Assignment 3

Calculator (2): Design Patterns and Concurrency

Due: November 18th, 2013. 9:00 AM **Group size:** 3 (same group as A02)

Description

In this assignment, you will improve your code by incorporating some design patterns. Then, you will add concurrency to your calculator.

Task

Part 1: MVC, delegate, and KVO

1. *MVC:* To allow reusing your calculator logic in other UIs, restructure your code with Model-View-Controller pattern. Modify the given *model* class `BasicCalculator` to provide the functions specified in A02.

2. *Delegate:* Enable `BasicCalculator` to perform the calculation in background and communicate the results to the controller via a delegate method. First, Make your `ViewController` class comply with the `BasicCalculatorDelegate` protocol. Then, make your `ViewController` class the delegate of the calculator.

3. *KVO:* Factor out the code that store the results as a separate `ResultManager` class. Now, you have multiple classes that need the result from `BasicCalculator`. Use KVO pattern to allow `ResultManager` to observe the property `lastResult` in `BasicCalculator`.

Part 2: Concurrency

You should skim Concurrency Programming Guide before attempting this part.

1. *Prime calculation:* Add a label and an activity indicator in your xib. Use `checkPrime:` method in `BasicCalculator` to determine if the integer part of the result (after pressing =) is a prime.

2. *Background calculation:* Prime check for a large number will block your UI. Put the checking in background and report the result via `PrimeCalculatorDelegate` protocol. Here, use the global concurrent dispatch queue (See: "Getting the Global Concurrent Dispatch Queues"). Code this point in `checkByGCD`*. Hint: To update the UI, you will need to do it on the `mainQueue`.

3. *Canceling:* Checking a large prime may block your CPU from processing the latest input. Cancels all the pending scheduled operations before adding a new operation. Use `NSOperationQueue`. (See: "Canceling Operations") Code this point in `checkByOpQueue`*.

4. **Extra credit:** *Canceling the executing task.* Based on `checkPrime:`, create `checkPrimeAllowCancel:` to allow cancellation of the executing prime checking.

5. **Extra credit:** *Dependency:* Instead of canceling the calculations, Allow multiple concurrent prime-checks and log the results to console in the same order as input. Code this point in `checkPerserveOrder`* Hint: You will need multiple operation queues and set the dependency of operations across queues.

---

* Feel free to modify the method signature as long as the given name a substring of the method name.

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 — Accomplishing 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 — Accomplishing all extra credit tasks
- 2.0 — Accomplishing 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.

- Concurrency programming guide describes several patterns for migrating from the thread-based practices. Check them out in "Migrating Away from Threads"
- At WWDC 2012, several design patterns and principles for working with blocks and GCD are shown in Asynchronous Design Patterns with Blocks, GCD, and XPC video.
- To make your UI works well with concurrency, check out some tips and design patterns in the video Building Concurrent User Interfaces on iOS from WWDC 2012.

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