ASSESSMENT

POWERING APPS WITH CODE

The tasks below ask you to construct various View Controllers and Views. These exercises will give you more experience with developing a fully interactive app, building a data model, and creating and laying out basic views using Swift and Interface Builder.

GOALS OF ASSESSMENT

- Create a working app that responds to user input and updates the UI.
- Demonstrate the application of basic object-oriented principles in a working app.
- Segue between View Controllers and pass data between them.
- Name the components of an app and describe the lifecycle of an app.

REQUIREMENTS

Your app must:

- Meet all the tasks below.
- What is it? A working task manager app for just 3 tasks, a la the "Commit-to-3" app.
- **Task 1** – The main View Controller should display 3 text fields, one for each task. Changing the text in field sets the title of the Task.

  Beside each text field should be a button that first displays an open circle or square. Tapping on this should change it to a checkmark (and set the corresponding Task's isComplete flag to true, see below).

  Also, beside each text field, should be a button labeled "Details" (or have an arrow "→").
- **Task 2** – Develop a data model for your app. First, there should be a class called TaskManager. That should hold three properties of type Task. You need to define a Task class as well. Recommended: TaskManager's init() method should instantiate the three Tasks.
- **Task 3** – The Task class should hold the following properties with the respective types:
  - title : String
  - notes : String
  - isComplete : Bool
  - isImportant : Bool
  - isUrgent : Bool
  - dateDue : NSDate
- **Task 4** – The Task class should have the following methods:
  - markComplete() - marks the task as completed

    - daysUntilDue() -> Int - returns the number of days left until it's due

- **Task 5** – Tapping on the "Details" button should reveal a View Controller enabling you to edit the task in greater detail. Use the following Views:
    - title → UITextField
    - notes → UITextView
    - isComplete → UISwitch
    - isImportant → UISwitch
    - isUrgent → UISwitch
    - dateDue → UIDatePicker (When you pick a new dateDue, a UILabel above it should say "Number of days to finish: NN" where NN is the result of calling the method daysUntilDue.)
- **Task 6** – Have a button in the nav bar takes you to an "Eisenhower Matrix." This should be a View Controller that displays 4 squares in 2 x 2 fashion. The two columns should have UILabels at the top saying "Urgent" and "Not Urgent." The two rows should say "Important" and "Not Important."

  The titles of the Tasks that have both isUrgent and isImportant set to "true" should appear in the upper-left box. Continue in this way so all three titles are represented in these squares. An example can be found here: http://jamesclear.com/wp-content/uploads/2014/04/eisenhower-box.jpg

DELIVERABLES

- The entire app folder, zipped, posted to Dropbox.

WAYS TO GET STARTED

Answer the following questions:

- What's a class? What's an object? What's a method? What's a property?
- What does it mean to instantiate a class?

RESOURCES

Some external links:

- https://www.weheartswift.com/object-oriented-programming-swift/
- https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/ClassesAndStructures.html

EVALUATION

Your assignment will be evaluated regarding the extent to which you meet the above requirements using this rubric:

Assignment readiness

Exceed (2): All exercises complete and fully functional. App does not crash under common or edge use cases.

Pass (1): All but one or two exercises complete and fully functional. App does not crash under common use cases.

Doesn't pass (0): Many exercises incomplete. App crashes under common use cases.

Stability & performance

Exceed (2): Manual inspection of code does not have obvious major or minor bugs. No obvious crashes on manual code inspection. Little copy/pasted code.

Pass (1): Code indicates a few small bugs. Some code duplication.

Doesn't pass (0): Code frequently exhibits bugs. Edge cases would cause code to crash. Duplicated functionality/code abounds.

Style and readability

Exceed (2): Code has consistent style throughout codebase in a manner that is consistent with other sample code. Commented well in all unclear areas. Variables and function names clear and consistent.

Pass (1): Code is mostly styled consistently, with a few outliers. A few comments peppered throughout the codebase. Variables and function names make sense and are relatively consistent.

Doesn't pass (0): Code is inconsistently styled throughout codebase, or styled in a non-Swift/non-iOS manner. Commented out code abounds. No comments about unclear code. Variable and function names inconsistent and unclear.