CIS4350_Lab04_Task List

Points
100 points.

Background

Xcode and Swift form the foundation by which all iOS applications are built.

This project is designed to give you some hands on experience with:
- Utilizing Xcode as a development environment
- Compiling Swift source code
- Foundation framework classes
- Writing your own custom Swift classes
- Declaring properties

Mission

For this lab you will model a basic task list. This will be modeled by creating `TaskList` & `Task` classes that provide mechanisms for creating, storing and filtering tasks. Specifically, each task will store a textual description, a due date, priority and completion status.

[30 points] Task. [mh]

Since Priorities will be used in a number of methods, start by defining the following in `Task.h`

```
typedef enum {
    low,
    medium,
    high
} Priority;
```

In `Task.[mh]` you should implement a class called `Task`.

This class **must** provide the following properties (all read/write):

- `text` — an `NSString` which is copied when performing assignment
- `dueDate` — an `NSDate` which is retained when performing assignment
- `priority` — one of the values from the `Priority` typedef
- `completed` — a boolean indicating if the task has been finished

The `Task` class must implement the following methods:

```
init() {
}
```

Creates a new `Task` with no `text` or `dueDate` (e.g. `nil`). The `priority` should default to `medium` and `completed` should default to `NO`.

```
init(text t: String, dueDate d: Date, priority p: Priority, completed c: Bool) {
}
```

Creates a new `Task` with the specified parameters.

```
convenience init() {
}
```

Creates a new `Task` with the same defaults as the no-argument `init` method.

```
convenience init(text t: String, dueDate d: Date, priority p: Priority, completed c: Bool) {
}
```

Creates a new `Task` with the specified parameters.

[70 points] TaskList. [mh]

The `TaskList` class must provide the following read-only property:

- `count` — an integer representing the number of items in the list

The `TaskList` class must implement the following methods:

```
init() {
}
```

Creates a new empty `TaskList` instance.

```
convenience init() {
}
```

Creates a new empty `TaskList` instance.

```
func completeTasks() -> [Any] {
}
```

Returns an `NSArray` containing all `Task`s which are marked completed.

```
func incompleteTasks() -> [Any] {
}
```

Returns an `NSArray` containing all `Task`s which are not completed.

```
func allTasks() -> [Any] {
}
```

Returns an `NSArrary` containing all `Task`s in the list.

```
func pastDueTasks() -> [Any] {
}
```

Returns an `NSArray` consisting of classes which are past due. Past due tasks are defined as tasks which have a due date prior to the current day. For example, a `Task` with a `dueDate` of 5:16:00pm on 9/11/2015 is not past due until the date rolls over to 9/12/2015.

```
func tasksBetween(_ start: Date, and stop: Date) -> [Any] {
}
```

Returns an `NSArray` consisting of classes which are between the specified `start` and `stop` dates/times (inclusive on both sides). These between times should be treated exactly as specified — no rounding to beginning or end of day.

```
func tasks(with p: Priority) -> [Any] {
}
```

Returns an `NSArray` containing all `Task`s with a `priority` matching the specified `Priority`.

```
func add(_ task: Task) -> Bool {
}
```

This method should add the specified `Task` to the list. If a `Task` already exists in the list where all properties match the specified `task`, then the method should return `NO` and not insert anything. If no matching `Task` is found, the item should be inserted and the method should return `YES`.

```
func removeAllTasks() {
}
```

This method should remove all `Task`s from the list.

```
func remove(_ task: Task) -> Bool {
}
```

This method should remove the specified `Task` from the list. If a `Task` exists in the list where all properties match the specified `task`, then the method should return `YES` and remove the item from the list. If no matching `Task` is found, then nothing should be removed and the method should return `NO`.

```
func removeCompletedTasks() {
}
```

This method should remove all `Task`s that are marked complete from the list.

Getting Started

Open Xcode, then create a new project by selecting File → New Project… then under OS X select Application and select Command Line Tool. Type name the project "`LasttName_FirstName_Lab04`", so mine will be "`Ajaj_Ola_Lab04`"Choose Organization name to be your first name followed by your last name (e.g., Ola Ajaj) and choose Swift from the last drop menu to be the Language.

This will build out a small project with a file called "`LasttName_FirstName_Lab04`" containing a `main.m` consisting of a simple "Hello World" implementation.

Next you'll want to have Xcode stub out your `Task` and `TaskList` classes for you.
To do this, simply right click (or control click) under Lab04 directory and select Add → New File

Under OS X select Source, then choose Cocoa Class, and be sure that the Subclass of drop-down specifies `NSObject`, and that Language points to Swift Then click Next.

Next, name the file `Task.m`, ensure the also create `Task.h` checkbox is selected, the default location should automatically be under the project location you selected, Add to Project should specify `Lab04` and the Target `Lab04` should be checked. Click Finish to create the classes. You should now have a `Task.m` and `Task.h` file stubbed out for you in the Sources folder. Repeat this process for the `TaskList` class.

`** Comments **` Add a comments section on top of your `main` file to add your name, TUID, class, Project number, and purpose of this project.

You should now be ready to start implementing the classes and test it using `Lab04`.

Notes

Since we will be using our own `main` to test your class, it is **very important** that you name your classes `TaskList` & `Task`, implement it in files called `TaskList.[mh]` & `Task.[mh]` and name your methods **exactly** as they are listed above (copy and paste is your friend).

You are encouraged to develop your own `main` which exercises the methods of this class. When the project is graded we will swap in our own `main` to test the robustness of your class. So, we should be able to simply add a `#import "TaskList.h"` and go.

Behind the scenes you are welcome to implement additional methods and classes to support both of these classes (e.g. supporting methods). However, these internal workings should not be exposed to the outside world — meaning that the user of this class should never have to call anything other than the methods listed above, as such they should only appear in the `.m` files.

I found the NSDateComponents class useful when performing the past due checking.

Testing

No sample test harnesses will be provided, though you are strongly encouraged to develop your own testing scenarios.

Grading

Your projects will be graded on the following criteria:

- Correctness of application
- Adherence to Swift and iPhone coding conventions
- Neatly formatted and indented code
- Well documented header files

Submission

To prep your project for submission, you need to zip up your entire project directory. To do this right click (or control click) on the project folder `LasttName_FirstName_Lab04` and select "Compress `LasttName_FirstName_Lab04`".

This will create a `LasttName_FirstName_Lab04.zip` file that contains your project.

Simply upload `LasttName_FirstName_Lab04.zip` through Lab04 link on Blackboard.