In-Class Homework #5 – iOS UITableViews

CUS 357 – Winter 2017
Due Date: End of class, Mar 15, 2017

Learning Objectives

- Become familiar with UITableViews and its delegates in iOS.

Please work in pairs to get as much of this assignment done as possible by the end of today's class session.

Overview

Begin this exercise using the finished product of your In-Class Homework #2 solution (e.g. your stylized iOS GeoCalculator). If you did not complete that assignment, you are welcome to use the instructor's solution (available on Blackboard with this assignment) as your starting point.

Using what you have learned in Lecture 11 you will be using UITableView and UITableViewController to add a "history" feature to your GeoCalculator app as shown in Figure 1. To help you get this done in 50 minutes, we will guide you through the coding in a step-by-step fashion.

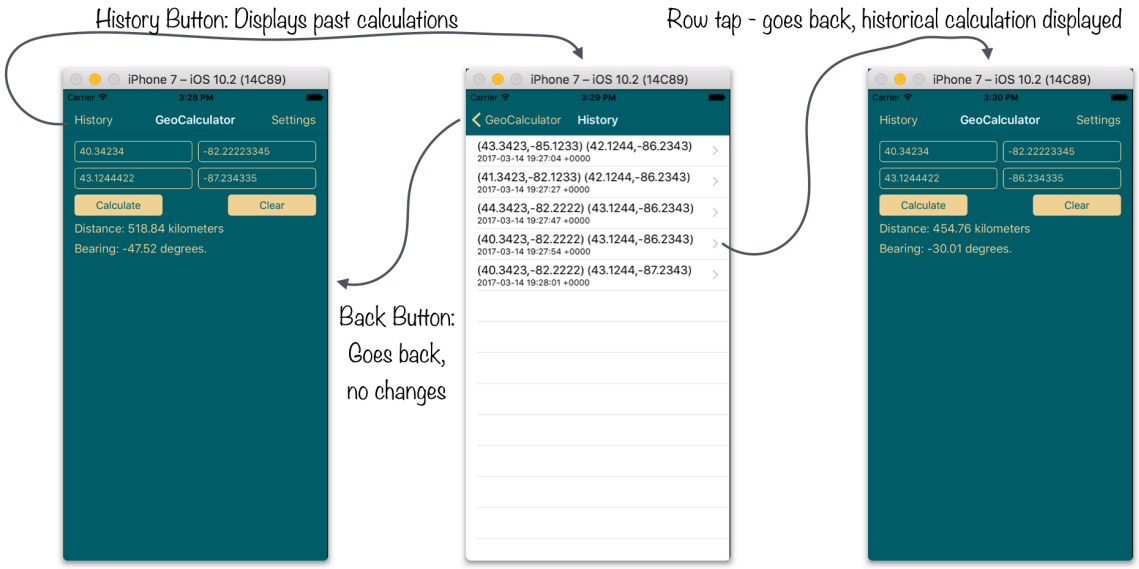

Figure 1. Adding a "history" feature via UITableView.

Each time the calculate button is pressed with valid inputs, a time stamped entry will be recorded for historical purposes. When the user taps the history button (left bar button) a UITableView populated with all of the history to-date will be displayed. If the user hits the back button, the history view controller is popped, and the user is returned to the calculator screen with no changes. However, if the user taps a cell in the history table, the history view controller is popped, and the user is returned to the calculator screen with that historical calculation repopulated in the fields and the distance and bearing computed and displayed below.

Introducing the Model

Recall, iOS uses the MVC pattern. You need to create a simple data structure to be able to hold a list of historical calculations.

Step 1: Introduce a new struct to your project called LocationLookup (code should go in LocationLookup.swift):

```
import Foundation

struct LocationLookup {
    var origLat:Double
    var origLng:Double
    var destLat:Double
    var destLng:Double
    var timestamp:Date

    init(origLat:Double, origLng:Double, destLat:Double, destLng:Double,
timestamp:Date) {
        self.origLat = origLat
        self.origLng = origLng
        self.destLat = destLat
        self.destLng = destLng
        self.timestamp = timestamp
    }
}
```

Step 2: Add a property to your calculator view controller to hold an array of LocationLookup instances:

```
var entries : [LocationLookup] = []
```

Step 3: Add a single line of code to your calculate button's IBAction to store a new instance to the array whenever it is pressed, and valid inputs are provided:

```
entries.append(LocationLookup(origLat: p1lt, origLng: p1ln, destLat: p2lt,
                    destLng: p2ln, timestamp: Date()))
```

Updating the Storyboard

In your main.storyboard file, a number of changes are in order:

Step 4: Note from Figure 1 that your calculator screen is now also managed by a UINavigationController. Select your calculator screen and embed it in a UINavigationController (review your notes for a refresher here if necessary – we looked at two different ways to accomplish this in the past.)

Step 5: Recall that previously, only the setting screen was in a UINavigationController in the past, but you don't need that nav controller anymore, so select it and delete it.

Step 6: Whoops! At this point, your setting screen is just hanging out all by its lonesome self, as the previous step eliminated the segue from the settings button on the calculator screen. No problem, we don't want the settings button either, so delete it.

Step 7. Drag **Bar Button Item**s from the object library to both the left and right side of your calculator screen as shown in Figure 1 above. Label the left as "History" and the right as "Settings".

Step 8: Add a segue from the settings button over to the settings screen and give it the identifier "settingsSegue".

Step 9: Now we need a new scene for our history table view. Drag a **Table View Controller** from the object library out onto your Interface Builder canvas.

Step 10: You will have one blank table view cell prototype in the scene's table view. Select it (you may need to use the structured list view on the left side of your IB editor to select it) and then in the Attribute Inspector set its style to Subtitle, and its identifier to "cellId". While you are at it, don't forget to set the cell's "Accessory" setting to "Disclosure Indicator".

Step 11: Create a segue from the "History" button to the new table view controller scene and give it identifier "historySegue".

Implementing the `HistoryTableViewController` class.

Now you need to implement a new view controller for the scene you just added. Recall that this controller will implement both the delegate and the dataSource for the tableView.

Step 12: Create a new class called HistoryTableViewController that extends UITableViewController. Store the code for the class in HistoryTableViewController.swift.

Step 13: Don't forget to go back into the storyboard, select the table view controller scene you added previously, and in the Identity Editor set the custom class to HistoryTableViewController. Note that because you dragged an complete table view controller assembly out from the library, you don't actually have to manually setup the delegate and dataSource like the instructor did in Lecture 11. Bonus!!

Step 14: Just like you did in your calculator screen, add a property called entries to your HistoryTableViewController that stores an array of LocationLookup instances

```
var entries : [LocationLookup] = []
```

Somehow… when you segue into the history controller you will need to establish the value of this property based on the history the calculator scene has retained to-date.

Step 15: Update the prepareForSegue method on your calculator to check for a segue with id "historySegue". Add the code that assigns the calculator's entries to the HistoryTableViewController's entries when this segue occurs.

Step 16: Now that the HistoryTableViewController's entries variable is getting set on segue, go ahead and implement the three dataSource methods discussed in Lecture 11. When displaying the lat/long data in the table, go ahead and round to 4 decimal places:

```
    // MARK: - Table view data source
    override func numberOfSections(in tableView: UITableView) -> Int {
        // your code goes here
```

```
    }

    override func tableView(_ tableView: UITableView, numberOfRowsInSection section:
Int) -> Int {
        // your code goes here
    }


    override func tableView(_ tableView: UITableView, cellForRowAt indexPath:
IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "cellId", for:
indexPath)

       // your code goes here.

        return cell
    }
```

Step 17: Isn't this exciting? Before you get too carried away, this would be a good time to run your app and clarify your history entries are getting displayed properly. Don't forget to do a few calculations before attempting to display the history, to make sure it has some data to work with. You still don't have the code that re-displays a tapped historical entry, but the back button should allow you to get back to the calculator from the history.

Displaying Historical Entries

You are almost done. One last feature is to get the history controller to "pop" itself and get the calculator to redisplay the historical calculation. You can accomplish this by using the tried and true delegation pattern, and a little bit of help from the UITableViewDelegate protocol.

Step 18: Introduce a new protocol that HistoryTableViewController will use to delegate back the historical entry when one of its rows is tapped (you can define this right in HistoricalTableViewController.swift):

```
protocol HistoryTableViewControllerDelegate {
    func selectEntry(entry: LocationLookup)
}
```

Step 19: Have your calculator controller implement this protocol. Shouldn't be too hard – just stuff the four values in the appropriate fields and cause your calculate logic to run!

Step 20: Define a property on HistoryTableViewController to hold a reference to its "historyDelegate". Make sure you update your calculator's prepareForSegue to properly set this property to itself when seguing to history.

```
var historyDelegate:HistoryTableViewControllerDelegate?
```

Step 21: You need to implement the UITableViewDelegate's tableView:didSelectRowAt method:

```
    override func tableView(_ tableView: UITableView, didSelectRowAt indexPath:
IndexPath) {
        // use the historyDelegate to report back entry selected to the calculator
scene
        if let del = self.historyDelegate {
            let ll = entries[indexPath.row]
            del.selectEntry(entry: ll)
        }
```

```
        // this pops to the calculator
        _ = self.navigationController?.popViewController(animated: true)
    }
```

Those annoying little details…

Before you attempt running and experiencing GeoCalculating nirvana, you might need to make a little fix to your original SettingsViewController. If you dismissed it by calling dismiss on the view controller when the save/cancel buttons were tapped, that's not going to work anymore! You need to replace the dismiss calls with code that "pops" the settings scene from the navigation controller's stack. But of course, you already know how to do that from the previous step!

```
_ = self.navigationController?.popViewController(animated: true)
```

If you did everything correctly, you should now be done! Do a quick demo for the instructor, and then have one of the pair post a zipped solution (or github url) to Blackboard. If you do not finish, post the project to Blackboard with a sentence or two description of what you finished.