Lab – Getting Started

OVERVIEW

Date Due: Thursday, Feb. 4th by 11:59 p.m. EST ✧ **Value:** 10 points

Objective: Getting started; using the tools; writing some Objective-C.

The intent of this project is to get you comfortable using iPhone development tools. You will have the opportunity to use Xcode, and Interface Builder. Your will be guided through the creation of your first application. In addition to the tools, you will get a chance to write your first Objective-C code. You will:

- Use Xcode and learn to build, run, and debug an application
- Use Interface Builder to construct some UI
- Write some code – implement the ToDoEvent model class (ToDoEvent.h, and ToDoEvent.m)

ASSIGNMENT

1. Download today's project from http://www.cs.umd.edu/class/spring2010/cmsc498i/files/lab1/resources.zip. This contains some pre–written code you will need. There are 4 ".h" files, and 4 matching ".m" files.
2. Run Xcode, and create a new "Navigation–based Application"
   - Select *File->New Project...*
   - In the "New Project" dialog, select "Navigation-based Application" found under the iPhone OS available options
   - Click the "Choose…" button
   - In the "Save As" sheet, name your project "Lab1", choose a place to save then click "Save"
3. Your Lab1 project window is now open, let's look around…
   - The "Groups & Files" source list contains all of your projects resources. Take a look at what is there.
   - Click on the arrow next to "Lab1" to expand and see its contents. This section contains subfolders used to group all of your project's resources including source code, user interface definitions, images, etc. Expand the subfolder to see what they contain.
   - Next, expand the "Targets" source list item. You will see 1 target named "Lab1" which contains subitems referencing code to be compiled, frameworks to link against, and resources to copy into the built application.
4. Incorporate the prewritten code (4 ".h", and 4 ".m" files). There are two way to import resources into a project…
   - Option 1 – Drag and drop. Drag each ".h", and ".m" file into the "Lab1"->"Classes" folder
   - Option 2 – Select *Files->Add To Project*, and pick the files to add in the file chooser that comes up
   - After dragging or choosing, a sheet is presented asking you what to do with the files. Check "Copy items into destination group's folder (if needed)" is checked, then click "Add".
   - Note, before dragging in new files, delete any existing files with the same name in the Groups & Files list

5. Build and run the application using the menu option *Build->Build and Run – Breakpoints Off*

    - Verify that the iPhone Simulator starts and "Not Implemented" is showing in the UI

    - Exit the simulator by choosing *iPhone Simulator -> Quit iPhone Simulator*. Or, you could just quit your application by clicking on the simulators home button.

        - If you don't do this, the next time you build and run, you may be presented with a dialog that says "Stop Executable?". Xcode will want to quit your previously running application before running it again. If you see this dialog, just click "OK" to let Xcode quit the old one, and run the new application.

6. Launch Interface Builder and open the "RootViewController.xib" file. A good way to do this is by double-clicking on "RootViewController.xib" within Xcode.

7. Customize the UI of the table view, by changing its background color to some shade of blue you like

    - Locate the table view in the "RootViewController.xib" window, and select it

    - Bring up the object inspector window using either cmd+shift+I, or *Tool->Inspector*.

    - Locate the background color attribute for the table view (you may need to scroll down), and set the color.

    - Save, then build and run again to verify the color changed

8. Try out the debugger

    - Go to Xcode and open the ToDoEvent.m file.

    - Set a debuggable break point in +isImplemented, by one of the following methods

        - Option 1 – select "return NO" line, then "*Run->Manage Breakpoints->Add Breakpoint at Current Line"*

        - Option 2 – The "gutter" is the area on the left side of your code showing line numbers. You can add a break point by single clicking on the line number you want to break on. If you control-click, you'll see more options

            - If you don't see the gutter, go to Xcode->Preferences to bring up preferences. Then in the "Text Editing" preference, make sure the gutter, and line number options are enabled.

    - Run the in the debugger by selecting *Build->Build and Debug - Breakpoints On*

    - Verify your breakpoint was hit.

        - The line of code you stopped at will be highlighted in blue with a red arrow in the gutter next to it

            - Above the source, you will see a row of icons offering context sensitive options, in this case, for debugging. The row of icons offers options for enabling/disabling breakpoints, resuming execution, stepping through code, and bringing up the debugger and gdb console window

    - Working with breakpoints

        - You can enable / disable breakpoints by clicking on them directly, or control clicking to get a menu of options

        - You can bring up the list of all breakpoints (and edit them) by choosing *Run->Show->Breakpoints*

- Check out the active call stack
  - Option 1 - choose *Run->Debugger* (or click on the "debugger" icon). This will present the debugger window. Among other things, the call-stack is shown in the upper left of the window.
  - Option 2 - choose *Run->Console* (or click on the "gdb console" icon) to see the low level gdb console. The gdb console will show up in a new window. If you don't see the (gdb) prompt, scroll to the bottom.
    - From the gdb console, type "backtrace" next to the (gdb) prompt
- Step through some code using one of the following methods
  - Option 1 – In the gdb console, you can type any of the following: next, step
  - Option 2 – Using Xcode UI (in the toolbar), you can click on "Step over", or "Step into"
  - Step through code until you are in the RootViewController.m on the line that says "return eventData_"
- Print out some information about the current object, and eventData_
  - First, bring up the gdb console, then type each of the following (the part after "(gdb)" and before "//")
  - (gdb) po self // po, or print-object: print results from calling [object description]
  - (gdb) po [self description] // po self.description doesn't work... gdb doesn't handle "dot" syntax
  - (gdb) p *self // print the objects memory contents, ivars, etc.
  - (gdb) po eventData_

9. Write some code; Implement ToDoEvent.m

- ToDoEvent.m contains stubs that you are going to replace with a real implementation
- Make the +isImplemented method in ToDoEvent.m return YES (the test harness relies you making this change)
- Delete the following methods, and replace with synthesized implementations (@synthesize)
  - -setDueDate: / -dueDate
  - -setCompleted: / -isCompleted
- The remaining methods (there should be 8 of them) are to be implemented by you.
  - +eventWithTitle:
  - -initWithTitle:, -initWithTitle:dueDate:
  - -copyWithZone:
  - -dealloc
  - -isEqual:
  - -setTitle:, -title

PROJECT SUBMISSION

When you've completed this project, submit it!