ITP 342 Homework 4

Goal

- You will create an iPhone app to display questions and answers.
- This homework focuses on creating and unit testing the model.
- Use gesture recognizers to display questions and answers.
- Future assignments will build upon this homework.
- **We strongly recommend you check your project on each device (via the simulator) against each instruction before submission. You'd be surprised at how many things you can fix!**

Assignment

- Open Xcode and create a new Single View application for the iPhone. You may not start with a previous project.
- Add a label on the storyboard to display a question. Set the proper AutoLayout constraints. Make sure the label scales for different displays.
  - A general rule of thumb is that UI should be somewhat consistent between device sizes.
  - For example, if your UI maintains its exact size and position between different screen sizes with only one chunk of whitespace expanding, you're likely not using AutoLayout as effectively as you could be.
  - A better approach would be to fix padding in different directions, expand UI element size, and in general make it look like the UI "fits" the device.
  - Try and make your project creative without copying the exact colors and fonts etc. from the example!
- Create a Cocoa Touch class called **`Flashcard`**:
  - This class needs to inherits from `NSObject`.
  - Add public properties for **question** and **answer** that each hold an `NSString`. These properties need to be *readonly*.
  - Add a public property called **isFavorite** that holds a `BOOL`.
  - Add the following public methods:

```
// Initializing the flashcard
- (instancetype) initWithQuestion: (NSString *) question
                           answer: (NSString *) ans;
- (instancetype) initWithQuestion: (NSString *) question
                           answer: (NSString *) ans
                       isFavorite: (BOOL) isFav;
```

  - Implement the methods in the implementation.

- Create a Cocoa Touch class for the model called **`FlashcardsModel`**:
  - The model needs to have a private property named **flashcards**, which is a mutable array containing `Flashcard` objects.
  - Create a public *readonly* property named **currentIndex**, which is an unsigned integer of the current index of the flashcard that is being displayed.
  - Create an `init` method in the implementation to initialize the model.
    - Initialize the flashcards array. (Use the code snippet option from the Library in order to get the correct method signature.)
    - Add **at least** 5 flashcards to the array.
    - Initialize the currentIndex to 0.
  - Add the following required public methods.

```
// Creating the model
+ (instancetype) sharedModel;

// Accessing number of flashcards in model
- (NSUInteger) numberOfFlashcards;

// Accessing a flashcard – sets currentIndex appropriately
- (Flashcard *) randomFlashcard;
- (Flashcard *) flashcardAtIndex: (NSUInteger) index;
- (Flashcard *) nextFlashcard;
- (Flashcard *) prevFlashcard;

// Inserting a flashcard
- (void) insertWithQuestion: (NSString *) question
                     answer: (NSString *) ans
                   favorite: (BOOL) fav;
- (void) insertWithQuestion: (NSString *) question
                     answer: (NSString *) ans
                   favorite: (BOOL) fav
                    atIndex: (NSUInteger) index;

// Removing a flashcard
- (void) removeFlashcard;
- (void) removeFlashcardAtIndex: (NSUInteger) index;

// Favorite/unfavorite the current flashcard
- (void) toggleFavorite;

// Getting the favorite flashcards
- (NSArray *) favoriteFlashcards;
```

- Implement the methods in the implementation.
  - When inserting a flashcard without an index, add it at the end.
  - When inserting a flashcard with an index, make sure the index is less than or equal to the number of flashcards.
  - For the remove and insert methods, call the similar NSMutableArray methods.
  - When removing a flashcard, check that the index is less than the number of flashcards.
  - Set the currentIndex appropriately in the "Accessing a flashcard" methods.

- Edit the View Controller:
  - Create a private property for the model.
  - In the viewDidLoad method, create the model by calling the sharedModel method. Then get a random flashcard and display its question.
  - Use Gesture Recognizers to display a random flashcard if the view is tapped and handle swiping left and right. Display the answer when the view is double tapped.
  - You are unable to test the all of the methods in the model using the UI. Use XCTest, which is Apple's unit testing framework, to test all of the public methods. You may use the XCTestCase class that is automatically generated for you or you may create a new class. In this class, add a private property of type FlashcardModel and update the setUp method to get an instance (use alloc and init). Create various test methods and use the XCTAssertEqual and XCTAssertEqualObjects functions.
    - Here is an outline to test the sharedModel method (i.e., singleton):

```
- (void) testSharedModel {
  FlashcardModel model1 = [FlashcardModel sharedModel];
  FlashcardModel model2 = [FlashcardModel sharedModel];
  // add a flashcard to model1
  // check model2 for increase in number of flashcards
}
```

- Add animations to the View Controller:
  - Use Animations to have the flashcard label animate. Have the old flashcard fade out and then have a new question or answer fade in. You may also make other changes to the label. You may have to make some helper methods. Display the question in one color and the answer in a different color.

Challenges (Encouraged, but not graded)

- Display a random flashcard when the user shakes the device.
- Play a short audio clip when a new flashcard appears.

Submission

- Ensure your name is in the header comment of each file.
- Compress your project folder.
- Rename the zip to *LastnameFirstname*HW4.zip where *Lastname* is your last name and *Firstname* is your first name.
- **You will lose points if naming the zip as stated is not followed.**
- **Improper submission of files will result in a substantial point deduction.**
- Submit the .zip file on Blackboard under Assignments.
- Ensure your submission is a reasonable file size without any large uncompressed assets or you may face deductions.

Grading (50 pts)

- Model: 15 pts – quotes array, public methods, singleton
  - Includes all of the methods described on page 2.
  - Failure to not include or name the functions as stated on page 2 will result in point deductions.
  - The singleton must be implemented correctly.
  - The array must hold your flashcards correctly.
- Unit Testing of Model: 12 pts
  - Each public method must be tested including the singleton.
  - There should be a total of 12 tests. See page 2.
  - Each test is worth 1 point.
- Gesture Recognizers: 10 pts
  - Each gesture is worth 2.5 points each.
  - You must implement tap, double tap, swipe left, and swipe right correctly.
- Animation (different color or something for answer): 5 pts
  - The answer and the question should be distinguishable.
  - Your animation should be smooth and not jitter or flash.
- App icon and appropriate name in Simulator: 2 pts
  - The name should pertain to the HW4 or flashcards and not be truncated in the simulator (“Flashcard Proje….” on the home screen is too long and wrong).
  - The app icon should follow guidelines set forth on the first week of class.
- Layout: 6 pts (looks good in iPhone SE, iPhone 8, and iPhone 8 Plus)
  - There should not be any AutoLayout warnings or errors.
  - Make sure it scales to each device appropriately.
  - Each device is worth 2 points.

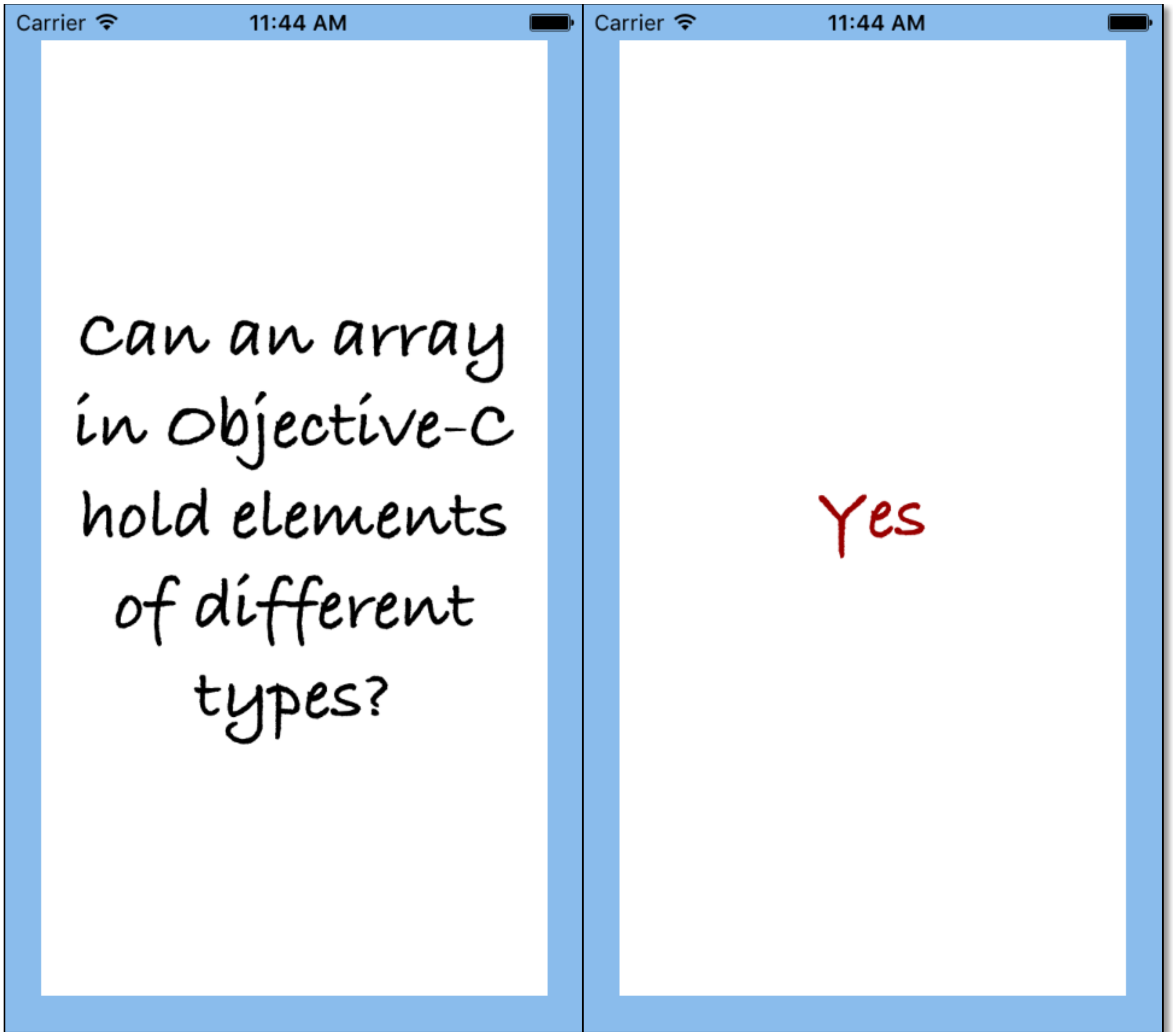
  - If there are AutoLayout runtime errors, you will receive 0/2 points for that device because this means you're explicitly contradicting yourself, which suggests you don't understand what your constraints are doing.
  - If the constraints work but make for an awkward UI on that screen, receive 1/2 points for that device.
  - Other layout deductions can include, but are not limited to:
    - UI overlapping with status bar.
    - Subviews hanging off of their container views.
    - Text of a reasonable length being truncated in a label etc.
  - Please see us in OH or post to Piazza if you still need help with AutoLayout or have questions about your UI.

Important Notes

- Challenges are not graded, but are strongly recommended!
- Please do not submit a project including a challenge if it does not work or causes other bugs in your project that are graded.
- Code quality should be good with helper functions as necessary, no legitimate compiler warnings, and no blocks of repeated copy-pasted code or chunks of unused commented code. Consistent style and indentation is also a good practice to get in the habit of. Points will be deducted for these issues.
  - If your code is clear and easy to follow, it's easier for us to give more partial credit if something were to go wrong, so it's in your best interest to make your code readable.
- Your name should still appear in the top of each file.
- You cannot reuse the lecture week's project or sample project for your homework, nor can you reuse an old HW project. Please create a new project.
- We reserve the right to deduct points if your implementation is unsatisfactory.
- Please follow the spirit of the project.
- If you would like to deviate slightly from these guidelines, please check with us before doing so, even if your way requires more work. We can sometimes make exceptions but want to ensure you're able to receive points for the items we are looking to assess in this assignment.
- If you have a bug in your assignment that makes it difficult or impossible to test, we will not fix the bug to test your app and you will not receive points for features we cannot test.
- You are responsible for testing your app thoroughly on all supported devices before submission. We are happy to help work through issues and check against the rubric in office hours, but cannot be responsible for all of your testing prior to grading.

- If you have concerns about your grade, make a private post on Piazza or visit Office Hours. Regrade requests can result in a change in your grade either up or down, if we find something we missed in the previous grading.
- You have one week from when the assignment was graded to dispute the grade.
- Copying someone else's code is a violation of academic integrity policies and you will be reported. It is trivial for us to tell a project has been copied, in part or in full.

Example


Carrier
11:44 AM
Can an array in Objective-C hold elements of different types?
Carrier
11:44 AM
Yes