

# Flashcards

## Lesson 4

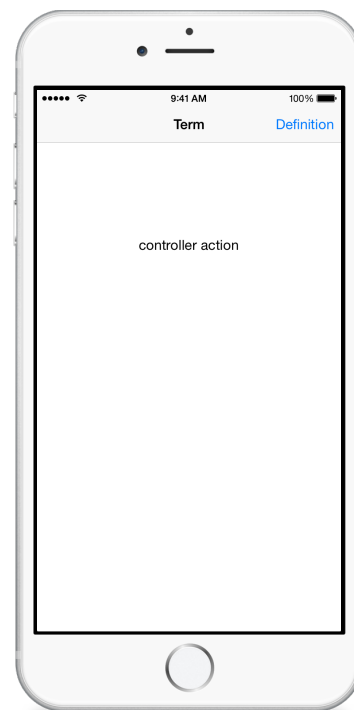


### Description

Add the `Deck` model to the Term Controller, and update the interface with a random card drawn from the `Deck`.

### Learning Outcomes

- Discover how to bind a view controller in Interface Builder to a class implementation.
- Practice declaring properties and implementing methods.
- Analyze code to identify potential causes of runtime errors.
- Recognize how Swift optionals can represent the presence or absence of a value.
- Distinguish a computed property from a method.
- Apply the optional binding syntax with `if let`.



### Vocabulary

identity inspector	property	method
optional	computed property	optional binding

### Materials

- **Flashcards Lesson 4** Xcode project

### Opening

What steps do we need to take to integrate a `Deck` with the Term Controller?

## Agenda

- Using Interface Builder and the Document Outline (⌘⇧D), select the Term Controller and use the Identity Inspector (⌘⇧3) to reveal the binding to the custom `ViewController` class.
- Explain how each individual view controller in the storyboard can be associated with a specific class within the project.
- Discuss the naming inconsistency of Term Controller in the storyboard, and the `ViewController` class name.
- Using the Project Navigator (⌘⇧1), rename **ViewController.swift** to **TermController.swift**, and update the class name to `TermController`.

```
class TermController : UIViewController {
```

- Using Interface Builder and the Document Outline (⌘⇧D), select the Term Controller and use the Identity Inspector (⌘⇧3) to change the **Custom Class** to `TermController`.
- Add a `Deck` property to the `TermController` class.

```
let deck = Deck()
```

- Discuss how the `TermController` `viewDidLoad` method will draw a random `Flashcard` from the deck, and use that `Flashcard` `term` property to update the text label.
- Add a naive `randomCard` method to the `Deck` class.

```
func randomCard() -> Flashcard {  
    let randomIndex = Int(arc4random_uniform(UInt32(cards.count)))  
    return cards[randomIndex]  
}
```

- Explain the necessary type conversion for the `UInt32` argument expected by `arc4random_uniform`, and the need to convert its return value to an `Int`, to subscript the array.
- Discuss whether or not the `randomCard` method would return a `Flashcard` object when the deck is empty, and `cards.count` is 0.
- Improve the `randomCard` method with an optional return type.

```
func randomCard() -> Flashcard? {
    if cards.isEmpty {
        return nil
    } else {
        let randomIndex = Int(arc4random_uniform(UInt32(cards.count)))
        return cards[randomIndex]
    }
}
```

- Explain how the optional `Flashcard?` type represents a "wrapped" value that will either be `nil` or a `Flashcard` when unwrapped.
- Discuss how the `randomCard` method has no parameters, only does the necessary work to return a value, and "feels" like a property of a `Deck`.
- Replace the `randomCard` method with a computed property.

```
var randomCard -> Flashcard? {
    if cards.isEmpty {
        return nil
    } else {
        return cards[Int(arc4random_uniform(UInt32(cards.count)))]
    }
}
```

- Discuss the similarities and differences between methods and read-only computed properties.
- In `TermController`, update the implementation of `viewDidLoad` to draw a `randomCard`, and use that card to update the text label.

```
override func viewDidLoad() {
    super.viewDidLoad()
    if let flashcard = deck.randomCard {
        termLabel.text = flashcard.term
    }
}
```

- Discuss the use of optional binding when accessing the `randomCard` property.
- Run the app (⌘R), and observe the random card term on the screen. Tap the Definition button, observe how the default text view text appears, and navigate back to the first view controller.

## Closing

What do you think would happen in our app if we had kept the naive implementation of `randomCard`? What if our app was loading terms and definitions from an external data source, and the loading of the data failed, resulting in an empty `Deck`?

## Modifications and Extensions

- Add a parameterized initializer to the `Deck` class. Enable the controller to declare a dictionary of card data and pass it to this new `Deck` initializer.

## Resources

View Controller Basics <https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/AboutViewControllers/AboutViewControllers.html>

UIViewController Class Reference [https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIViewController\\_Class/index.html](https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIViewController_Class/index.html)

The Swift Programming Language: Properties [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/Properties.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Properties.html)

The Swift Programming Language: Methods [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/Methods.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Methods.html)

The Swift Programming Language: Computed Properties [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/Properties.html#//apple\\_ref/doc/uid/TP40014097-CH14-ID259](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Properties.html#//apple_ref/doc/uid/TP40014097-CH14-ID259)

The Swift Programming Language: Conditional Statements [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/ControlFlow.html#//apple\\_ref/doc/uid/TP40014097-CH9-ID127](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/ControlFlow.html#//apple_ref/doc/uid/TP40014097-CH9-ID127)

The Swift Programming Language: Subscripts [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/Subscripts.html](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/Subscripts.html)

The Swift Programming Language: Optionals [https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift\\_Programming\\_Language/TheBasics.html#//apple\\_ref/doc/uid/TP40014097-CH5-ID330](https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID330)

BSD Library Functions Manual: ARC4RANDOM(3) [https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man3/arc4random\\_uniform.3.html](https://developer.apple.com/library/mac/documentation/Darwin/Reference/ManPages/man3/arc4random_uniform.3.html)