checkbox labeled 'Use Core Data' which is currently unchecked. At the very bottom of the dialog, there are three buttons: 'Cancel', 'Previous', and 'Next'.

- Select the location to which you wish to save your project and click 'Create'.
- Download the UCIFahrenheitConverter app from the class homepage.
- Find the view controller file called `TemperatureConverterViewController.swift` and add it to your project





UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad Homework Guide

by clicking the 'File' menu, selecting 'Add Files to "SwiftConverxns"...' and then selecting the TemperatureConverterViewController.swift file.

- h. Take a minute to look at the code in this file. If you don't understand what it does, run the UIFahrenheitConverter app. It is very straightforward and running it should give you an idea of what the code does.

2. **Create the View Hierarchy.** Your first task will be to setup your UITabBarController and UINavigationController instances. Do this in the application:didFinishLaunchingWithOptions: callback function of your Application Delegate class. (Every project has an AppDelegate.swift file.)

- a. I find it easier to start at the destination and work backwards. This also helps us see what we have to provide as parameters and then we just create them to suit our needs by adding lines of code above what we've written to that point.

Recall that our view hierarchy begins with the application window. Let's get a pointer to the window, and then add a tab bar controller to the window. We will fill in the contents of the tab bar controller in the steps that follow. Add the following lines of code inside of the method in AppDelegate.swift:

```
self.window = UIWindow(frame: UIScreen.mainScreen().bounds)
self.window?.rootViewController = tabBarController
self.window?.addSubview(tabBarController.view)
self.window?.makeKeyAndVisible()
```





UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad
Homework Guide

- b. We added a `UITabBarController` instance to our window and set it as the window's `rootViewController`. But for our application to be able to do this, we first have to actually create the `UITabBarController` instance. Add the following line of code **above** what you have typed so far. Note that our tab bar controller itself will not change, so we make it immutable with the `let` keyword.

```
let tabBarController = UITabBarController()
```

- c. Next, we need to populate the `@property viewControllers` of `tabBarController` to create our tabs remembering to house each of our converter view controllers inside of a `UINavigationController` so that we get the `UINavigationController` with the "screen title" and the back buttons, etc. Add the following line of code **below** the line of code you just added in the previous step. You will get some errors, but we will fix those in the following steps.

```
tabBarController.viewControllers = [firstNavController]
```

- d. Again, before we do this, we have to create the instance. Add the following code **above** everything you have so far:

```
let firstNavController = UINavigationController(rootViewController:
temperatureConverterViewController)

firstNavController.tabBarItem.title = "Temperature"
```





UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad Homework Guide

- e. After step “2d”, you will get some errors because you are trying to use the view controller before it has been created. To fix the errors, we need to create the converter view controller instances that we are setting as the rootViewController for each of the navigation controllers we just created. Take a look at

`TemperatureConverterViewController.swift`. You will need an initializer method so that we can create an instance of this class. We will be loading our view controller’s view from a NIB (.xib), so we will use the `-initWithNibName:bundle:` flavor of “init” functions. The compiler will remind us that we must also implement the required `-initWithCoder:` method. Let’s add those now, remembering to call super’s implementation for any init method on the `UIViewController` class:

```
override init(nibName nibNameOrNil: String?, bundle
nibNameOrNil: NSBundle?) {

    super.init(nibName: nibNameOrNil, bundle: nibBundleOrNil)

}

required init(coder aDecoder: NSCoder) {

    super.init(coder: aDecoder)

}
```

- f. Remember that ViewControllers own their Views. Now that we have a way to initialize our UIViewController subclasses (in this case, an instance of `TemperatureConverterViewController.swift`) and provide its view, let’s practice doing that inside of our application delegate (`AppDelegate.swift`). Pay close attention as you will have to do this yourself in the following steps without help. Inside of the





UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad
Homework Guide

`application:didFinishLaunchingWithOptions:` function in `AppDelegate.swift`, add the following lines of code above everything you have written so far in this function:

```
let temperatureConverterViewController = TemperatureConverterViewController (nibName:  
"TemperatureConverter", bundle: NSBundle.mainBundle())
```

- g. At this point, your `AppDelegate.swift` file should look appear as follows on the next page.



Copyright © 2014 Smilefish Corporation. All Rights Reserved.
This material was written and developed by Justin-Nicholas Toyama.



UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad
Homework Guide

```
//
// AppDelegate.swift
// SwiftConverxns
//
// Created by justin on 2015/01/13.
// Copyright (c) 2015年 Justin Yukito Toyama. All rights reserved.
//

import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?


    func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool {
        // Override point for customization after application launch.

        let temperatureConverterViewController = TemperatureConverterViewController
(nibName: "TemperatureConverter", bundle: NSBundle.mainBundle())

        let firstNavController = UINavigationController(rootViewController:
temperatureConverterViewController)

        firstNavController.tabBarItem.title = "Temperature"

        let tabBarController = UITabBarController()

        tabBarController.viewControllers = [firstNavController]

        self.window = UIWindow(frame: UIScreen.mainScreen().bounds)

        self.window?.rootViewController = tabBarController

        self.window?.addSubview(tabBarController.view)

        self.window?.makeKeyAndVisible()

        return true
    }
}
```

Note: If you have some extra code from the project template, you can delete it so that your file appears exactly as above (or leave it – it won't hurt).



Copyright © 2014 Smilefish Corporation. All Rights Reserved.
This material was written and developed by Justin-Nicholas Toyama.



UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad
Homework Guide

- h. Go ahead and Build the project to see if you have typos or other errors by pressing Command+b. If you run the project at this point (Command+r), it will run and then crash shortly thereafter because it cannot find the nib we specified. We will fix this in step three (next page).



Copyright © 2014 Smilefish Corporation. All Rights Reserved.
This material was written and developed by Justin-Nicholas Toyama.



UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad
Homework Guide

3. **Create the UI.** Now that we have setup our AppDelegate to create our view hierarchy and put it on screen, we need to create the actual UI for the view controllers whose views we want to display.
- From the 'File' menu, select 'New' > 'File...'.
b. Navigate to the iOS – User Interface section in the wizard dialog that appears.
c. Select 'View' and click 'Next'.
d. Name the file 'TemperatureConverter.xib'
e. Once the file is created, it should be open for you in Interface Builder. Add the UILabels and UITextFields to you View so that your UI looks like this (Look at the Week 1 Lecture video if you don't remember how to do this):

A screenshot of a mobile application interface. It features two labels, 'Fahrenheit:' and 'Celsius:', each followed by a text input field. The input fields are empty and have a light gray border. The labels are in a sans-serif font. The entire interface is enclosed in a thin gray border, with a small status bar icon in the top right corner.



UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad Homework Guide

- f. Now, select the File's Owner object, open the Identity Inspector and set `TemperatureConverterViewController` as the view controller class that owns this view. (Hint: If you don't remember how to do this, look at slide 11 from the Lecture Notes for Week 2.)
- g. Next, open the Assistant Editor. You should see `TemperatureConverterViewController.swift` on the right hand side.
- h. Control-drag from the UITextField on top to a blank spot near the top of your class. Name this text field `fahrenheitTextField`.
- i. Repeat the previous step for the other text field and name it `celsiusTextField`.

Your class should now look like this:

```
1  //
2  //  TemperatureConverterViewController.swift
3  //  UCISwiftConverter
4  //
5  //  Created by justin on 2015/01/13.
6  //  Copyright (c) 2015年 Justin. All rights reserved.
7  //
8
9  import UIKit
10
11 class TemperatureConverterViewController: UIViewController, UITextFieldDelegate {
12
13     @IBOutlet weak var fahrenheitTextField: UITextField!
14
15     @IBOutlet weak var celsiusTextField: UITextField!
16
17 }
```

- j. Next, in Interface Builder, Control-drag from each of the text fields to the File's Owner object and select 'delegate' to set our converter class as the delegate of each text field. This will let you take advantage of the code you have been given so that the conversion will be made when the user enters a value and hits the return key on the keyboard via the





UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad Homework Guide

- textField:ShouldReturn: method of the UITextFieldDelegate protocol.
- k. Control-drag from the File's Owner object to the main 'View' in your UI. (Your application will crash if you forget to do this! Try without first and take a look at the error you get in your console.)
 - l. Now we need to guarantee that our UI will look good on all our device screens. Create constraints by selecting the top text field, and then from the 'Editor' menu, select 'Resolve Auto Layout Issue' and select 'Add Missing Constraints' from the 'All Views In View' section.
 - m. Now that we have created the NIB (XIB) the application was looking for, we should be able to run the application. Select the iPhone 6 simulator and run the application by clicking the 'Run' button in Xcode or pressing Command+r. You should see your application running with a tab bar onscreen (with one tab) and your Temperature converter inside that tab. Go ahead and try to do a conversion in the simulator.
 - n. It worked! But there is a little problem, the Navigation bar is covering our UI! Let's fix this now.
 - i. Go back to interface Builder and select the File's Owner object, then select the main 'View'.
 - ii. From the Attributes Inspector, select 'Translucent Navigation Bar' of the 'Top Bar' attribute and select 'Translucent Tab Bar' for the 'Bottom Bar' attribute in the 'Simulated Metrics' section at the top.
 - iii. Next drag all your subview down so that they are visible below the navigation bar. Then, select one of the labels or text fields and from the 'Editor' menu, select 'Resolve Auto Layout Issue' and select





UCI Extension

I&C SCI X402.37

Mobile Development for Apple iPhone and iPad Homework Guide

‘Update Constraints’ from the ‘All Views In View’ section to update the constraints of all your subviews.

- o. Run the app again. Everything should work as expected and fit nicely on screen.

4. **Now do it on your own.** Following the instructions from Steps 1 and 2, create a distance converter controller and a volumetric converter controller, the XIBs for their UI, and add them to the tab bar controller in your AppDelegate. Adjust your UI to make sense for the type of converter. (Hint: duplicate the TemperatureConverterViewController.swift file and rename it. Don’t forget to rename the actual class inside your swift file. The class name and the filename are separate even though they may look the same!) Also note that the formulas you will need to use for these different converters are on the first page of the Week 2 assignment in this guide.

- **Challenge 1:**
 - o Create another view controller and xib to hold an image of the Fahrenheit-Celsius curves (provided) and push it on to the navigation stack inside your `firstNavController`. You can accomplish this by adding a button to your UI and creating an action for that button that will create an instance of your new view controller class and push it onto the navigation stack. (Hint: remember, we can control-drag to create actions the same way we create referencing outlets).
- **Challenge 2:**
 - o Put the C-F curve diagram into a zoomable, scrollable interface using UIScrollView (see the lecture notes and my sample project.)
- **Challenge 3:**
 - o Create one or more extra converter view controllers and allow users to swap the tabs using the SDK-provided “More” tab
- **SUBMISSION REQUIREMENT: SUBMIT A SCREENSHOT OF YOUR APP WITH THE 3 CONVERTERS IN THE TAB BAR AND THE VOLUMETRIC CONVERTER SELECTED AND ON SCREEN. PLEASE FOLLOW THE INSTRUCTIONS ON PAGE 1 OF THIS GUIDE WHEN EMAILING YOUR HOMEWORK. THANK YOU AND GOOD LUCK!**

