mariejinpark/swift-dictionaryBillOrSteve-lab-ios-0916

Dictionaries: Steve & Bill Facts

Overview

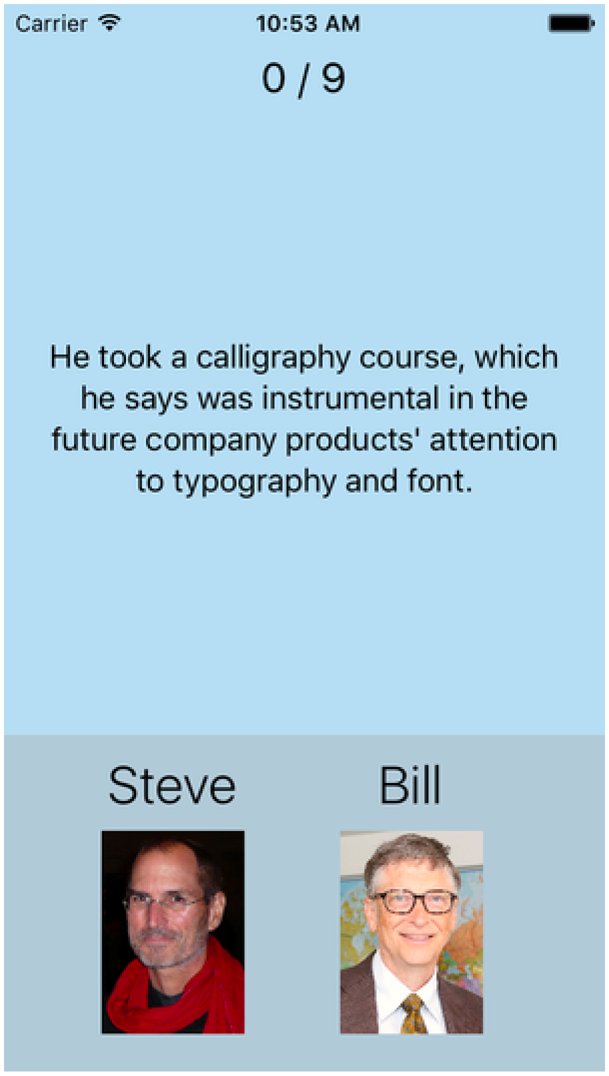
This lab encompasses the full use of dictionaries and accessing elements within a dictionary.

But it's not solely be about dictionaries (as we will be using Xcode), so the student should be able to tackle all the problems presented within this lab which might include: * IBActions * IBOutlets * Tuples * Functions with return values * Arrays * Dictionaries * Accessing elements within a dictionary

How Much Do You Know About Bill & Steve?

Steve Jobs and Bill Gates: Two pioneers of the personal computing revolution, yet so little is known about them. In this lab, you're going to change that. You're going to create an iPhone app that will quiz users on how much they know about these two guys.

Your iPhone app will ultimately look like this:



He took a calligraphy course, which he says was instrumental in the future company products' attention to typography and font.

The user will be presented with a random fact about Bill or Steve in the top portion of the UI. They can then press one of the buttons below to guess whether the fact pertains to Bill or Steve, and the app will let them know if they're right or not.

Some parts of this app have already been written for you. You can open `BillOrSteve.xcworkspace` from this repo to get started. You can try to build and run the app, but it won't do anything useful or interesting yet. That's the part you need to fix.

Let's get started.

`ViewController`

The crux of this application will be contained in `ViewController` (in the `ViewController.swift` file). Some parts of the view controller have been given to you. Right now, it looks like this:

```
class ViewController: UIViewController {
    override func viewDidLoad() {
        super.viewDidLoad()
    }

    func randomNumberFromZeroTo(number: Int) -> Int {
        return Int(arc4random_uniform(UInt32(number)))
    }

    func randomPerson() -> String {
        let randomNumber = arc4random_uniform(2)

        if randomNumber == 0 {
            return "Steve Jobs"
        } else {
            return "Bill Gates"
        }
    }
}
```

There are two methods already filled out for you:

`randomNumberFromZeroTo()` takes an `Int` as a parameter and returns a random `Int` between 0 and the parameter you passed it. You will probably find this useful when grabbing a random fact to show to the user.

`randomPerson()` will return a `String` with a random name: either "Bill Gates" or "Steve Jobs". You can use this to randomly select which person to show a fact for.

There is also a partially implemented `viewDidLoad()` method which, for now, calls `super.viewDidLoad()`. You'll probably need to add some code to the end of this method to initialize your app.

These methods aren't much, but they should prove useful in implementing this app. The rest, however, is up to you to fill out.

Instance Variables

First things first: You should probably create an *instance variable* to store your random facts. You're storing 4-5 facts for two different people, Bill and Steve. What type of variable do you think you should create?

A `Dictionary` would make sense. Since Bill and Steve are identified using `String`s from the `randomPerson()` method, it would make sense to have a dictionary with `String` keys. And since they are mapped to a set of facts, it would make sense for the values to be of type `[String]`. Therefore, you should create an instance variable of type `[String: [String]]`.

You can declare this variable outside of the `viewDidLoad()` method, right above it.

Next you should populate the dictionary with facts. For the sake of organization, create another method called `createFacts()` to do this, and then call `createFacts()` at the end of `viewDidLoad()`. Remember, the dictionary instance variable you created to store the facts should probably have two keys, "Bill Gates" and "Steve Jobs", which correspond to the strings returned by `randomPerson()`. Each key should be assigned an array containing the facts about each person.

Here are the facts you can use:

- **Steve Jobs**
  - "He took a calligraphy course, which he says was instrumental in the future company products' attention to typography and font."
  - "Shortly after being shooed out of his company, he applied to fly on the Space Shuttle as a civilian astronaut (he was rejected) and even considered starting a computer company in the Soviet Union."
  - "He actually served as a mentor for Google founders Sergey Brin and Larry Page, even sharing some of his advisers with the Google duo."
  - "He was a pescetarian, meaning he ate no meat except for fish."

- **Bill Gates**
  - "He aimed to become a millionaire by the age of 30. However, he became a billionaire at 31."
  - "He scored 1590 (out of 1600) on his SATs."
  - "His foundation spends more on global health each year than the United Nation's World Health Organization."
  - "The private school he attended as a child was one of the only schools in the US with a computer. The first program he ever used was a tic-tac-toe game."
  - "In 1994, he was asked by a TV interviewer if he could jump over a chair from a standing position. He promptly took the challenge and leapt over the chair like a boss."

Go ahead and populate that dictionary in `createFacts()`.

Methods

Finally, you should create a method called `getRandomFact()`. This method should return a tuple consisting of the fact and who it pertains to. Since you will have to check to see if the proper keys have been created, you will also have to handle the case in which the keys are not present in the dictionary. (This is a *bug*, but you should still handle that case.) This means that the return value of `getRandomFact()` should probably be an `Optional` tuple.

Implementing this method takes a bit of thought, but you have the plumbing in place to create this method. First of all, you already have the method `randomPerson()` which will return either the `String` "Bill Gates" or "Steve Jobs", which also correspond to keys in your dictionary. You also have the method `randomNumberFromZeroTo()`, which will return a random number between 0 and the argument you pass to it. You can use these two methods to randomly select a key from the dictionary of facts, then randomly select a fact from the array associated with that key.

Now it's time to write up the UI.

User Interface

Open up the user interface (`Main.storyboard`). The basic UI has been designed for you already, but you have to write up the outlets and actions.

First, you should create an outlet for the label that shows the current fact in the middle of the user interface. This is a `UILabel`. Call this outlet something like `factLabel`. You will change this to the text of the randomly-selected fact.

You should also create an outlet for the score counter at the top of the screen. This is where you'll let the user know how many facts they have gotten correct.

You should also create outlets for both the **Bill** and **Steve** buttons, as you may want to interact with them in some way through your code (such as enabling or disabling them).

Finally, you need to create actions for the **Bill** and **Steve** buttons at the bottom of the screen. This allows users to make a guess, and lets you check to see if the user was correct or not.

Now you can finish implementing the logic of the game in the view controller.

Back to the View Controller

You should next create a method called `showFact()` that will show the first fact. This method is fairly simple: It should get a random fact and person using the method you wrote, then change the text in the middle of the UI to the text of that fact. You'll probably also want to store the current fact and person in an instance variable, so you can later check to see if a correct guess has been made.

When a user presses a button, your logic should check to see if the button pressed is the correct guess. You should have wired the buttons up to different IB actions, so you'll easily know which one is pressed. You can check to make sure the right person was selected. If the guess is correct, you should increment the counter at the top of the screen. If the guess was incorrect, don't increment the counter!

Regardless of whether the guess was correct or not, you should then display another fact and let the user play another round. You can easily accomplish this by calling your `showFact()` method again.

Extra Credit

This lab has been left fairly open-ended. There's a lot to cover here, but if you use everything you've learned in previous lessons, you should be able to complete it.

In fact, you may find it fairly easily! If that's the case, there are some more advanced features you can add to the game. Try removing a fact from the dictionary when it is randomly selected, so that a fact won't be shown more than once. You can then limit the game to 9 rounds and show a total score out of 9 at the top of the screen.

Don't worry if you get a bit stuck—this lab is challenging. Take some time to think through its design and implementation. Good luck!