**Milestone 4 - a Contact List**

***Xcode Project "ContactList"***

Create a new Xcode project (don't use a previous project!) consisting of a universal iOS **Master-Detail Application** called **ContactList** (make sure that "Use Core Data" is **not** selected). When you set up the project, make sure the Company Identifier is set to **au.edu.griffith.ict.prog** and Devices is set to **Universal**. After clicking **Next**, select the *Desktop* folder to create your project in and make sure you select **Create local git repository for thisprojec**t, before clicking **Create**.

Make sure you regularly commit any changes you make to your project using the **Source Control / Commit** menu item in Xcode!  Briefly document each change!

***Tests and Model classes***

First, download the PersonTests.swift file attached to this assignment and add it to the ContactListTests target in your project created by Xcode. *For the moment, these tests should fail!*

**Person class**

Copy and add your Person.swift from Milestone 2 into the project (or simply create a new Person class).  To make the tests work again, you need to modify your Person class so that it stores a person's birth year in an optionalyearOfBirth integer property (i.e., Int?).  Also, you need to change the age property so that it now becomes a computed property whose getter derives the age from the yearOfBirth and the current year and whose setter calculates and stores the yearOfBirth from the current year and the person's age.

Hint: you can get the current year by fetching the current date using NSDate and then extracting the year component using NSCalendar, e.g.:

            let todaysDate = NSDate()

            let calendar =  NSCalendar.currentCalendar()

            let currentYear = calendar.component(.Year, fromDate: todaysDate)

For compatibility to the old API, make the computed age property an implicitly unwrapped optional (Int!) that only will be nil if the yearOfBirth is nil (and vice versa)!  Then create a designated initialiser that now takes an optionalyearOfBirth instead of a person's age, i.e.:

    init(firstName: String, lastName: String, yearOfBirth: Int? = nil, middleName: String? = nil)

Change your original intialiser to become a convenience initialiser, i.e.:

    convenience init(firstName: String, lastName: String, age: Int, middleName: String? = nil)

If you implemented everything correctly, **the PersonTests.swift tests will succeed now!**

**ContactListEntry subclass**

Create a ContactListEntry subclass of your Person class that contains two more optional String properties, address and phoneNumber.  Download the ContactListsEntryTests.swift file attached to this assignment and add it to yourPersonTests target (at this stage, testing will fail).  Create a designated initialiser with the following signature that allows to initialise these two properties as well as all the others:

    init(firstName: String, lastName: String, yearOfBirth: Int? = nil, middleName: String? = nil, address: String? = nil, phoneNumber: String? = nil)

If you implemented everything correctly, **the ContactListEntryTests.swift tests will succeed now!**

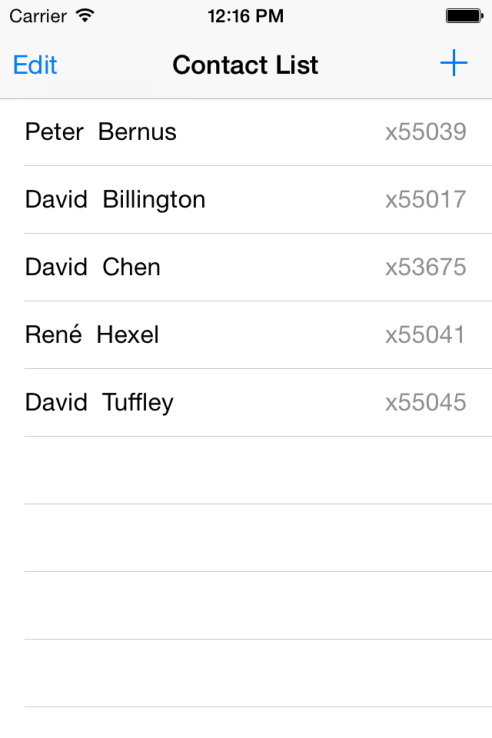
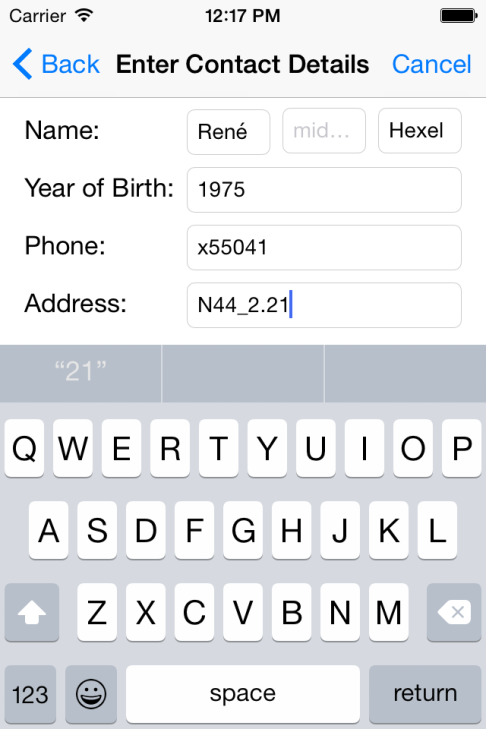
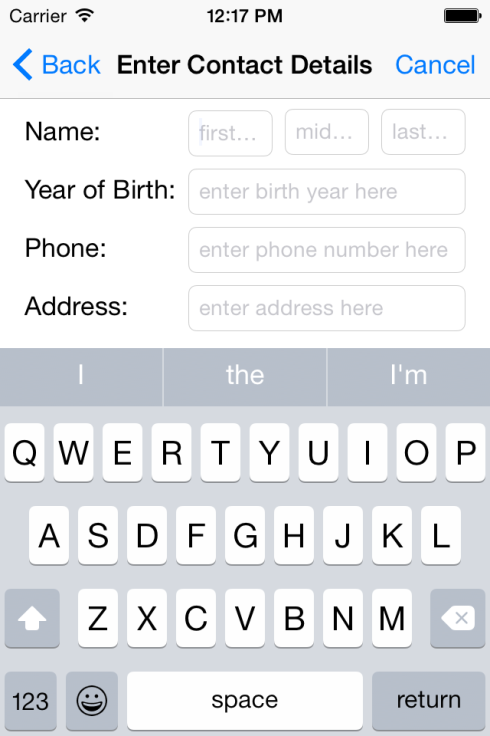
**ContactList class**

Create a ContactList class that simply contains an (initially empty) array of ContactListEntrys.  Make sure this array is stored in a property called entries.

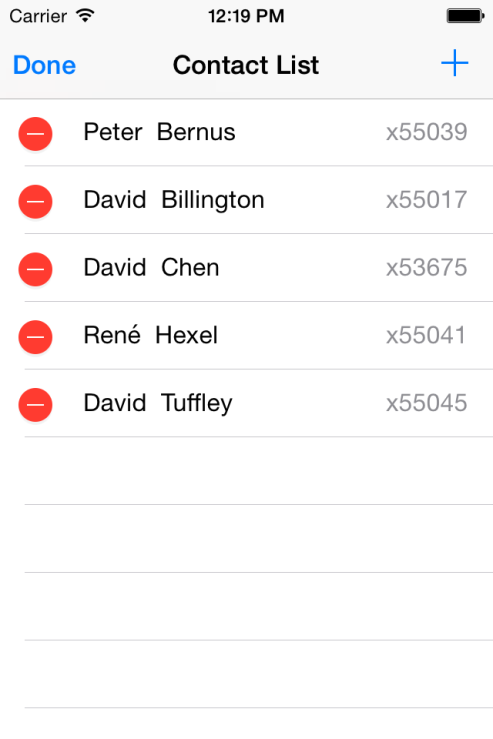
Download the ContactListTests.swift file attached to this assignment and use it to replace the one in your project created by Xcode.  If you implemented everything correctly, **all the tests should succeed now!**

***Master and Detail Views***

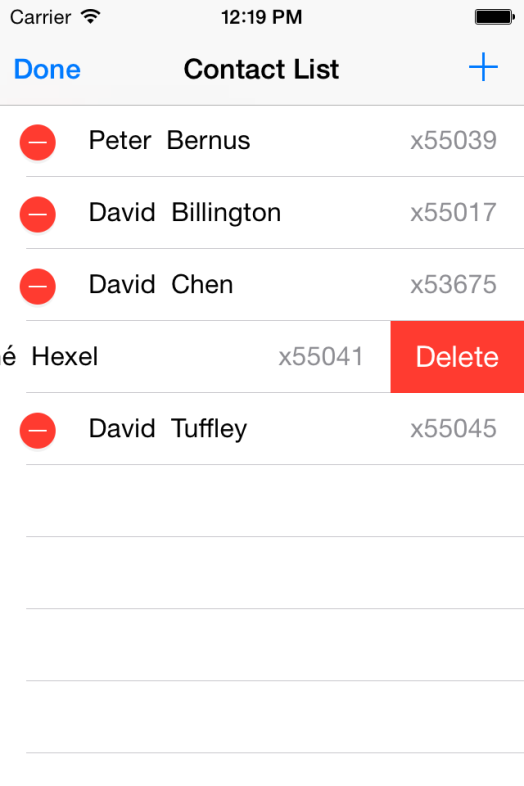
The views in the storyboard should already be in a **Navigation Controller**.  [*Read the****TableView Programming Guide****for iOS*](http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/tableview_iphone/AboutTableViewsiPhone/AboutTableViewsiPhone.html)*in the Developer Documentation!*The table view inside will be your **master** view.   Modify the **detail** view so the user can edit contact list entries.  On the master view, the user should see a table of all the names.  There should be a [+] button at the top right that takes the user to the detail view to **add** a new entry.  Tapping on an existing entry in the table should also take the user to the detail view to **edit** the existing entry.  The entries that the user should be able to enter are **First Name**, **Middle Name, Last Name**, **Year of Birth, Phone Number,** and **Address**. Here is an image of what your master and detail view User Interfaces should look like (don't forget the **Cancel** button on the detail view that should undo any changes the user did).  The detail view is shown twice, once for a new entry, and once with data already filled in (e.g., for an existing entry):

There should also be an **Edit** button at the top left of the master view (see above) that allows the user to delete rows from the table view.  When the Edit button gets tapped (or clicked on the simulator), that master view's table should look like this:



When the user clicks on one of these minus (-) signs, a delete button should appear (and clicking the delete button should delete the corresponding entry):



Make sure you link up the necessary actions and outlets!

***Controller: Putting it all together***

*Make sure you have*[*read the****TableView Programming Guide****for iOS*](http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/tableview_iphone/AboutTableViewsiPhone/AboutTableViewsiPhone.html)*in the Developer Documentation!*  Link up all your GUI elements and add the necessary code to your controller to make the buttons and all other user interface elements work. Selecting a name on the master view should take you to the detail view to edit the corresponding entry.  Pressing the [+] button on the master view should take you to the detail view with empty fields to add a new entry.  On the detail view, pressing the **Phone Book** (back) button should save the current changes and return to the master view. On the master view, pressing the **Edit** button should allow the user to select and delete entries as shown above.  (As usual, on the detail view, make sure the text fields are editable so the user can enter/modify data via the keyboard.)

***Code Quality and Commenting***

Use best practices when writing your code.  E.g., use the best existing methods for a job (such as the **fullName()** method in your Person class to display the full name of a person in the table in your master view).  E.g., usesubclassing, delegation, and other object-oriented design patterns as appropriate.  Make sure all source files (.swift files) have a comment header that contains your name as the author and your student ID.  Put JavaDoc (orheaderdoc or swiftdoc) style **documentation comments** in front of your classes, properties, and methods, describing what each of these do!  For methods, make sure you document parameters and return types as well as pre- and post-conditions!  Make sure your code is well formatted and readable, and uses consistent indentation!

Also make sure that individual changes you **commit** are documented!  **Commit early and often (at least once every time you compile/run/or test your program)!**

***Submission***

Zip up the folder that contains your project (e.g. if your project is called ContactList, you should find a *ContactList* folder on the Desktop). On the Mac, you can easily create a ZIP file of a folder by holding the Control key, clicking on the folder you want to zip up, and then select Compress "PhoneBook" to compress).  Submit the ZIP file to BBlearn (click on the link that reads *Weeks 4 and 5* at the top of this milestone page, then attach your ZIP file, and submit).