Assignment 2

Calculator (1): Target-Action and Foundation Classes

Due: November 4th, 2013. 9:00 AM

Group size: 3 (for A02–03)
Please team up with the students you haven't work with in previous assignments.

Description

In this assignment, you will learn how to connect your UI elements with code, use some of the Foundation classes, and store states of your app.

Task

Part 1: Complete the basic calculator

We provide a template for a calculator that already adds numbers (at least 1 and 2). Take a close look at the code and write the necessary code to have a fully working calculator. (Ignore the buttons ← and →.)

Extra credit: Visualize the currently active operator by letting the button be pushed until the second operand is entered.

Extra credit: Handle edge cases: division by zero, unnecessary leading zero, double decimal points, etc.

Part 2: Storing user preferences

1. *Swipe to change the number of decimal places:* Patch your `ViewController.m` with `ViewController.m-gestureRecognizer.patch`. This patch adds left- and right-swipe gestures to your calculator. Implement the gestures to allow swiping to the left to increase the decimal places and vice versa for swiping right. Handle the decimal rounding appropriately.

2. *Storing the decimal display settings:* If you restart the app, the decimal point settings will be reset. Learn about `NSUserDefaults` form "Accessing Preference Values". Then, make your app remember the number of the decimal point shown and restore the setting after the application restarts.

3. **Extra credit:** Remember and restore the last numbers or the last operator entered. Tips: Use Enumerations for states.

Part 3: Storing results

4. *Remembering results:* Make your app remember the last 10 results (after each "=" presses) in an `NSMutableArray`. Allow the user to use the arrow buttons to navigate the results on the screen and use the result as an operand.

5. *Restoring history after restarts:* Store the history array in a property list file at the `Application Support` directory. (See: "Accessing Files and Directories" and "OS X Library Directory Details".) Restore the history when the application restarts.

6. **Extra credit:** Show the the number of remaining entries on the arrow buttons, e..g, "←(3)". Hint: You will need to create a new `IBOutlet`.

Submission
Create a zip archive including the following items

- ❏ Your iCalc source code
- ❏ `Members.txt` — Modify the template to match your information
- ❏ `TaskReport.pdf` — Modify the template to match your information and submit as a PDF file.

Email your submission to `iphone@cs.rwth-aachen.de`

Grading
We will grade this assignments using the following rough scale.

- 1.0 — Accomplish all extra credit tasks and clearly went above and beyond what was given in the assignment sheet by improving usability, features, or performance of the implementation
- 1.3 — Accomplish all extra credit tasks
- 2.0 — Accomplish all basic tasks according the described requirements without any compilation errors or warnings.
- 5.0 — Late submission, submission that doesn't compile and run on Xcode 5 + iOS 7 Simulator.

Looking forward
For advanced students, the following pointers will shape your mindset for the topic we will discuss in the next lab and beyond this class.

- Should your user uses multiple devices, you may want to use iCloud to store user preferences. See "Designing for Key-Value Data in iCloud".
- In an advanced applications, you may want to store the current screen that the user is working on together with the data. UIKit provide a more elaborate mechanism to support application state restoration. See "State Preservation and Restoration".
- If you have a big file to load, you might want to load files asynchronously. See. "Techniques for Reading and Writing Files Without File Coordinators"

No submission is needed in this point, but feel free to discuss your thoughts in our Facebook group.