UI and Code

Dr. Sarah Abraham

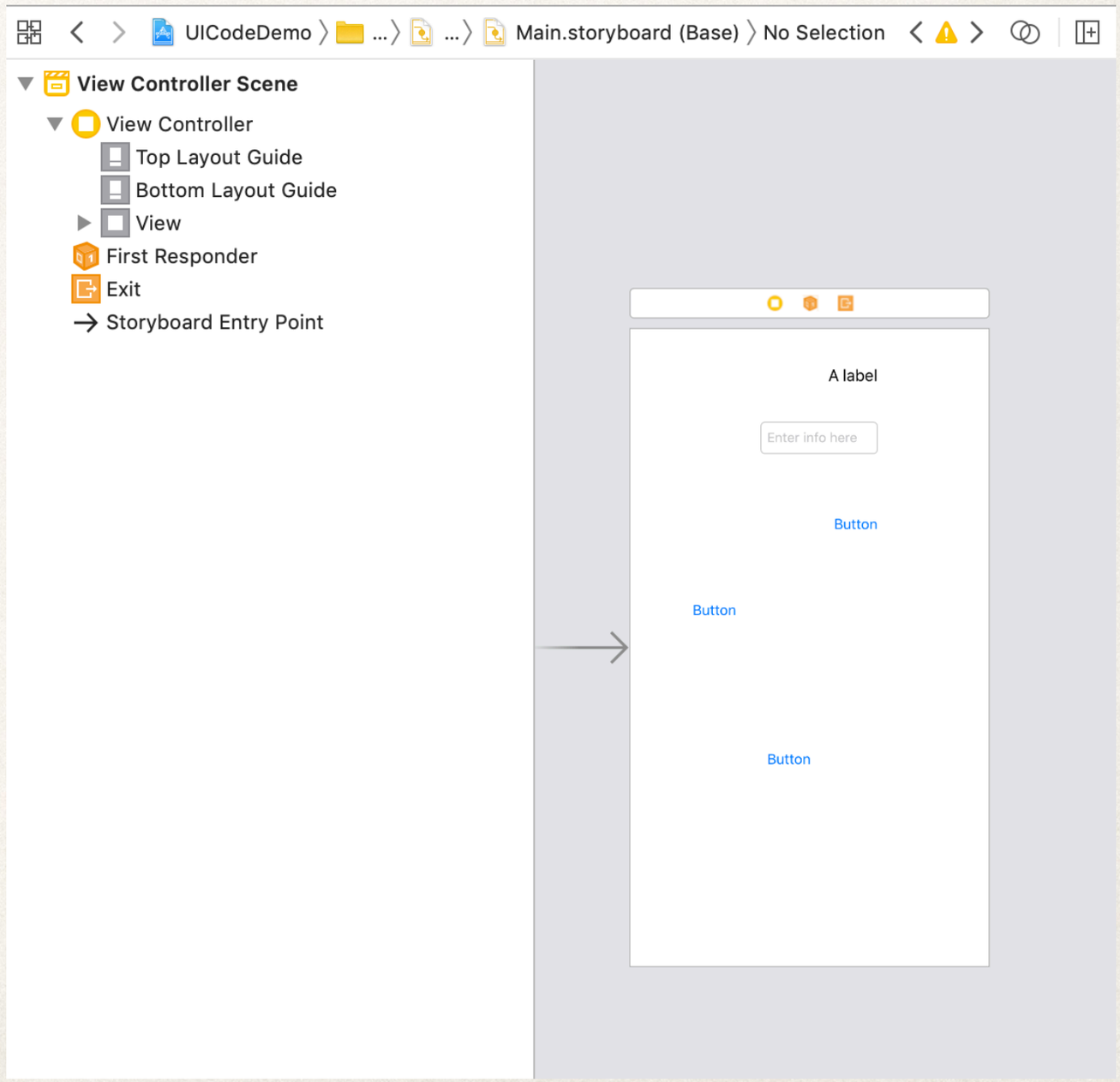
University of Texas at Austin
CS329e
Spring 2020

Providing UI Functionality

- UI (user interface) displays images, text, widgets etc that a user might need to see and interact with
- Code backend determines widget actions and responses to user input
- iOS developers work with two systems:
  - Interface Builder (visual interface editors for storyboards)
  - Xcode editor (text-based interface for code creation and editing)
  - Both of these systems exist within Xcode

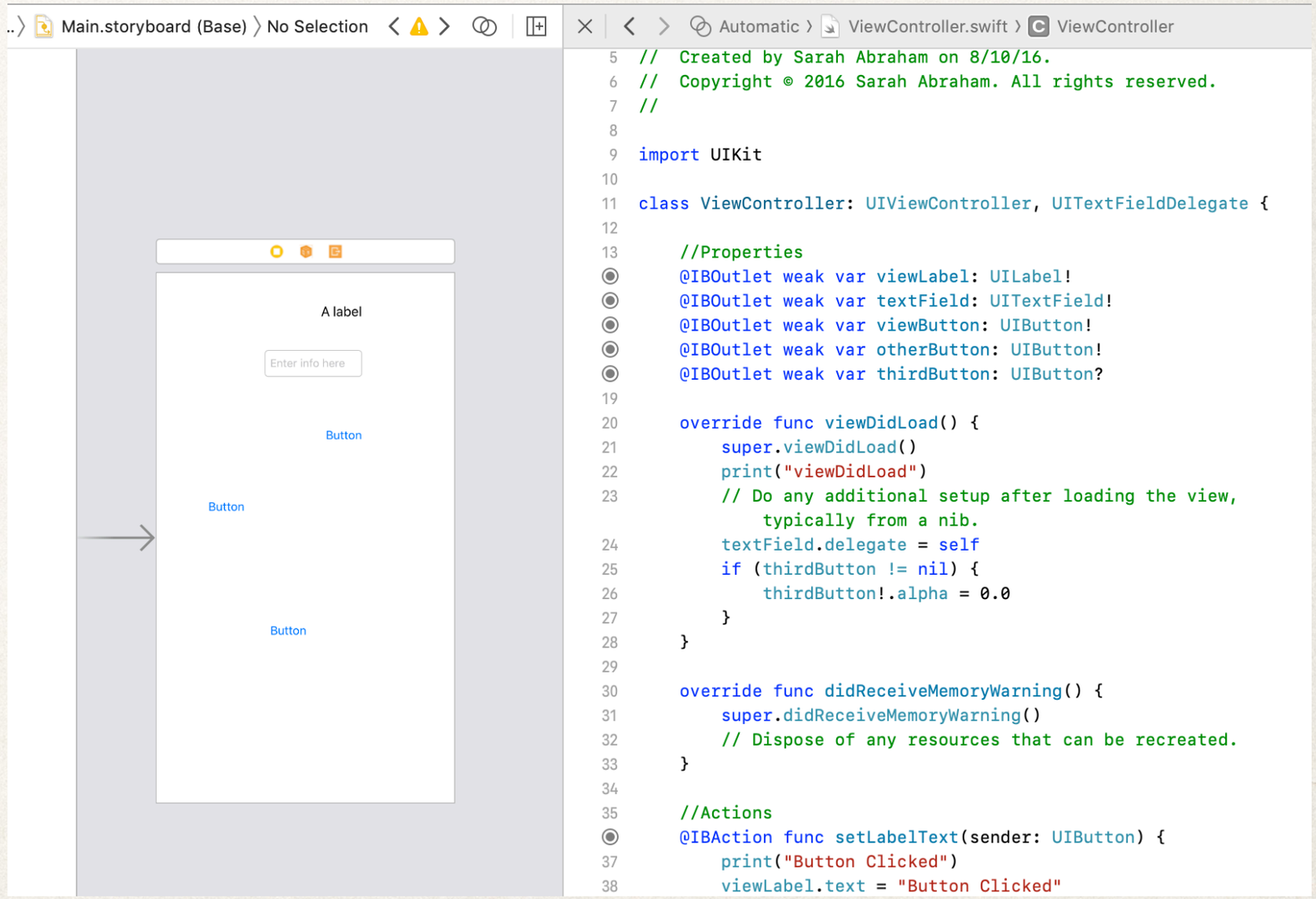
Create a UI

- Using Interface Builder:
  - Define layout for Main.storyboard's View Controller
  - Add necessary widgets/elements



Create Outlets for UI Elements

- Using Xcode editor:
  - Create outlets for widgets
  - Connect outlets to widgets



What are Outlets?

- Outlets provide way to reference interface objects in storyboard using source code

- Create outlets by control-dragging from storyboard interface object to ViewController swift file:

  - `@IBOutlet weak var widgetName: widgetType!`

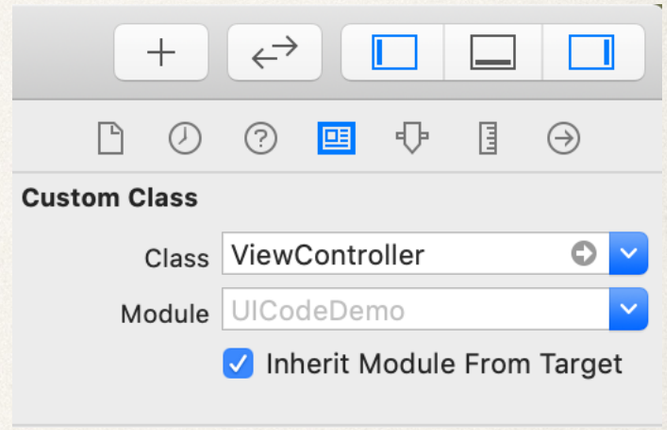
- This property is connected to the interface object so it can be manipulated at runtime

Understanding IBOutlets

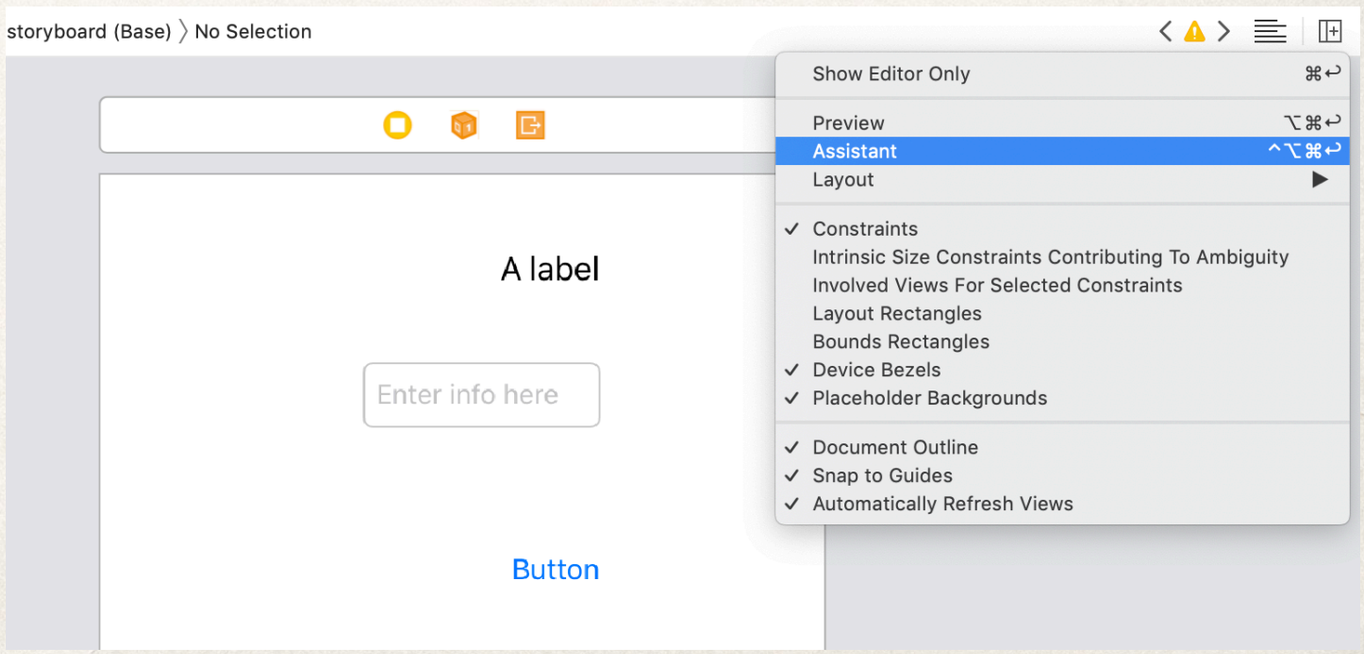
- Consider `IBOutlet` property:
  - `@IBOutlet weak var widgetName: widgetType!`
- `@IBOutlet` allows property to connect to Interface Builder object
- `weak` keyword allows property to have a nil value
  - Also not reference counted
- `widgetType!` specifies that type is an unwrapped optional
  - Will always have value after value is first set

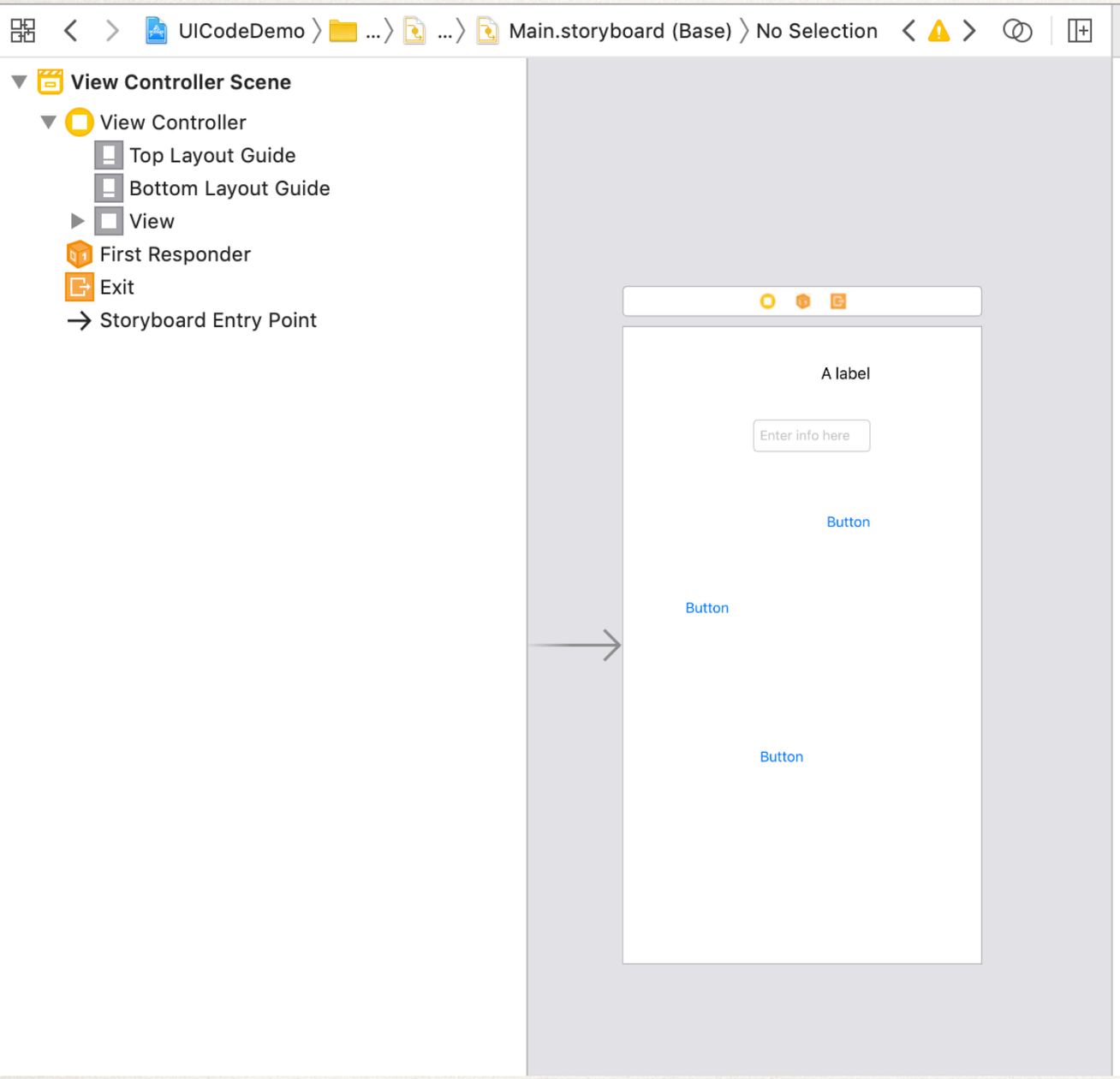
Connecting IB to Code...

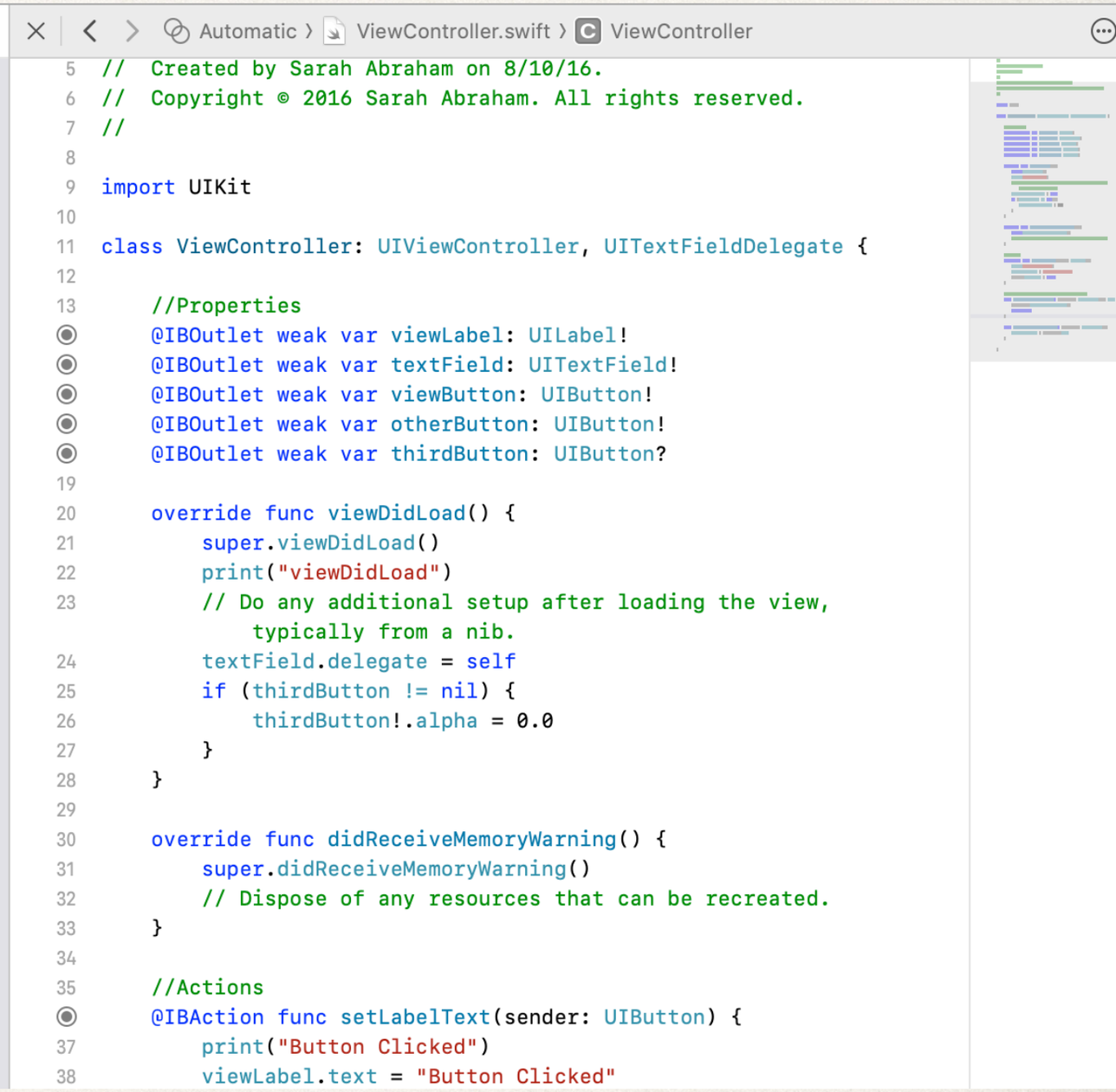
- View Controller in Interface Builder must be a custom view controller type



- Accessing Assistant will display both IB layout and associated View Controller code








```
5  //  Created by Sarah Abraham on 8/10/16.
6  //  Copyright © 2016 Sarah Abraham. All rights reserved.
7  //
8
9  import UIKit
10
11 class ViewController: UIViewController, UITextFieldDelegate {
12
13     //Properties
       @IBOutlet weak var viewLabel: UILabel!
       @IBOutlet weak var textField: UITextField!
       @IBOutlet weak var viewButton: UIButton!
       @IBOutlet weak var otherButton: UIButton!
       @IBOutlet weak var thirdButton: UIButton?
19
20     override func viewDidLoad() {
21         super.viewDidLoad()
22         print("viewDidLoad")
23         // Do any additional setup after loading the view,
               typically from a nib.
24         textField.delegate = self
25         if (thirdButton != nil) {
26             thirdButton!.alpha = 0.0
27         }
28     }
29
30     override func didReceiveMemoryWarning() {
31         super.didReceiveMemoryWarning()
32         // Dispose of any resources that can be recreated.
33     }
34
35     //Actions
       @IBAction func setLabelText(sender: UIButton) {
37         print("Button Clicked")
38         viewLabel.text = "Button Clicked"
```

Interface Builder and Xcode Editor

- Once IB and code are both displayed...
- Connect IB object to Xcode by:
  - Control-dragging from Interface Builder to line in Xcode editor
  - Typing property information in Xcode editor and connecting empty circle to the left of it back to object in Interface Builder
- Connections and properties can be changed/updated at any time
- Be aware that name changes or updating connections might break functionality
  - Requires manually reconnecting the object to its source code property

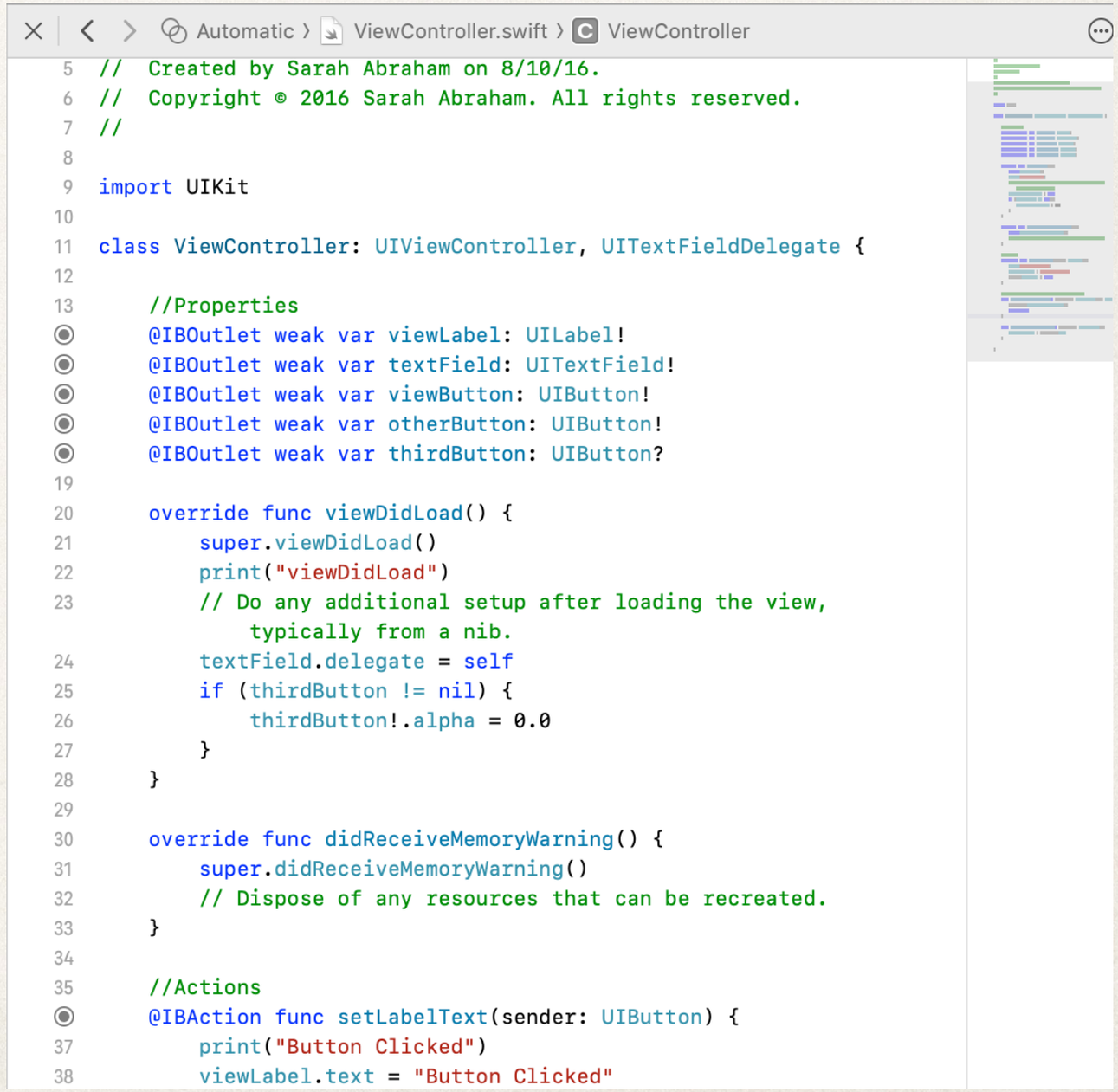
Instapoll: Creating Objects in IB

- Which of these is correct syntax for creating an object in Xcode to be connected to an object in Interface Builder?

- `@IBOutlet var widgetType: widgetName!`
- `@IBAction weak widgetName: widgetType!`
- `@IBOutlet var widgetType: widgetName?`
- `@IBOutlet weak var widgetName: widgetType!`

Create Actions for UI Elements

- Using Xcode editor
- Create actions for widgets
- Connect actions to widgets



What are Actions?

- Based on *event-driven programming paradigm*

- User input into interface trigger events in the app

- Actions methods link to events

  - Define behavior based on user input or system events

IBAction Example

- Consider `IBAction` method:
  - `@IBAction func buttonPressed(sender: UIButton) {}`
- `@IBAction` allows method to connect to IB object
- `func` declares it a function/method
- `sender` keeps track of which object triggered the action
  - Useful if object needs to be updated during the action!

Text Field Delegation Example

- Text fields use *delegation* to communicate with view controllers or other delegate objects

- Delegate can use information text fields provide to save data, clear the screen, or dismiss the keyboard etc

- Object must conform to UITextFieldDelegate protocol in order to be a delegate

UI Code Example

```swift
import UIKit

class ViewController: UIViewController, UITextFieldDelegate {

    //Properties
    @IBOutlet weak var viewLabel: UILabel!
    @IBOutlet weak var textField: UITextField!
    @IBOutlet weak var viewButton: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a nib.
        textField.delegate = self
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    //Actions
    @IBAction func setLabelText(sender: UIButton) {
        viewLabel.text = "Button Clicked"
    }

    //MARK: UITextFieldDelegate optional methods
    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return true
    }

    func textFieldDidEndEditing(_ textField: UITextField) {
        viewLabel.text = textField.text
    }

}
```