Developing Applications for iOS

Lab 1:
HelloWorld App

Radu Ionescu
raducu.ionescu@gmail.com
Faculty of Mathematics and Computer Science
University of Bucharest

Task 1

Task: Create a new application in Xcode called “HelloWorld”.

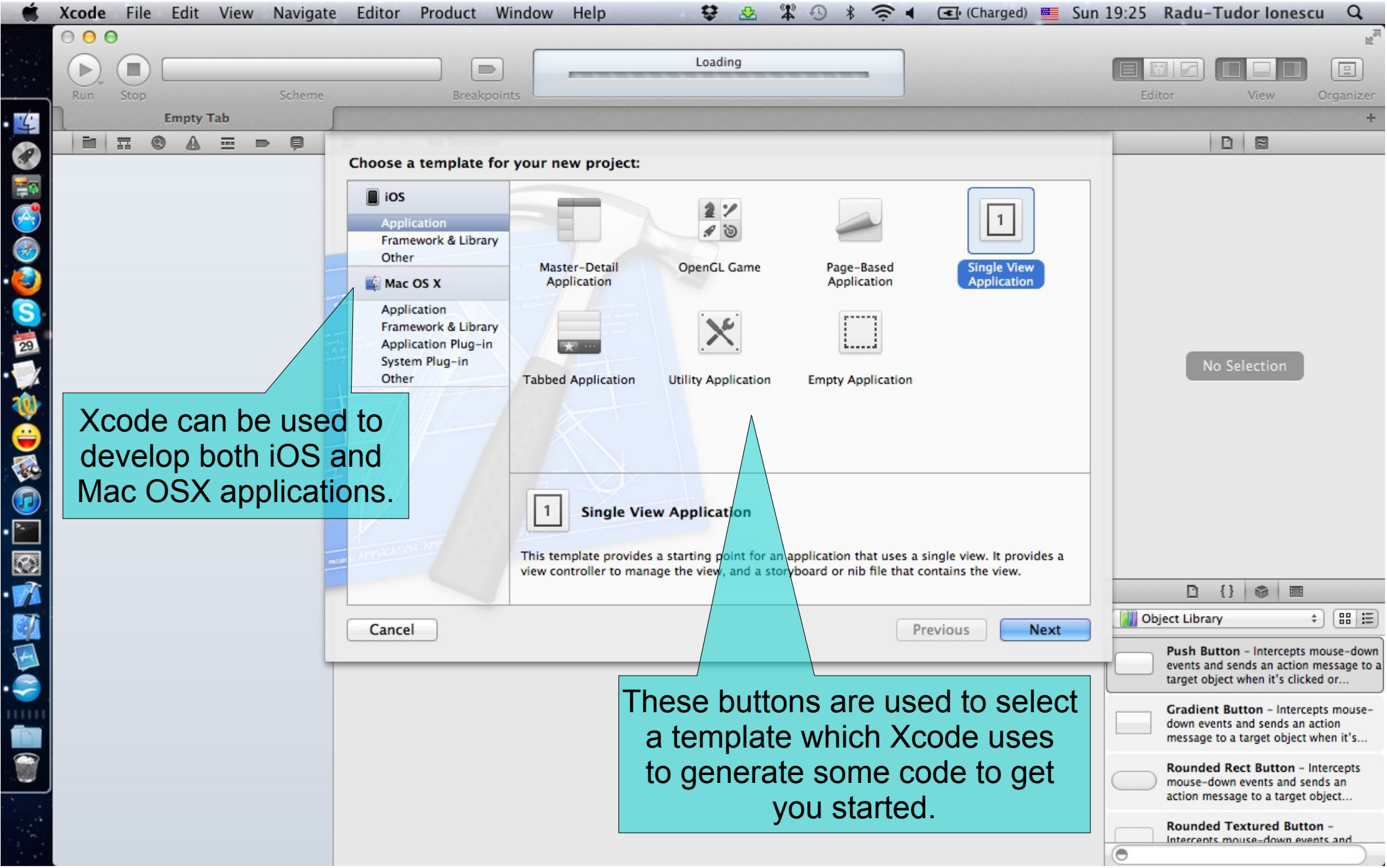
1. Launch Xcode and take a look at the splash window. On the left side you will find a few options: Create a new Xcode project, Connect to a repository, etc. Recent projects show up on the right side of this window.


Xcode File Edit View Navigate Editor Product Window Help
(Charged) Sun 18:38 Radu-Tudor Ionescu
Here is the Xcode menu. Like every Mac OSX application, Xcode displays it's menu on the upper side of the screen.
The Create New Project option.
Welcome to Xcode
Version 4.2 (4D199)
Create a new Xcode project
Start building a new Mac, iPhone or iPad application from one of the included templates
Connect to a repository
Use Xcode's integrated source control features to work with your existing projects
Learn about using Xcode
Explore the Xcode development environment with the Xcode 4 User Guide
Go to Apple's developer portal
Visit the Mac and iOS Dev Center websites at developer.apple.com
Recents
Calculator
~/iPhone/iOSLab/Apps
NextNomads
~/iPhone
SkinScan
~/iPhone
PlacesAR
EyeCare
iGeShopping
GeoDeals
~/iPhone/iOSLab/Apps
ListAdder
~/iPhone/Samples
InsertRowTest
~/iPhone/Samples
Last opened Today 18:32 PM
As you create projects, they will appear in the Recents Projects section (right here).
Open Other...
Show this window when Xcode launches
Cancel
Open

Task 1

Task: Create a new application in Xcode called “HelloWorld”.

2. Click on the Create a new Xcode project option. If you don't see the splash window, you should go to “File > New > New Project...” in Xcode menu.



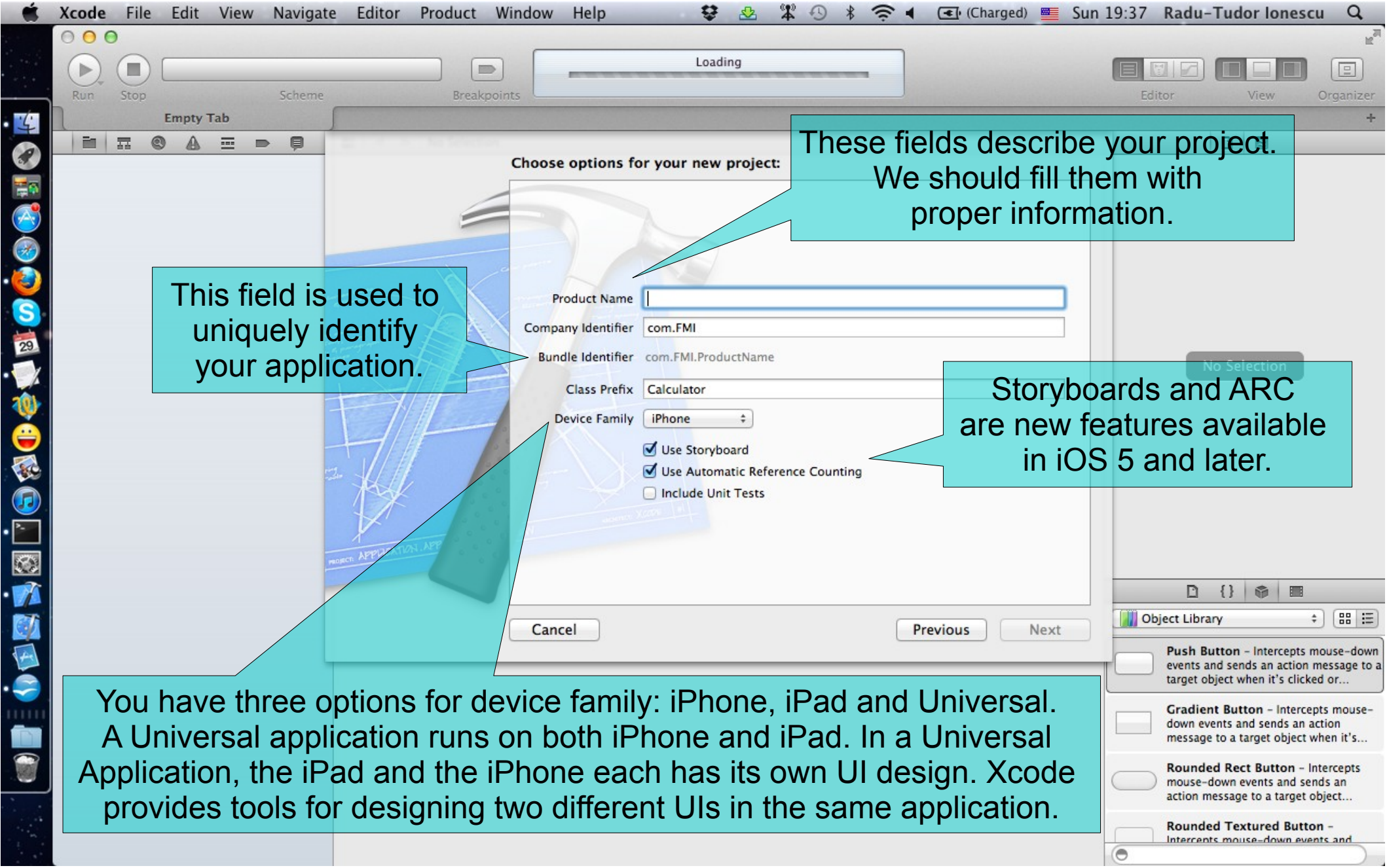
Xcode can be used to develop both iOS and Mac OSX applications.

These buttons are used to select a template which Xcode uses to generate some code to get you started.

Task 1

Task: Create a new application in Xcode called “HelloWorld”.

3. Select the Single View Application template and click Next.



These fields describe your project. We should fill them with proper information.

This field is used to uniquely identify your application.

Storyboards and ARC are new features available in iOS 5 and later.

You have three options for device family: iPhone, iPad and Universal. A Universal application runs on both iPhone and iPad. In a Universal Application, the iPad and the iPhone each has its own UI design. Xcode provides tools for designing two different UIs in the same application.

Task 1

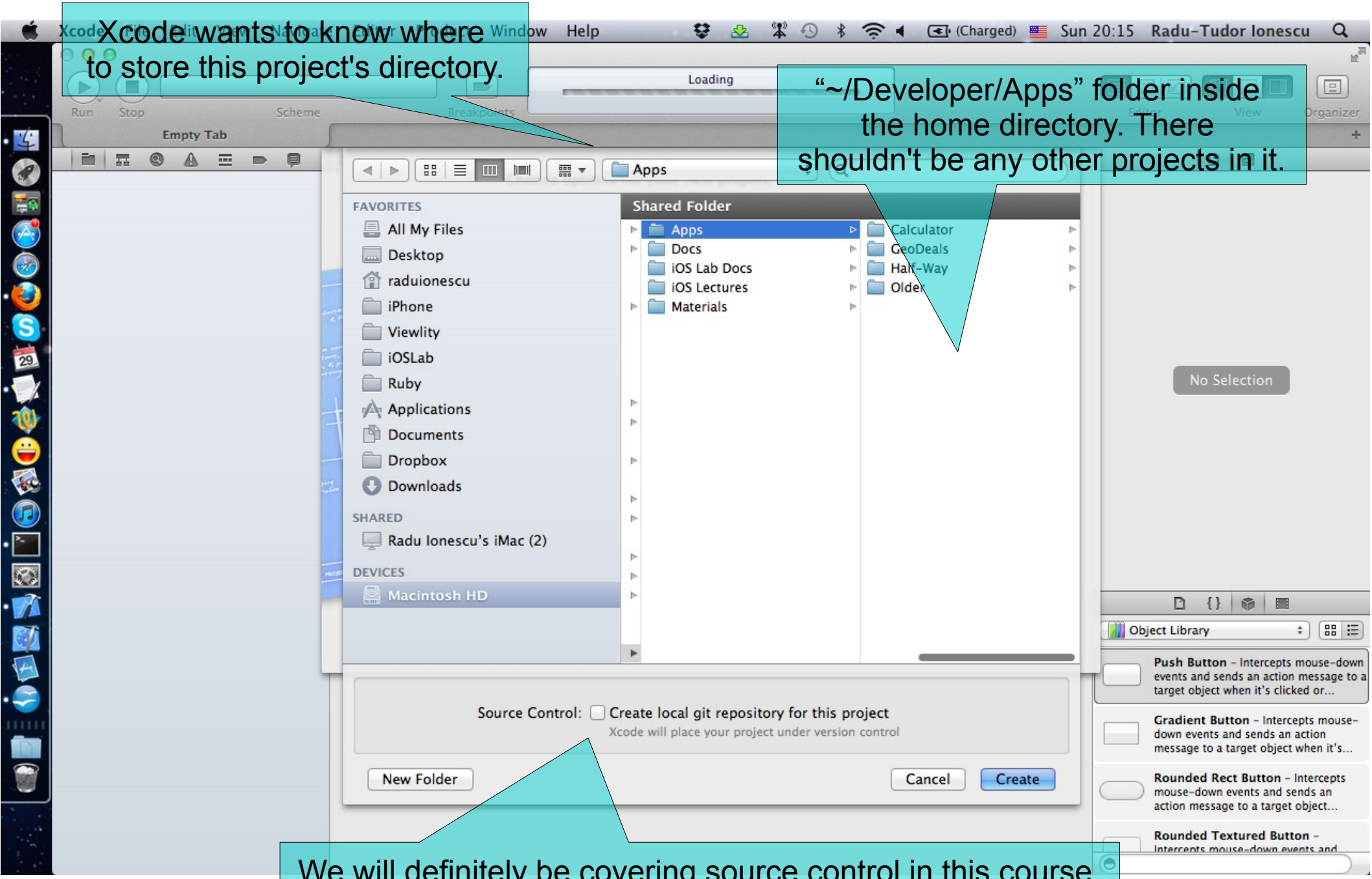
Task: Create a new application in Xcode called “HelloWorld”.

4. Type in “HelloWorld” for the Product Name.
5. Type in “com.FMI.FirstName.LastName” for the Company Identifier. Notice how Bundle Identifier changes as you type. You should obtain something like “com.FMI.Radu.Ionescu.HelloWorld” as your bundle identifier. Using an entity's reverse DNS lookup string is a pretty good way to get a unique identifier.
6. Type in “HelloWorld” for Class Prefix. We don't want the names of the classes generated by the template to be too generic. That's why we specify this prefix. Usually we use the name of the application for this prefix. In fact, older versions of Xcode would automatically do this whether we wanted it or not.
7. Select “iPhone” for Device Family. Our first application is going to be for the iPhone (not iPad).

Task 1

Task: Create a new application in Xcode called “HelloWorld”.

8. Check “Use Storyboard”. Storyboards are a new (iOS 5) way to organize your MVC's Views. We are going to use them.
9. Check “Use Automatic Reference Counting”. ARC is a fantastic upgrade to the compiler (in iOS 5) which causes it to generate all the code necessary to manage the memory allocation of objects. We definitely want ARC to be on!
10. We won't be creating Unit Tests for our first application so we are going to leave the “Include Unit Tests” option unchecked.
11. Click Next.


Xcode wants to know where to store this project's directory.
"~/Developer/Apps" folder inside the home directory. There shouldn't be any other projects in it.
We will definitely be covering source control in this course. But not for this first project, so leave this switch turned off.
Apps
Shared Folder
FAVORITES
All My Files
Desktop
raduionescu
iPhone
Viewlity
iOSLab
Ruby
Applications
Documents
Dropbox
Downloads
SHARED
Radu Ionescu's iMac (2)
DEVICES
Macintosh HD
Apps
Docs
iOS Lab Docs
iOS Lectures
Materials
Calculator
GeoDeals
Half-Way
Older
Source Control:
Create local git repository for this project
Xcode will place your project under version control
New Folder
Cancel
Create
No Selection
Object Library
Push Button - Intercepts mouse-down events and sends an action message to a target object when it's clicked or...
Gradient Button - Intercepts mouse-down events and sends an action message to a target object when it's...
Rounded Rect Button - Intercepts mouse-down events and sends an action message to a target object...
Rounded Textured Button -

Task 1

Task: Create a new application in Xcode called “HelloWorld”.

12. Navigate to “~/Developer/Apps” folder inside the home directory. There shouldn't be any other projects in it.

13. Click Create to create your project directory inside the “~/Developer/Apps” folder.

Task 2

Task: Run the application in iPhone Simulator.

1. Check out the general structure of the project window and identify the important sections: the Toolbar, the Tab bar, the Project Navigator, the Details Editor and the Utilities Area.

Toolbar – Contains buttons to quickly build your application, select your scheme, show/hide sections.

Tab bar – Each tab can save your current work settings or restore previous work settings.

Project Navigator

It shows all the files in your project in a hierarchical arrangement of folders. The arrangement of folders is conceptual, it does not necessarily match what's in the file system. This area can also show symbols, search results, breakpoints, issues, etc. (see the icons at the top).

Details (Editor)

It shows the content of the selected file in Project Navigator. If the file is a code file (.h or .m), this area will show the source code. If the file is a view (.xib) or a storyboard it will show up in this area using Interface Builder (which is integrated in Xcode). You can also view the content of other iOS specific files (such as .plists) or almost every type of file that can be included in your project (such as images, plain text, etc.).

Utilities

The top part of this area shows information (identity, attributes, help, connections, dimensions) about the currently selected thing (file, object, etc.) at the left.

The bottom is a library of items (objects, code snippets, file templates).

Task 2

Task: Run the application in iPhone Simulator.

2. View the content of the “HelloWorldAppDelegate.h” file. This file contains the declaration of instance variables, public properties and public methods of the `HelloWorldAppDelegate` class. In Objective-C we call this part the class **interface**.

Notice the class conforms to the `UIApplicationDelegate` protocol. This protocol declares methods that are implemented by the delegate of the singleton `UIApplication` object. These methods provide you with information about key events in an application’s execution such as when it finished launching, when it is about to be terminated, when memory is low, and when important changes occur. Implementing these methods gives you a chance to respond to these system events and respond appropriately.

Comma separated list of protocols.
`HelloWorldAppDelegate` conforms to a single protocol: `UIApplicationDelegate`.

Class name

Superclass

```
//
//  HelloWorldAppDelegate.h
//  HelloWorld
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldAppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@end
```

The `@interface` compiler directive indicates that this file contains the interface of `HelloWorldAppDelegate`.

The application window. Each iPhone application has a singleton `UIApplication` object that has a reference to the application window. All the content of the application must be displayed in this window. However, the window usually contains a hierarchy of views and layers.
Our application window contains a single view.

Task 2

Task: Run the application in iPhone Simulator.

3. To read more about the `UIApplicationDelegate` protocol hold down the option key and click on the `UIApplicationDelegate`. A pop-up window with detailed information should appear on the screen.

Hold down the option key and click here to see detailed information about this item.

Double-click here to open the `UIApplicationDelegate` full API documentation.

```
//
//  HelloWorldAppDelegate.h
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldAppDelegate : UIResponder <UIApplicationDelegate>

@property (strong, nonatomic) UIWindow *window;

@end
```

UIApplicationDelegate

Name: UIApplicationDelegate

Availability: iOS (2.0 and later)

Abstract: The UIApplicationDelegate protocol declares methods that are implemented by the delegate of the singleton UIApplication object. These methods provide you with information about key events in an application's execution such as when it finished launching, when it is about to be terminated, when memory is low, and when important changes occur. Implementing these methods gives you a chance to respond to these system events and respond appropriately.

Declared In: UIApplication.h

Reference: UIApplicationDelegate Protocol Reference

Related Documents: Local and Push Notification Programming Guide, iOS App Programming Guide

Task 2

Task: Run the application in iPhone Simulator.

4. To open the `UIApplicationDelegate` API reference in Organizer double-click on the link displayed on the pop-up window. Read the full documentation of the `UIApplicationDelegate`.

Make sure you are reading the right version (iOS 5.0).

This is the Organizer window.

You are on the Documentation tab.

Notice that Organzier can also give you information about the connected devices (installed certificates, log information, installed apps, etc.), repositories (integrated git control), projects (create and manage snapshots) and archives (manage and share ad-hoc builds or submit AppStore builds).

There are two versions of each class: one for iOS 5.0 and one for iOS 4.3.

Scroll down and read all the information here.

Devices Repositories Projects Archives Documentation

iOS 5.0 Library › Data Management › Event Handling › UIApplicationDelegate Protocol Reference

Next

UIApplication

Reference
Show 49 More Results
- UIApplication
- UIApplication
- UIApplicationDelegate
- UIApplicationDelegate
- UIApplicationDid...ctiveNotification
- UIApplicationDid...ctiveNotification
- UIApplicationDid...rameNotification
- UIApplicationDid...rameNotification
- UIApplicationDid...ationNotification
- UIApplicationDid...ationNotification

System Guides
Show 42 More Results
- Opening Supported File Types
- Previewing and Opening Files
- Tips for Allocating Memory
- Opening Supported File Types
- Previewing and Opening Files

Tools Guides
0 Results

Sample Code
Show 7 More Results
- GLEssentials
- Sampler Unit Pre...oadPresetDemo)
- avTouch
- avTouch
- AppPrefs
- AppPrefs
- RosyWriter
- GKRocket
- GKRocket

UIApplicationDelegate Protocol Reference

Conforms to	NSObject
Framework	/System/Library/Framework
Availability	Available in iOS 2.0 and later
Declared in	UIApplication.h
Companion guides	iOS App Programming Guide Local and Push Notification Programming Guide
Related Sample Code	GLEssentials GLSLRenderTest International Mountains MyMixer SpeakHere TableViewSuite

Overview

The `UIApplicationDelegate` protocol declares methods that are implemented by the delegate of the singleton `UIApplication` object. These methods provide you with information about key events in an application's execution such as when it finished launching, when it is about to be terminated, when memory is low, and when important changes occur. Implementing these methods gives you a chance to respond to these system events and respond appropriately.

One of the main jobs of the application delegate is to track the state transitions the application goes through while it is running. Prior to iOS 4.0, applications were either active, inactive, or not running. In iOS 4.0 and later, applications can also be running in the background or suspended. All of these transitions require a response from your application to ensure that it is doing the right thing. For example, a background application would need to stop updating its user interface. You provide the response to these transitions using the methods of the application delegate.

Launch time is also a particularly important point in an application's life cycle. In addition to the user launching an application by tapping its icon, an application can be launched in order to respond to a specific type of event. For example, it could be launched in response to an incoming push notification, it could be asked to open a file, or it could be launched to handle some background event that it had requested. In all of these cases, the options dictionary passed to the `application:didFinishLaunchingWithOptions:` method provides information about the reason for the launch.

In situations where the application is already running, the methods of the application delegate are called in response to key changes. Although the methods of this protocol are optional, most or all of them should be implemented.

Task 2

Task: Run the application in iPhone Simulator.

5. View the content of the “HelloWorldAppDelegate.m” file. This is the implementation of `HelloWorldAppDelegate`. Notice the superclass declaration is missing. Check if the methods declared by the `UIApplicationDelegate` are implemented there.

The `@implementation` compiler directive indicates that this file contains the implementation of `HelloWorldAppDelegate`.

Superclass declaration missing.

```
#import "HelloWorldAppDelegate.h"

@implementation HelloWorldAppDelegate

@synthesize window = _window;

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)l
{
    // Override point for customization after application launch.
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application
{
    /*
     Sent when the application is about to move from active to inactive state. This can occur fo
     Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame r
     */
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    /*
     Use this method to release shared resources, save user data, invalidate timers, and store e
     If your application supports background execution, this method is called instead of applica
     */
}

- (void)applicationWillEnterForeground:(UIApplication *)application
{
    /*
     Called as part of the transition from the background to the inactive state; here you can un
     */
}

- (void)applicationDidBecomeActive:(UIApplication *)application
{
    /*
     Restart any tasks that were paused (or not yet started) while the application was inactive.
     */
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    /*
     Called when the application is about to terminate.
     Save data if appropriate.
     See also applicationDidEnterBackground:.
     */
}
```

Task 2

Task: Run the application in iPhone Simulator.

6. Inspect the Toolbar items. Run the application in iPhone Simulator.

Run the application from here.

When you run the application in debug mode, you can set breakpoints. You can enable/disable all breakpoints from here.

Show/Hide Navigator.

This is the "scheme chooser". It lets you choose where to run your application. For example, the iPhone Simulator, iPad Simulator or on a device. Make sure you choose iPhone 5.0 Simulator.

Show/Hide Utilities.

Open Organizer.

If you press and hold this Run button, other run options will be available.

When it starts building the application, the Stop button becomes active.

This area provides information about the project status. It lets know that Xcode is building the application.

The Details Editor can be adjusted to display content in 3 modes: Standard (selected right now), Assistant (we'll cover it in a few moments) and Version (to view more versions of the same file).

Xcode File Edit View Navigate Editor Product Window Help (Charged) Tue 18:03 Radu-Tudor Ionescu

HelloWorld.xcodeproj — HelloWorldAppDelegate.m

Run Stop Scheme: HelloWorld 〉 iPhone 5.0 Simulator Breakpoints Building HelloWorld: HelloWorld Editor View Organizer

HelloWorldAppDelegate.m

HelloWorldViewController.h
HelloWorldViewController.m
Supporting Files
HelloWorld-Info.plist
InfoPlist.strings
main.m
HelloWorld-Prefix.pch
Frameworks
Products
HelloWorld.app

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)l
{
    // Override point for customization after application launch.
    return YES;
}

- (void)applicationWillResignActive:(UIApplication *)application
{
    /*
     Sent when the application is about to move from active to inactive state. This can occur fo
     Use this method to pause ongoi
     */
}

- (void)applicationDidEnterBackground:(UIApplication *)application
{
    /*
     Use this method to release shared resources, save user data, invalidate timers, and store e
     If your application supports background execution, this method is called instead of applica
     */
}

- (void)applicationWillEnterForeground:(UIApplication *)application
{
    /*
     Called as part of the transition from the background to the inactive state; here you can un
     */
}

- (void)applicationDidBecomeActive:(UIApplication *)application
{
    /*
     Restart any tasks that were paused (or not yet started) while the application was inactive.
     */
}

- (void)applicationWillTerminate:(UIApplication *)application
{
    /*
     Called when the application is about to terminate.
     Save data if appropriate.
     See also applicationDidEnterBackground:.
```

Identity and Type
File Name: HelloWorldAppDelegate.m
File Type: Default - Objective-C so...
Location: Relative to Group
HelloWorldAppDelegate.m
Full Path: /Users/raduionescu/iPhone/iOSLab/Apps/HelloWorld/HelloWorld/HelloWorldAppDelegate.m

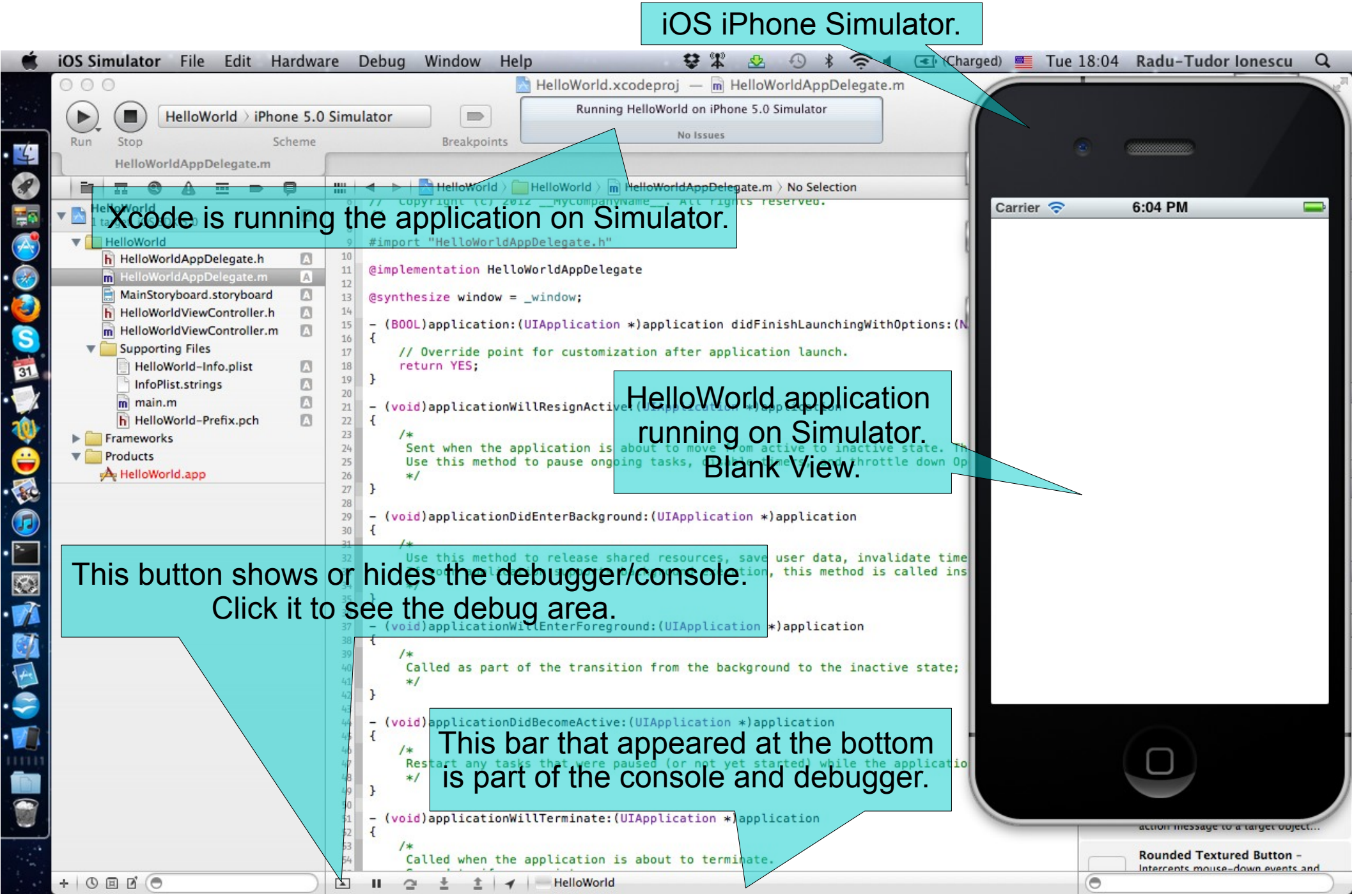
Text Settings
Text Encoding: Default - Unicode (UTF-8)

Object Library
Push Button - Intercepts mouse-down events and sends an action message to a target object when it's clicked or...
Gradient Button - Intercepts mouse-down events and sends an action message to a target object when it's...
Rounded Rect Button - Intercepts mouse-down events and sends an action message to a target object...
Rounded Textured Button -

Task 2

Task: Run the application in iPhone Simulator.

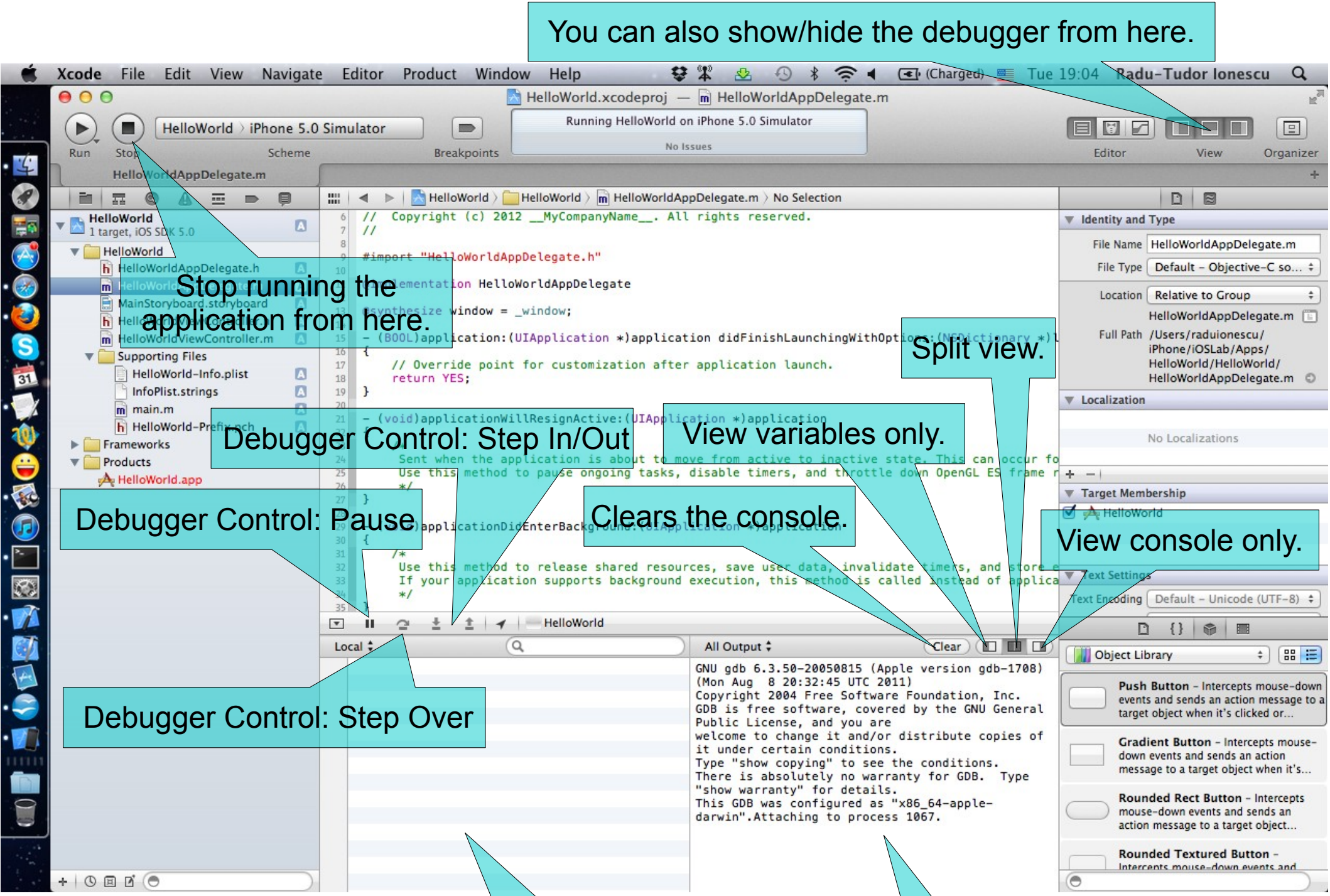
7. Open the debugger/console.


iOS iPhone Simulator.
Xcode is running the application on Simulator.
HelloWorld application running on Simulator. Blank View.
This button shows or hides the debugger/console. Click it to see the debug area.
This bar that appeared at the bottom is part of the console and debugger.
iOS Simulator File Edit Hardware Debug Window Help
Tue 18:04 Radu-Tudor Ionescu
HelloWorld.xcodeproj — HelloWorldAppDelegate.m
HelloWorld 〉 iPhone 5.0 Simulator
Run Stop Scheme Breakpoints
Running HelloWorld on iPhone 5.0 Simulator
No Issues
HelloWorldAppDelegate.m
HelloWorldAppDelegate.h
HelloWorldAppDelegate.m
MainStoryboard.storyboard
HelloWorldViewController.h
HelloWorldViewController.m
Supporting Files
HelloWorld-Info.plist
InfoPlist.strings
main.m
HelloWorld-Prefix.pch
Frameworks
Products
HelloWorld.app
#import "HelloWorldAppDelegate.h"
@implementation HelloWorldAppDelegate
@synthesize window = _window;
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(N
{
// Override point for customization after application launch.
return YES;
}
- (void)applicationDidEnterBackground:(UIApplication *)application
- (void)applicationWillEnterForeground:(UIApplication *)application
Called as part of the transition from the background to the inactive state;
- (void)applicationDidBecomeActive:(UIApplication *)application
- (void)applicationWillTerminate:(UIApplication *)application
Called when the application is about to terminate.
Carrier 6:04 PM
Rounded Textured Button -

Task 2

Task: Run the application in iPhone Simulator.

8. Stop running the application. Notice that the debug area automatically disappears when you stop running.


You can also show/hide the debugger from here.
Stop running the application from here.
Split view.
Debugger Control: Step In/Out
View variables only.
Debugger Control: Pause
Clears the console.
View console only.
Debugger Control: Step Over
Variables View
Console

Task 3

Task: Add a label to display our greeting message to the user.

1. Open the MainStoryboard.storyboard file.

All iOS applications use the MVC design pattern.
Our first application will only have a view and a controller.
We will discuss the MVC design pattern in detail later.

Our MVC's View is inside MainStoryboard.storyboard.

HelloWorldViewController.[hm] is the code for our MVC's Controller.

Let's open up and look at our MVC's View by clicking on MainStoryboard.storyboard.

Xcode File Edit View Navigate Editor Product Window Help Wed 14:41 Radu-Tudor Ionescu

HelloWorldAppDelegate.m

HelloWorld 1 target, iOS SDK 5.0
- HelloWorld
  - HelloWorldAppDelegate.h
  - HelloWorldAppDelegate.m
  - MainStoryboard.storyboard
  - HelloWorldViewController.h
  - HelloWorldViewController.m
  - Supporting Files
    - HelloWorld-Info.plist
    - InfoPlist.strings
    - main.m
    - HelloWorld-Prefix.pch
- Frameworks
- Products
  - HelloWorld.app

```
6   //  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
7   //
8
9   #import "HelloWorldAppDelegate.h"
10
11  @implementation HelloWorldAppDelegate
12
13  @synthesize window = _window;
14
15  - (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)l
16  {
17      /* Override point for customization after application launch.
18      return YES;
19  }
20  - (void)applicationWillResignActive:(UIApplication *)application
21  {
22      /*
23       Sent when the application is about to move from active to inactive state. This can occur fo
24       Use this method to pause ongoing tasks, disable timers, and throttle down OpenGL ES frame r
25       */
26  }
27
28  - (void)applicationDidEnterBackground:(UIApplication *)application
29  {
30      /*
31       Use this method to release shared resources, save user data, invalidate timers, and store e
32       If your application supports background execution, this method is called instead of applica
33       */
34  }
35
36  - (void)applicationWillEnterForeground:(UIApplication *)application
37  {
38      /*
39       Called as part of the transition from the background to the inactive state; here you can un
40       */
41  }
42
43  - (void)applicationDidBecomeActive:(UIApplication *)application
44  {
45      /*
46       Restart any tasks that were paused (or not yet started) while the application was inactive.
47       */
48  }
49
50  - (void)applicationWillTerminate:(UIApplication *)application
51  {
52      /*
53       Called when the application is about to terminate.
54       Save data if appropriate.
55       See also applicationDidEnterBackground:
```

Identity and Type
- File Name: HelloWorldAppDelegate.m
- File Type: Default - Objective-C so...
- Location: Relative to Group
- HelloWorldAppDelegate.m
- Full Path: /Users/raduionescu/iPhone/iOSLab/Apps/HelloWorld/HelloWorld/HelloWorldAppDelegate.m

Localization
- No Localizations

Target Membership
- HelloWorld

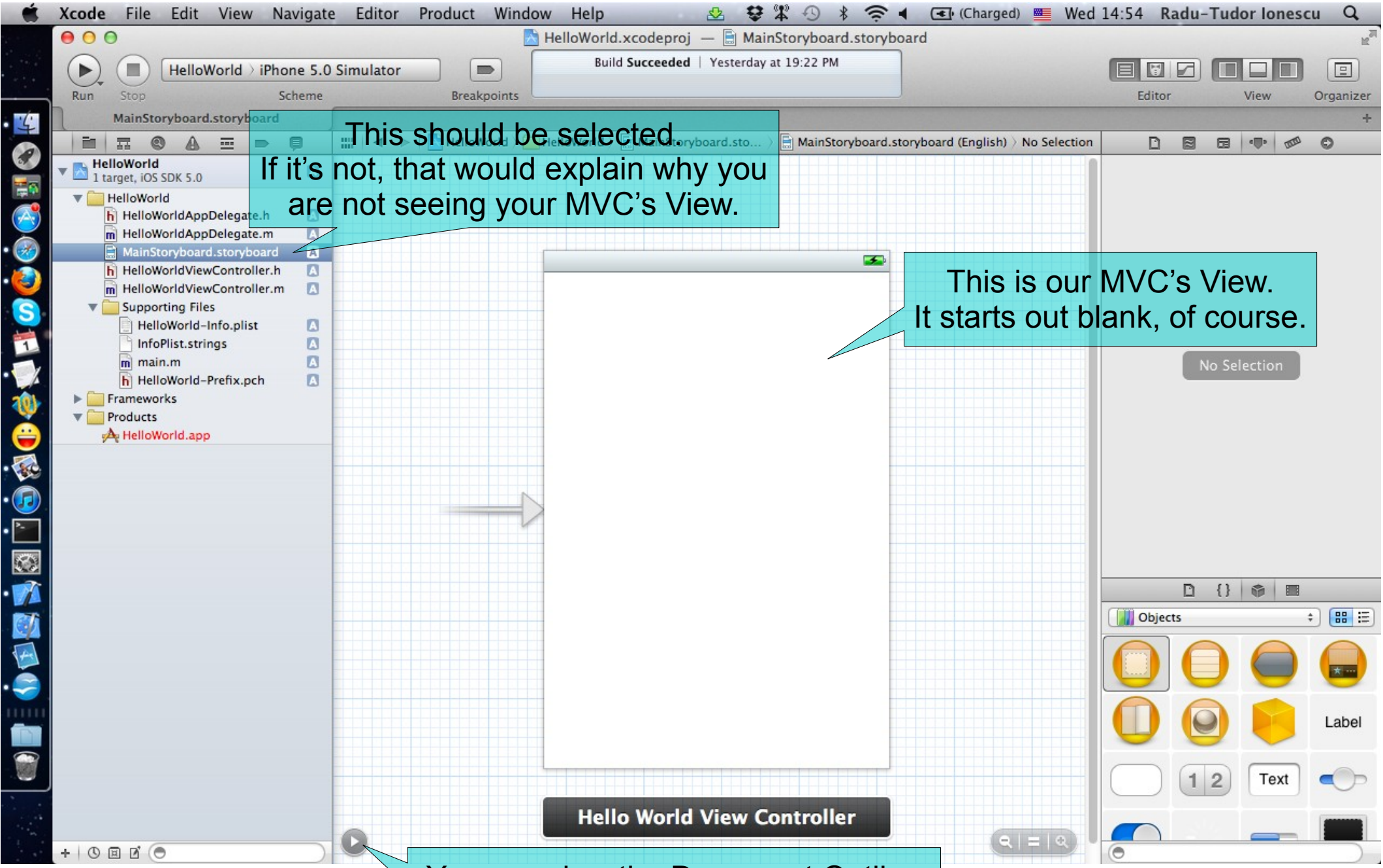
Text Settings
- Text Encoding: Default - Unicode (UTF-8)

Objects
- Label
- Text

Task 3

Task: Add a label to display our greeting message to the user.

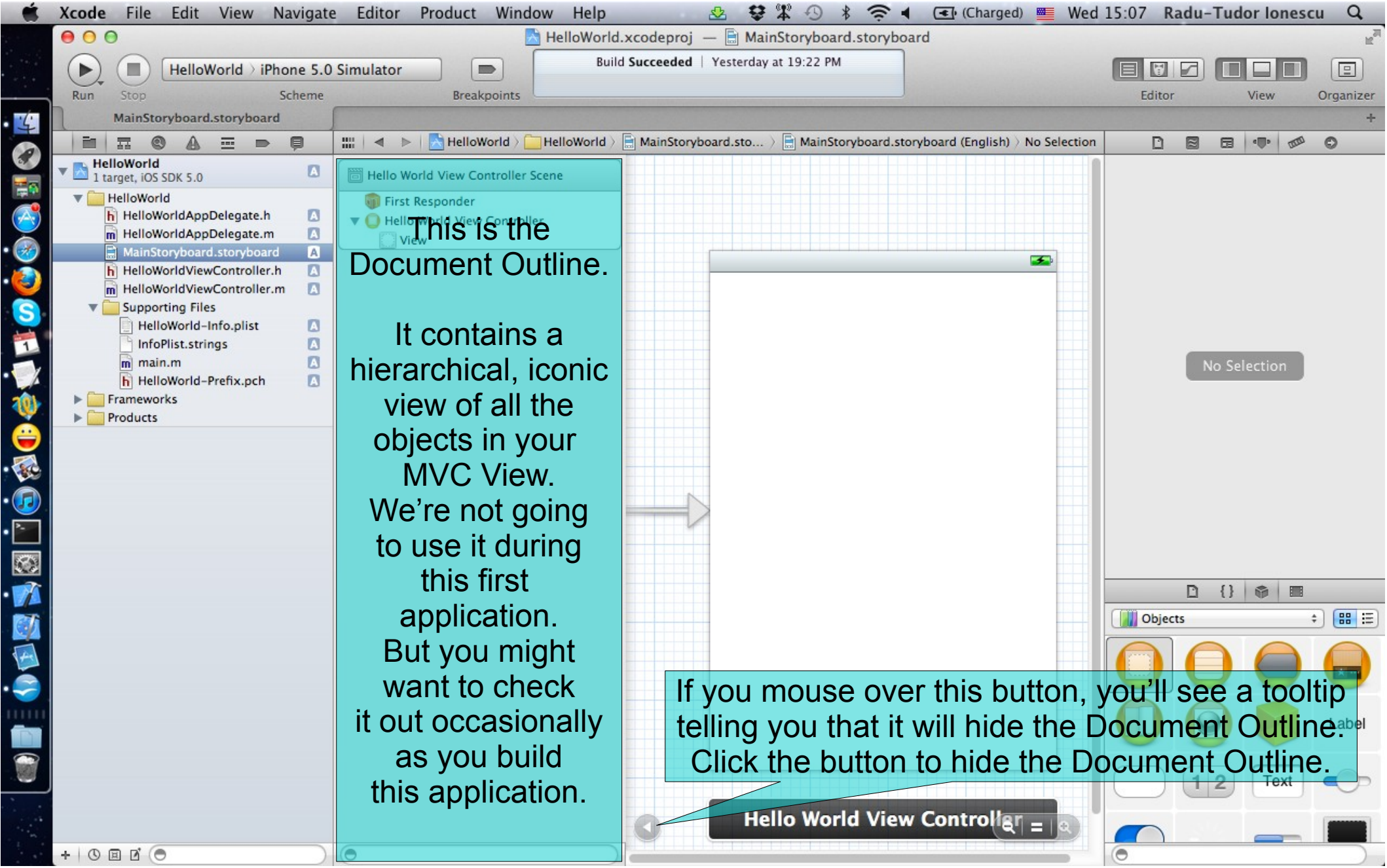
2. Open the Document Outline.


Xcode File Edit View Navigate Editor Product Window Help
Wed 14:54 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Run Stop Scheme Breakpoints
HelloWorld iPhone 5.0 Simulator
Build Succeeded | Yesterday at 19:22 PM
Editor View Organizer
MainStoryboard.storyboard
MainStoryboard.storyboard (English) No Selection
HelloWorld
1 target, iOS SDK 5.0
HelloWorld
HelloWorldAppDelegate.h
HelloWorldAppDelegate.m
MainStoryboard.storyboard
HelloWorldViewController.h
HelloWorldViewController.m
Supporting Files
HelloWorld-Info.plist
InfoPlist.strings
main.m
HelloWorld-Prefix.pch
Frameworks
Products
HelloWorld.app
This should be selected.
If it's not, that would explain why you are not seeing your MVC's View.
This is our MVC's View.
It starts out blank, of course.
No Selection
Objects
Label
Text
Hello World View Controller
You can view the Document Outline by clicking this button.

Task 3

Task: Add a label to display our greeting message to the user.

3. Hide back the Document Outline.


Xcode File Edit View Navigate Editor Product Window Help
Wed 15:07 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Run Stop Scheme Breakpoints
HelloWorld iPhone 5.0 Simulator
Build Succeeded | Yesterday at 19:22 PM
Editor View Organizer
MainStoryboard.storyboard
HelloWorld
1 target, iOS SDK 5.0
HelloWorldAppDelegate.h
HelloWorldAppDelegate.m
MainStoryboard.storyboard
HelloWorldViewController.h
HelloWorldViewController.m
Supporting Files
HelloWorld-Info.plist
InfoPlist.strings
main.m
HelloWorld-Prefix.pch
Frameworks
Products
Hello World View Controller Scene
First Responder
View
No Selection
Objects
Text
Hello World View Controller
This is the Document Outline.
It contains a hierarchical, iconic view of all the objects in your MVC View.
We're not going to use it during this first application.
But you might want to check it out occasionally as you build this application.
If you mouse over this button, you'll see a tooltip telling you that it will hide the Document Outline.
Click the button to hide the Document Outline.

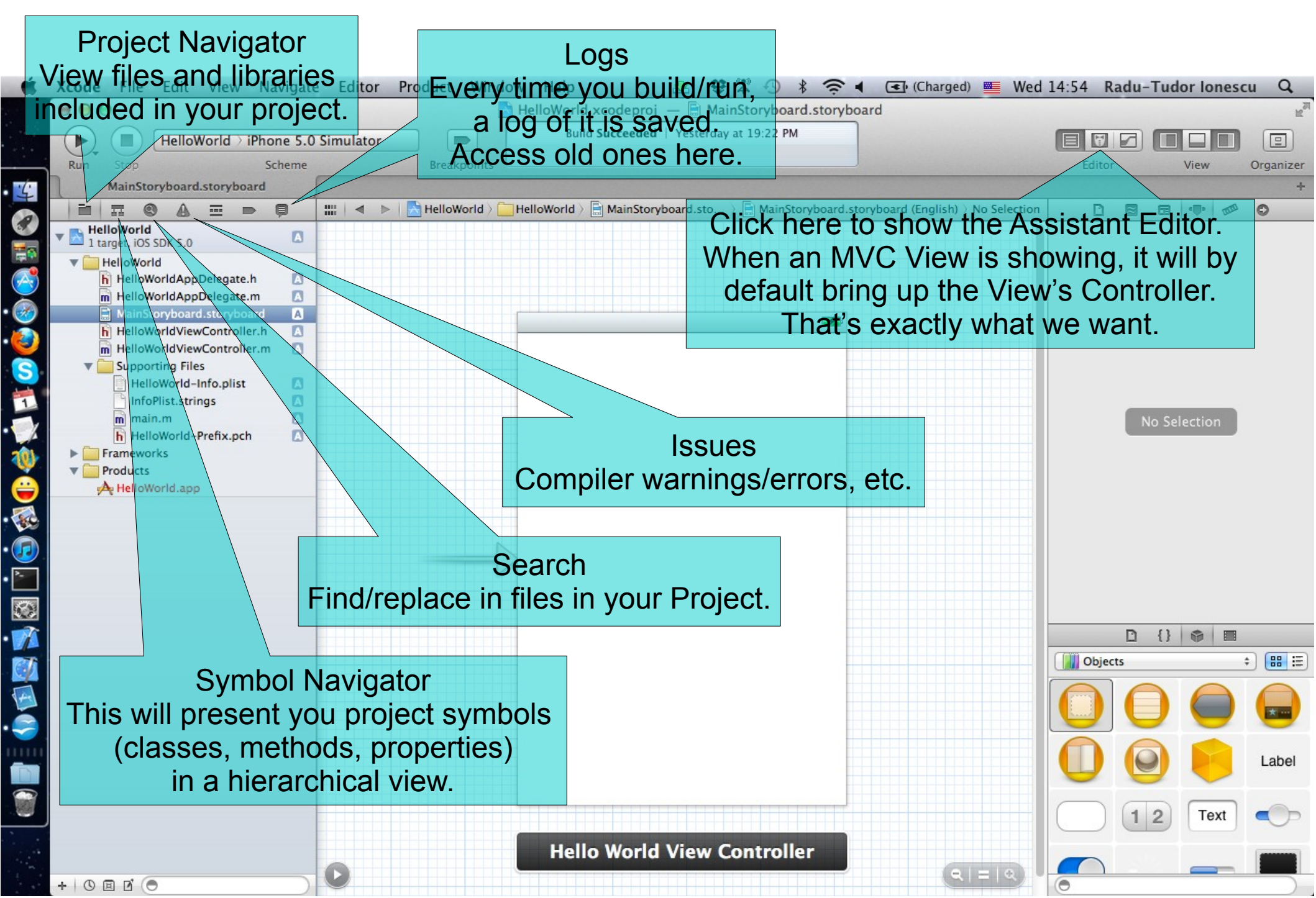
Task 3

Task: Add a label to display our greeting message to the user.

4. Open the Assistant Editor.

We need to see our MVC Controller now. But we still want our MVC View on screen at the same time. The way to have two things on the screen at once is to use the Assistant Editor. It is shown/hidden using the “butler” icon from the toolbar.

5. Check out the Navigator's bar items.


Project Navigator
View files and libraries included in your project.
Logs
Every time you build/run, a log of it is saved.
Access old ones here.
Click here to show the Assistant Editor.
When an MVC View is showing, it will by default bring up the View's Controller.
That's exactly what we want.
Issues
Compiler warnings/errors, etc.
Search
Find/replace in files in your Project.
Symbol Navigator
This will present you project symbols (classes, methods, properties) in a hierarchical view.
Hello World View Controller

Task 3

Task: Add a label to display our greeting message to the user.

6. Hide the Project Navigator.

When the Assistant Editor is in Automatic mode, it will always be trying to put something sensible up in the right-hand side of the Editor.

We don't need the Navigator at the far left, so let's hide it by turning this button off.

ALT-clicking on our Controller's header file (HelloWorldViewController.h) would also have brought it up in the Assistant Editor, but it would have taken the Assistant Editor out of "Automatic mode".

Task 3

Task: Add a label to display our greeting message to the user.

7. Study the header file of the HelloWorldViewController class.

Let's make even more room for our code on the right by dragging this center bar to the left.

UIKit.h imports all the iOS user-interface classes. `#import` is like `#include`, but better.

Notice the `@interface` - `@end` syntax.

Here is the header (.h) file for our MVC Controller. It contains its public methods and properties and also defines its superclass public methods and properties (all Controllers in iOS inherit from `UIViewController`).

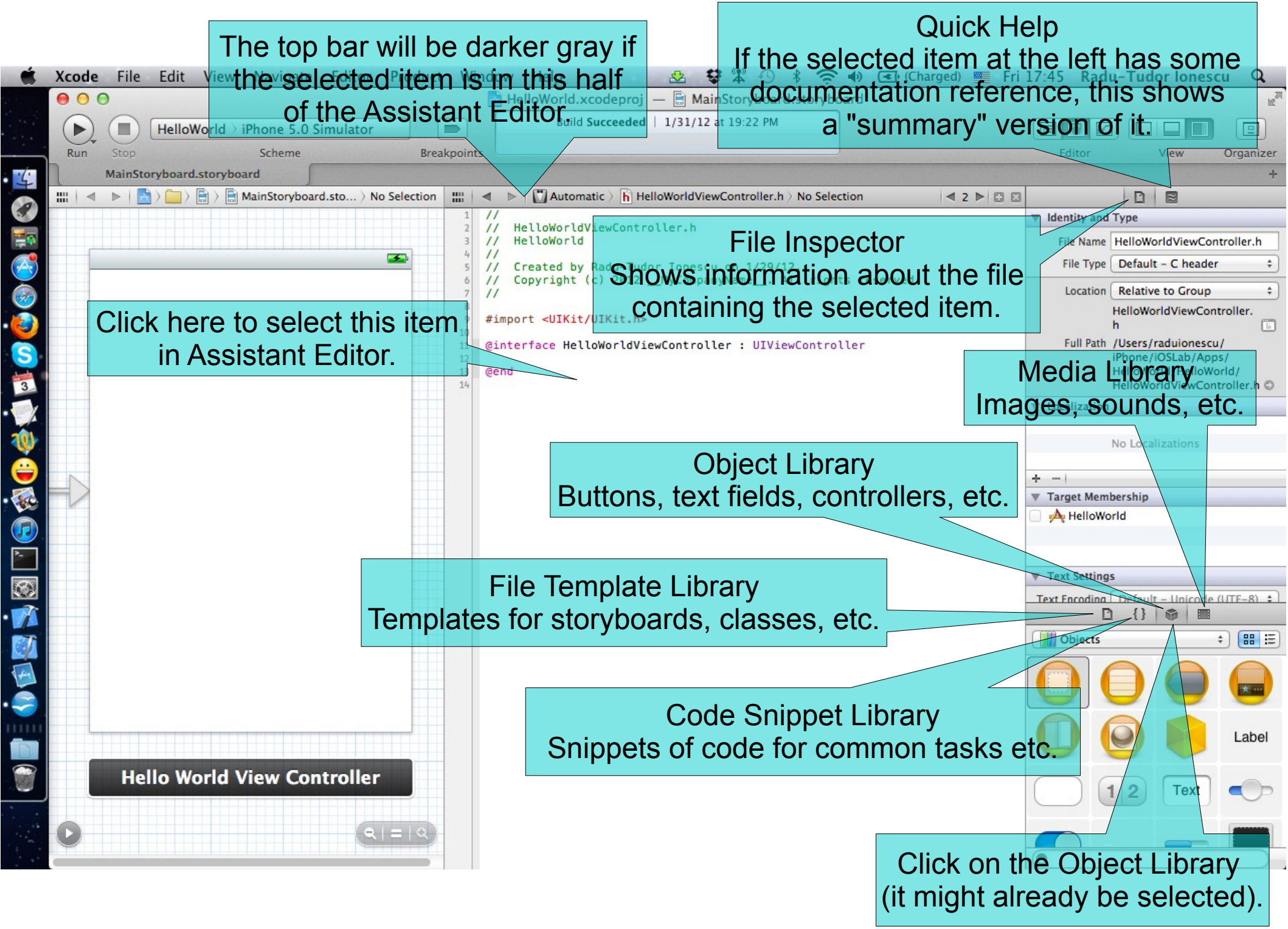
```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController

@end
```

Task 3

Task: Add a label to display our greeting message to the user.

8. Click on the header file. Study the items in the Utilities area.

9. Select the Object Library in Utilities area. We're going to start building the user-interface in our MVC View. To do that, we'll need a text label, a text field and a button. We get those from the Object Library.


The top bar will be darker gray if the selected item is in this half of the Assistant Editor.
Quick Help
If the selected item at the left has some documentation reference, this shows a "summary" version of it.
File Inspector
Shows information about the file containing the selected item.
Click here to select this item in Assistant Editor.
Media Library
Images, sounds, etc.
Object Library
Buttons, text fields, controllers, etc.
File Template Library
Templates for storyboards, classes, etc.
Code Snippet Library
Snippets of code for common tasks etc.
Click on the Object Library (it might already be selected).
Xcode File Edit View
HelloWorld iPhone 5.0 Simulator
Run Stop Scheme Breakpoints
Build Succeeded | 1/31/12 at 19:22 PM
Editor View Organizer
MainStoryboard.storyboard
MainStoryboard.sto... No Selection
Automatic HelloWorldViewController.h No Selection
//
// HelloWorldViewController.h
// HelloWorld
//
// Copyright (c)
//
#import <UIKit/UIKit.h>
@interface HelloWorldViewController : UIViewController
@end
Identity and Type
File Name HelloWorldViewController.h
File Type Default - C header
Location Relative to Group
HelloWorldViewController.h
Full Path /Users/raduionescu/iPhone/iOSLab/Apps/HelloWorld/HelloWorldViewController.h
No Localizations
Target Membership
HelloWorld
Text Settings
Text Encoding Default - Unicode (UTF-8)
Objects
Label
Text
Hello World View Controller

Task 3

Task: Add a label to display our greeting message to the user.

10. Select the View on the left. Notice the items in the Utilities area change. Study these items.

Note that the top bar is darker gray when the View is selected.

Connections Inspector
Connections between your View and Controller.

Attributes Inspector
See and set attributes of the selected item.

Size Inspector
Position and size the selected item.

The Object Library can display the objects in a list view (currently selected), or in a icons view. The list view gives detailed information about the objects, while the icons view allows you to quickly select objects to add in your view.

Click on your View to select it.

Some objects
(those appropriate to dragging into your View) should appear in the Object Library.

Task 3

Task: Add a label to display our greeting message to the user.

11. Drag an `UILabel` from the Object Library to your View.

The `UILabel` class implements a read-only text view. In general, you can use this class to draw one or multiple lines of static text, such as those you might use to identify other parts of your user interface. The base `UILabel` class provides control over the appearance of your text, including whether it uses a shadow or draws with a highlight.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Fri 18:49 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints

Build Succeeded | 1/31/12 at 19:22 PM

Editor View Organizer

MainStoryboard.storyboard

Hello World... › View

Automatic › HelloWorldViewController.h › No Selection

```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController

@end
```

View

Mode: Scale To Fill

Tag: 0

Interaction: User Interaction Enabled, Multiple Touch

Alpha: 1

Background: Default

Drawing: Opaque, Hidden, Clears Graphics Context, Clip Subviews, Autoresize Subviews

Stretching: X 0, Y 0, Width 1, Height 1

Scroll down to find a Label object.

Objects

objects and controllers not directly available in Interface Builder.

Label – A variably sized amount of static text.

Round Rect Button – Intercepts touch events and sends an action message to target object when it's tapped.

Segmented Control – Displays multiple segments, each of which functions as a discrete button.

Start dragging the Label.

Drag the Label from the Object Library to your View.

Notice the dashed blue lines that Xcode displays as you drag which help you line things up nicely.

Labels are instances of the `UILabel` class in the iOS SDK.

Task 3

Task: Add a label to display our greeting message to the user.

12. Resize the label to 280 width x 36 height pixels.

13. Open the Attributes Inspector.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Fri 19:42 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Build Succeeded | 1/31/12 at 19:22 PM Editor View Organizer

MainStoryboard.storyboard

Label – Label

Automatic › HelloWorldViewController.h › No Selection

```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController

@end
```

Label

Notice the little “handles” around the label.
These can be used to resize it.
That’s exactly what we’re going to do next.

Label

Text Label
Lines 1
Behavior Enabled
Baseline Align Centers
Line Breaks Truncate Tail
Alignment
Font System 17.0
Minimum Size 10 Autoshrink
Text Color Default
Highlighted Default
Shadow Default
Shadow Offset 0 Horizontal -1 Vertical

View

Mode Left
Tag 0

Objects

objects and controllers not directly available in Interface Builder.

Label – A variably sized amount of static text.

Round Rect Button – Intercepts touch events and sends an action message to a target object when it's tapped.

Segmented Control – Displays multiple segments, each of which functions as a discrete button.

Click on the Attributes Inspector. You should see attributes of the Label you just created.

Grab the lower right "handle" on the label and resize it. Use the dashed blue guidelines to pick a good size.

This little indicator will show you the exact size you're resizing to.

Task 3

Task: Add a label to display our greeting message to the user.

14. Change the alignment of the text in our greeting label to be centered.

15. Set the font size to 24 (Helvetica).

Change the alignment of the text in our greetings label by clicking on this button in the inspector.

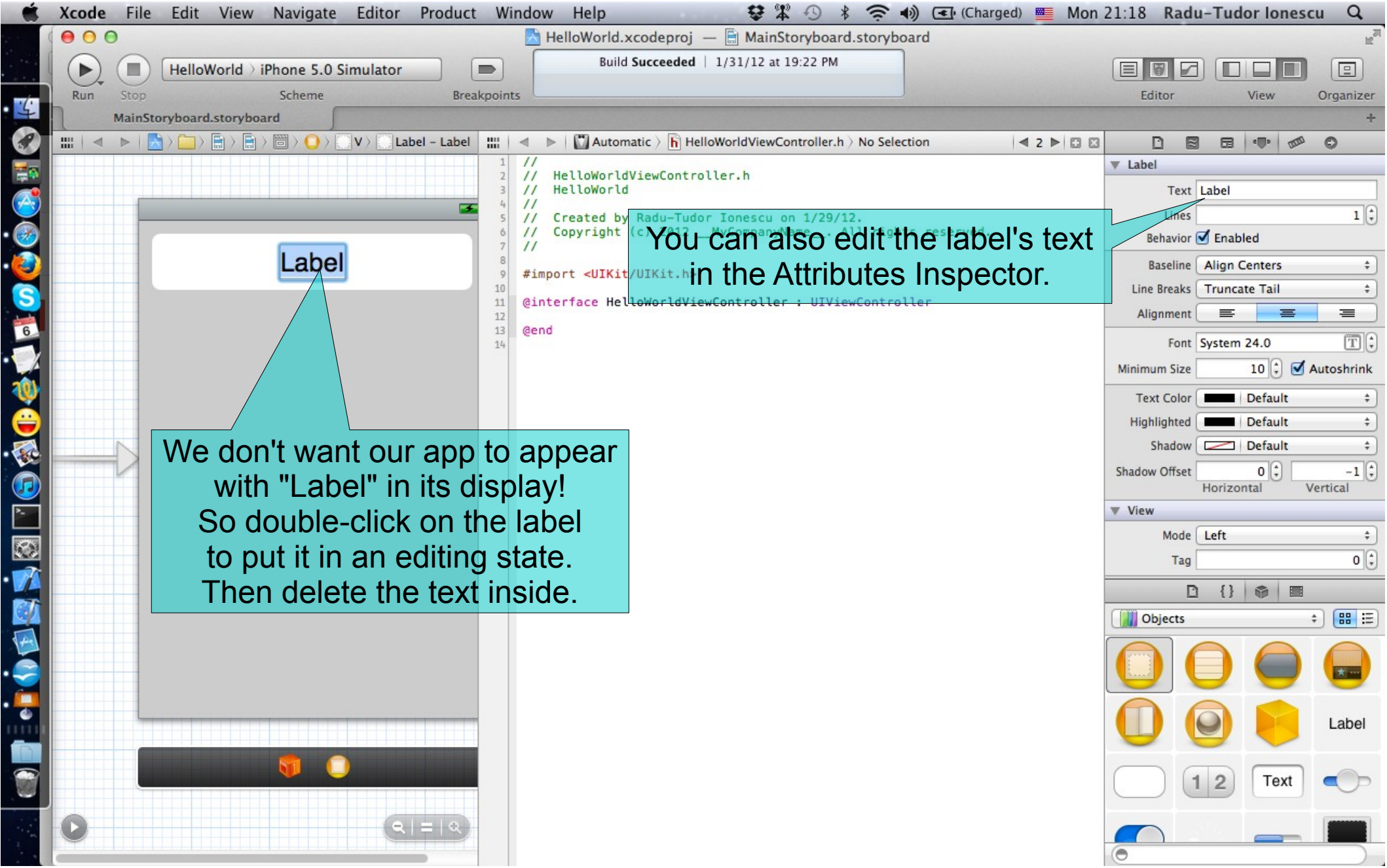
Note that changes in the inspector are reflected immediately in the View.

Let's also make the font bigger. Click this tiny up arrow to increase the font size. Helvetica 24 is nice.

Task 3

Task: Add a label to display our greeting message to the user.

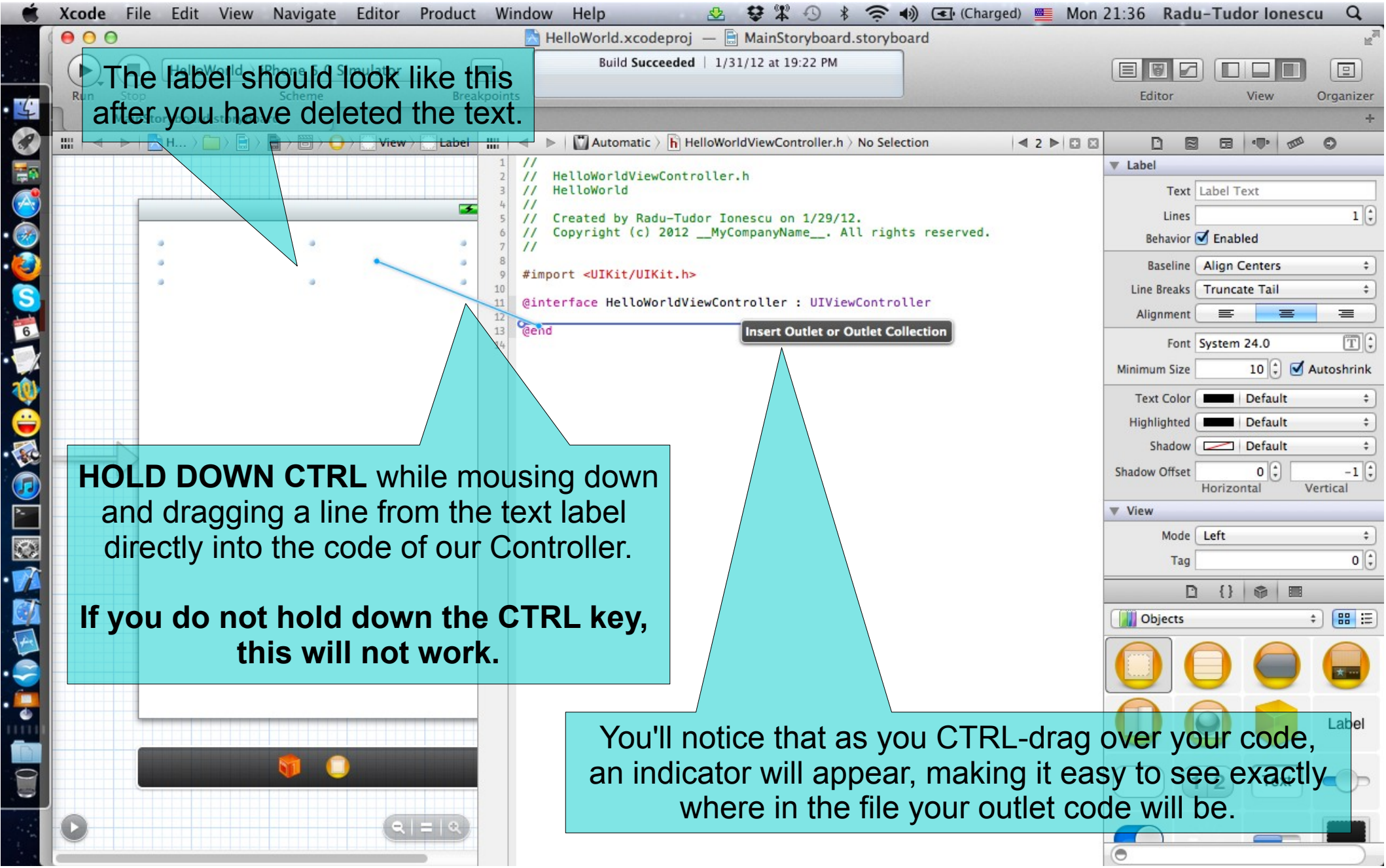
16. Edit the label's text and leave it blank.


Xcode File Edit View Navigate Editor Product Window Help
Mon 21:18 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Build Succeeded 1/31/12 at 19:22 PM
Run Stop Scheme Breakpoints Editor View Organizer
MainStoryboard.storyboard
Label – Label
Automatic HelloWorldViewController.h No Selection
//
// HelloWorldViewController.h
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 MyCompanyName. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface HelloWorldViewController : UIViewController
@end
Label
Text Label
Lines 1
Behavior Enabled
Baseline Align Centers
Line Breaks Truncate Tail
Alignment
Font System 24.0
Minimum Size 10 Autoshrink
Text Color Default
Highlighted Default
Shadow Default
Shadow Offset 0 -1
Horizontal Vertical
View
Mode Left
Tag 0
Objects
Label
Text
You can also edit the label's text in the Attributes Inspector.
We don't want our app to appear with "Label" in its display! So double-click on the label to put it in an editing state. Then delete the text inside.

Task 3

Task: Add a label to display our greeting message to the user.

17. Declare a property in the Controller for the newly added label. Our Controller needs to be able to talk to its View. For example, in this case, we need to be able to update the greeting label as the “Say Hello” button (that we are going to add it in a few slides) is pressed. We can make this connection between Controller and View directly with the mouse.



The label should look like this after you have deleted the text.

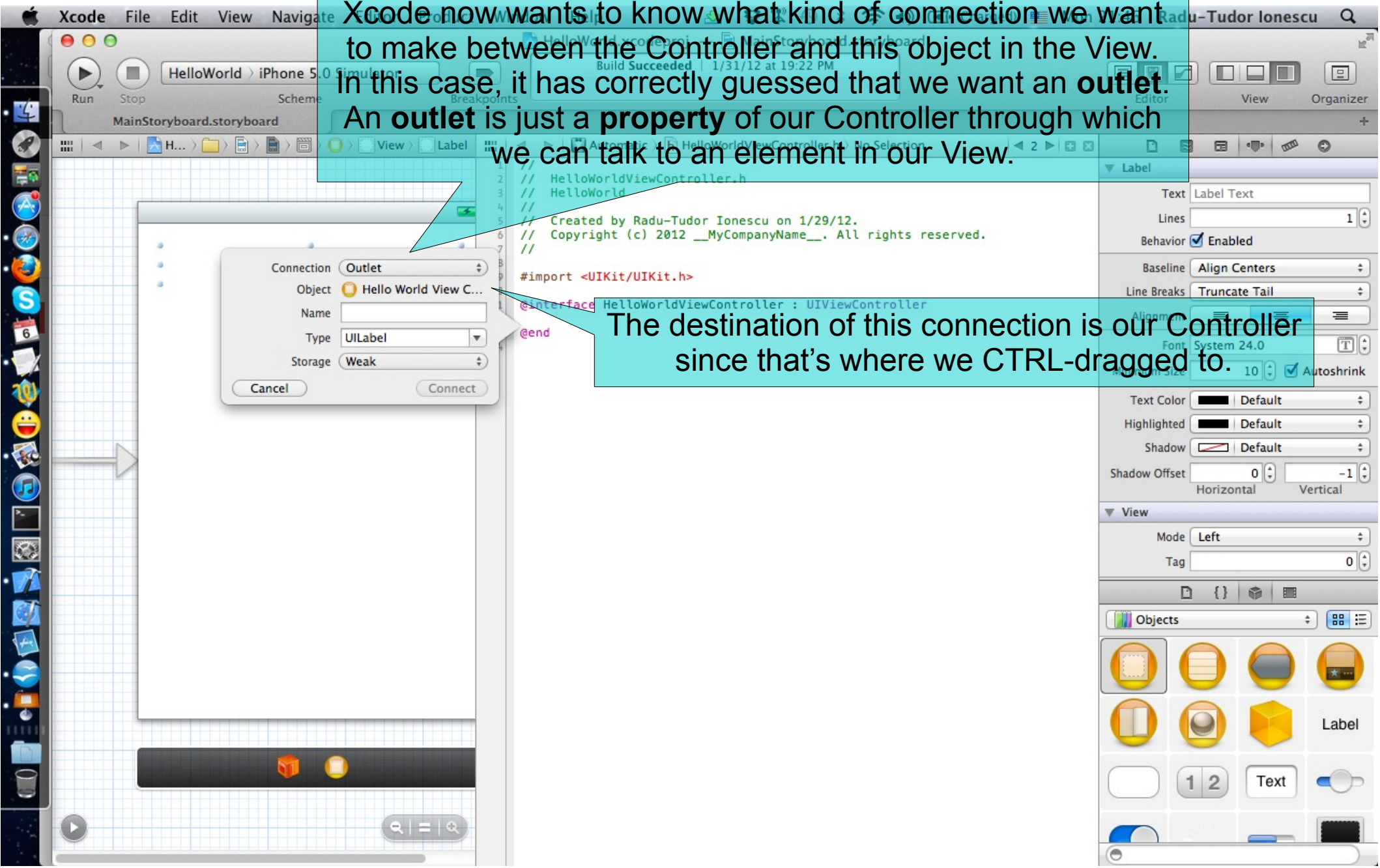
HOLD DOWN CTRL while mousing down and dragging a line from the text label directly into the code of our Controller.

If you do not hold down the CTRL key, this will not work.

You'll notice that as you CTRL-drag over your code, an indicator will appear, making it easy to see exactly where in the file your outlet code will be.

Xcode now wants to know what kind of connection we want to make between the Controller and this object in the View. In this case, it has correctly guessed that we want an **outlet**. An **outlet** is just a **property** of our Controller through which we can talk to an element in our View.

The destination of this connection is our Controller since that's where we CTRL-dragged to.



Task 3

Task: Add a label to display our greeting message to the user.

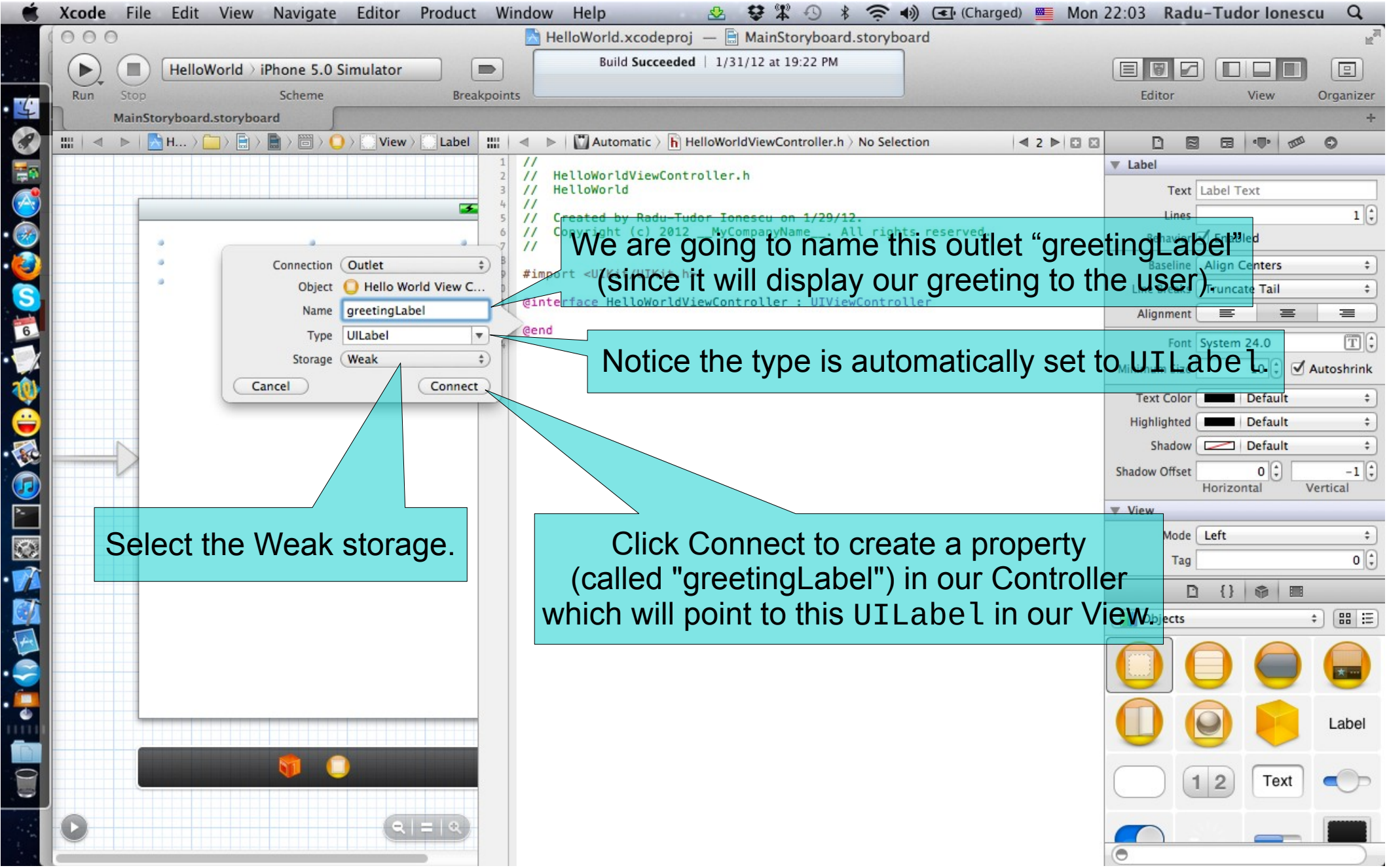
18. Name the property “greetingLabel”.

19. Declare it as a **weak** pointer. An **outlet** is a pointer to an object (a `UILabel` in this case).

A **strong** pointer means the `UILabel` will stick around until we are done using the `UILabel`.

A **weak** pointer means the `UILabel` will only stick around as long as somebody else has a strong pointer to it.

As soon as no one else has a strong pointer to an object that we have a weak pointer to, that object will go away and our pointer to it will be cleared and we won't be able to talk to it (because it will be gone). Since this window already has a strong pointer to this `UILabel`, weak is a good choice here.


Xcode File Edit View Navigate Editor Product Window Help
Mon 22:03 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Build Succeeded | 1/31/12 at 19:22 PM
HelloWorld iPhone 5.0 Simulator
Run Stop Scheme Breakpoints Editor View Organizer
MainStoryboard.storyboard
Connection Outlet
Object Hello World View C...
Name greetingLabel
Type UILabel
Storage Weak
Cancel Connect
We are going to name this outlet "greetingLabel" (since it will display our greeting to the user).
Notice the type is automatically set to UILabel.
Select the Weak storage.
Click Connect to create a property (called "greetingLabel") in our Controller which will point to this UILabel in our View.
HelloWorldViewController.h
Text Label Text
Lines 1
Baseline Align Centers
Truncate Tail
Alignment
Font System 24.0
Autoshrink
Text Color Default
Highlighted Default
Shadow Default
Shadow Offset 0 -1
Horizontal Vertical
Mode Left
Tag 0
Label
Text

Whenever our Controller sends messages to the `greetingLabel` `@property`, it will be talking to this `UILabel` instance.

Xcode has added a `@property` to our MVC Controller which is a pointer to a UILabel object. It has also hooked this `@property` up to the text label we dragged out into our MVC View.

```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *greetingLabel;
@end
```

"greetingLabel" is the name of this `@property`

This property is a weak pointer.

`nonatomic` means "not thread-safe" (more on this later)

`UILabel *` is the type of this `@property` (that means "pointer to a `UILabel` object").

`IBOutlet` is just a word Xcode throws in here so that it can remember that this is an outlet `@property`. It doesn't actually mean anything to the compiler.

Task 3

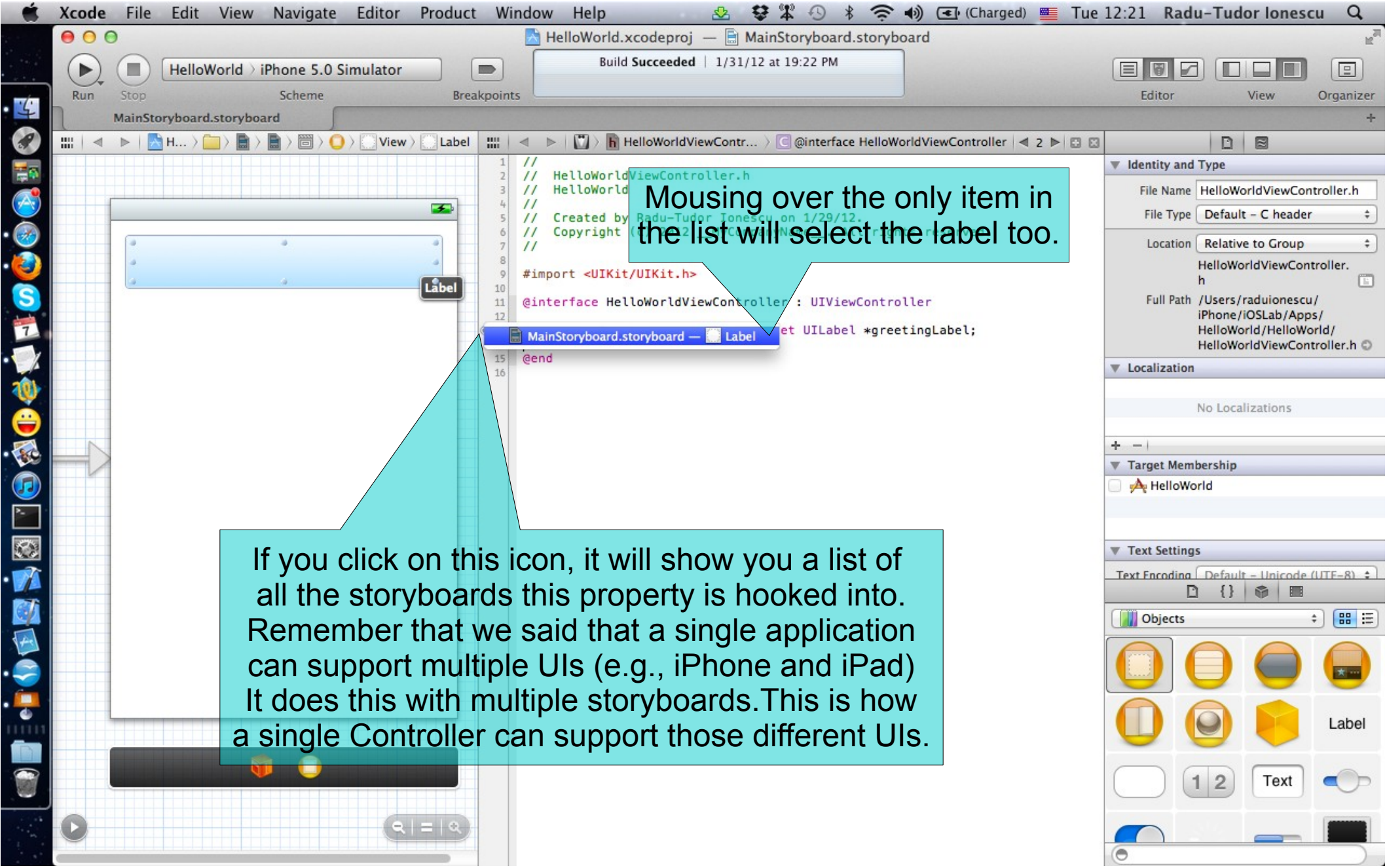
Task: Add a label to display our greeting message to the user.

20. Highlight the label to check if the connection is done right.

Highlighted label.

Mouse over (i.e. hover the mouse over, do not click on) this little icon to see where this `@property` is connected. Notice that the label highlights.

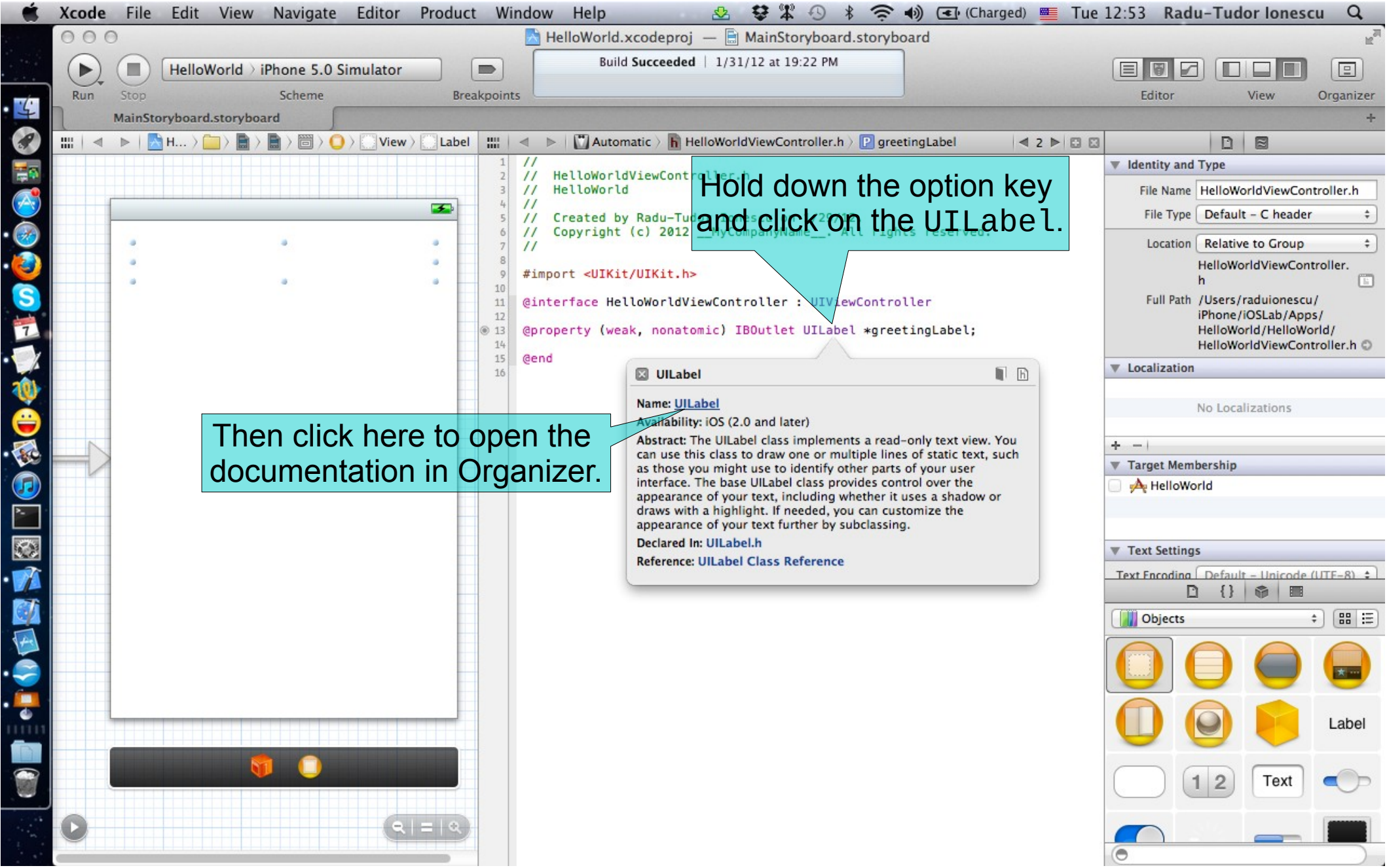
```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *greetingLabel;

@end
```


Xcode File Edit View Navigate Editor Product Window Help
Tue 12:21 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Build Succeeded | 1/31/12 at 19:22 PM
Run Stop Scheme Breakpoints Editor View Organizer
HelloWorld iPhone 5.0 Simulator
MainStoryboard.storyboard
Label
HelloWorldViewContr... @interface HelloWorldViewController
//
// HelloWorldViewController.h
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright
//
#import <UIKit/UIKit.h>
@interface HelloWorldViewCon troller : UIViewController
MainStoryboard.storyboard — Label
et UILabel *greetingLabel;
@end
Mousing over the only item in the list will select the label too.
If you click on this icon, it will show you a list of all the storyboards this property is hooked into. Remember that we said that a single application can support multiple UIs (e.g., iPhone and iPad) It does this with multiple storyboards.This is how a single Controller can support those different UIs.
Identity and Type
File Name HelloWorldViewController.h
File Type Default - C header
Location Relative to Group
HelloWorldViewController.h
Full Path /Users/raduionescu/iPhone/iOSLab/Apps/HelloWorld/HelloWorld/HelloWorldViewController.h
Localization
No Localizations
Target Membership
HelloWorld
Text Settings
Text Encoding Default - Unicode (UTF-8)
Objects
Label
Text

Task 3

Task: Add a label to display our greeting message to the user.

20. Read the documentation of the `UILabel` class. Notice the superclass is `UIView` (all UIKit objects inherit from this class).


Xcode File Edit View Navigate Editor Product Window Help
Tue 12:53 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Run Stop Scheme Breakpoints
HelloWorld iPhone 5.0 Simulator
Build Succeeded | 1/31/12 at 19:22 PM
Editor View Organizer
MainStoryboard.storyboard
Automatic HelloWorldViewController.h greetingLabel
Hold down the option key and click on the UILabel.
Then click here to open the documentation in Organizer.
UILabel
Name: UILabel
Availability: iOS (2.0 and later)
Abstract: The UILabel class implements a read-only text view. You can use this class to draw one or multiple lines of static text, such as those you might use to identify other parts of your user interface. The base UILabel class provides control over the appearance of your text, including whether it uses a shadow or draws with a highlight. If needed, you can customize the appearance of your text further by subclassing.
Declared In: UILabel.h
Reference: UILabel Class Reference
Identity and Type
File Name HelloWorldViewController.h
File Type Default - C header
Location Relative to Group
Full Path /Users/raduionescu/iPhone/iOSLab/Apps/HelloWorld/HelloWorld/HelloWorldViewController.h
Localization
No Localizations
Target Membership
HelloWorld
Text Settings
Objects
Label
Text

Task 3

Task: Add a label to display our greeting message to the user.

21. Learn what are the properties of an `UILabel` object. Read the details about the `text` property. We are going to use it later when we will present the greetings on the display.

The superclass of `UILabel` is `UIView`.

Devices Repositories Projects Archives Documentation

iOS 5.0 Library › User Experience › Windows & Views › UILabel Class Reference

UILabel Class Reference

Inherits from	UIView : UIResponder : NSObject
Conforms to	NSCoding NSCoding (UIView) UIAppearance (UIView) UIAppearanceContainer (UIView) NSObject (NSObject)
Framework	/System/Library/Frameworks/UIKit.framework
Availability	Available in iOS 2.0 and later.
Declared in	UILabel.h
Related sample code	iPhoneCoreDataRecipes SimpleFTPSample SimpleNetworkStreams Teslameter URLCache

Overview

The `UILabel` class implements a read-only text view. You can use this class to draw one or multiple lines of static text, such as those you might use to identify other parts of your user interface. The base `UILabel` class provides control over the appearance of your text, including whether it uses a shadow or draws with a highlight. If needed, you can customize the appearance of your text further by subclassing.

The default content mode of the `UILabel` class is `UIViewContentModeRedraw`. This mode causes the view to redraw its contents every time its bounding rectangle changes. You can change this mode by modifying the inherited `contentMode` property of the class.

New label objects are configured to disregard user events by default. If you want to handle events in a custom subclass of `UILabel`, you must explicitly change the value of the `userInteractionEnabled` property to `YES` after initializing the object.

Tasks

This is the `text` property. Click here to see details about this property.

Accessing the Text Attributes

- `text` *property*
- `font` *property*
- `textColor` *property*
- `textAlignment` *property*
- `lineBreakMode` *property*
- `enabled` *property*

Scroll down to see all the properties.

Sizing the Label's Text

Devices Repositories Projects Archives Documentation

iOS 5.0 Library › User Experience › Windows & Views › UILabel Class Reference › Properties › text

See Also
`@property shadowColor`

Declared In
`UILabel.h`

A brief description of the property.

text

The text displayed by the label.

```
@property(nonatomic, copy) NSString *text
```

How is this property declared in the `UILabel` interface.

Discussion
This string is `nil` by default.

Availability
Available in iOS 2.0 and later.

Since when is this property available.

Related Sample Code
BatteryStatus
iPhoneCoreDataRecipes
LocateMe
URLCache
XMLPerformance

It also gives you related sample code.

Declared In
`UILabel.h`

These are Xcode projects that you can download and look into.

textAlignment

The technique to use for aligning the text.

```
@property(nonatomic) UITextAlignment textAlignment
```

Discussion
This property applies to the entire text string. The default value of this property is `UITextAlignmentLeft`.

Availability
Available in iOS 2.0 and later.

Related Sample Code
BatteryStatus
BubbleLevel
iPhoneCoreDataRecipes
QuickContacts
TableViewSuite

Declared In
`UILabel.h`

Scroll up to top to have a look at the entire organization of this documentation page.

textColor

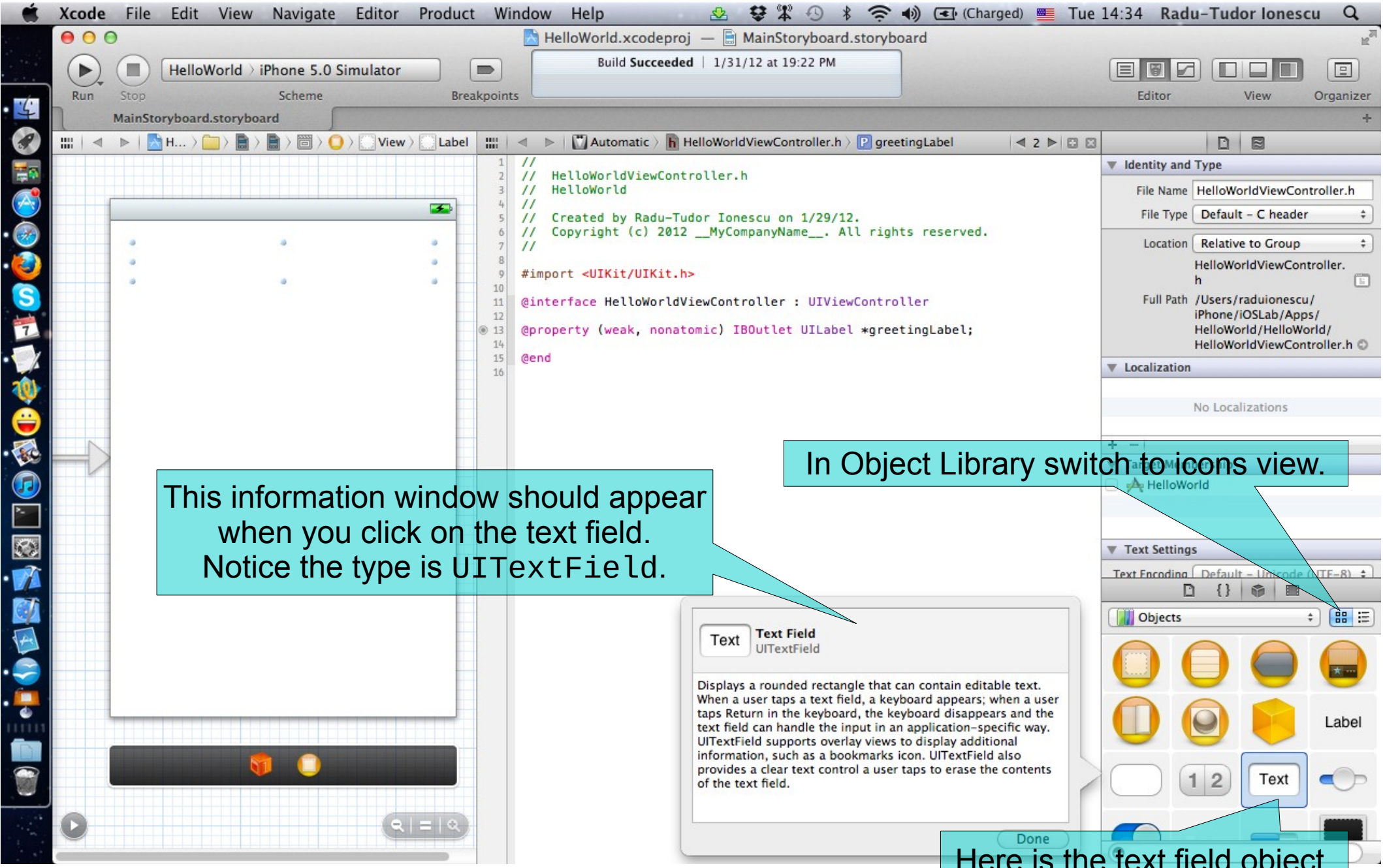
The color of the text.

```
@property(nonatomic, retain) UIColor *textColor
```

Task 4

Task: Add a text field for the user's name. We are going to let the user enter his name in this text field.

1. Look for a text field in the Object Library.

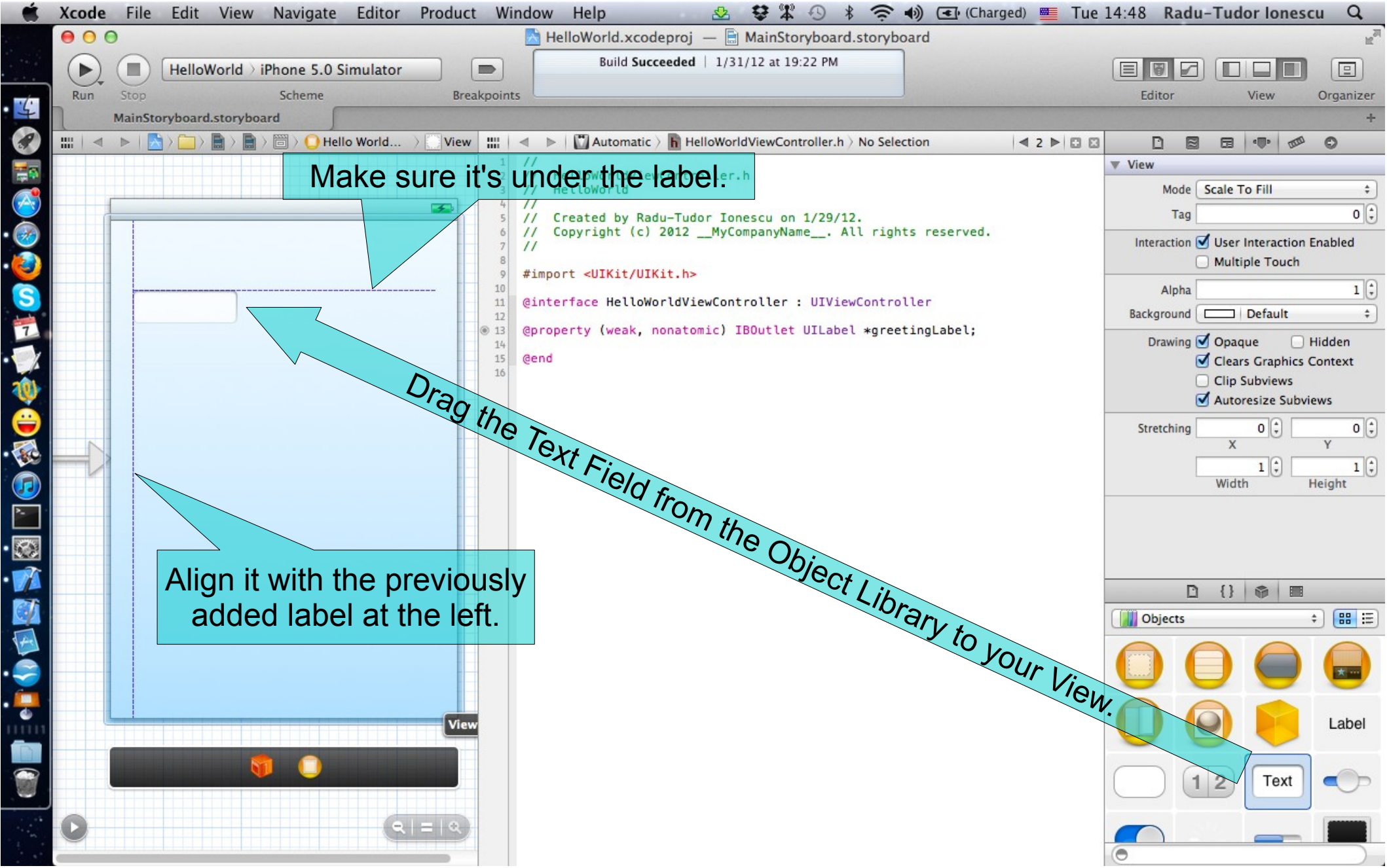

Xcode File Edit View Navigate Editor Product Window Help
Tue 14:34 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Run Stop Scheme Breakpoints
HelloWorld iPhone 5.0 Simulator
Build Succeeded 1/31/12 at 19:22 PM
Editor View Organizer
MainStoryboard.storyboard
View Label
Automatic HelloWorldViewController.h greetingLabel
//
// HelloWorldViewController.h
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface HelloWorldViewController : UIViewController
@property (weak, nonatomic) IBOutlet UILabel *greetingLabel;
@end
Identity and Type
File Name HelloWorldViewController.h
File Type Default - C header
Location Relative to Group
HelloWorldViewController.h
Full Path /Users/raduionescu/iPhone/iOSLab/Apps/HelloWorld/HelloWorld/HelloWorldViewController.h
Localization
No Localizations
Text Settings
Objects
Label
Text
Text Field
UITextField
Displays a rounded rectangle that can contain editable text. When a user taps a text field, a keyboard appears; when a user taps Return in the keyboard, the keyboard disappears and the text field can handle the input in an application-specific way. UITextField supports overlay views to display additional information, such as a bookmarks icon. UITextField also provides a clear text control a user taps to erase the contents of the text field.
Done
This information window should appear when you click on the text field. Notice the type is UITextField.
In Object Library switch to icons view.
Here is the text field object. Click on it.

Task 4

Task: Add a text field for the user's name. We are going to let the user enter his name in this text field.

2. Drag an `UITextField` from the Object Library to your View.

A `UITextField` object is a control that displays editable text and sends an action message to a target object when the user presses the return button. You typically use this class to gather small amounts of text from the user and perform some immediate action, such as a search operation, based on that text.


Xcode File Edit View Navigate Editor Product Window Help
Tue 14:48 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Run Stop Scheme Breakpoints
Build Succeeded 1/31/12 at 19:22 PM
Editor View Organizer
MainStoryboard.storyboard
Hello World... View
Automatic HelloWorldViewController.h No Selection
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface HelloWorldViewController : UIViewController
@property (weak, nonatomic) IBOutlet UILabel *greetingLabel;
@end
View
Mode Scale To Fill
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching 0 X 0 Y
1 Width 1 Height
Objects
Label
Text
Make sure it's under the label.
Drag the Text Field from the Object Library to your View.
Align it with the previously added label at the left.

Task 4

Task: Add a text field for the user's name. We are going to let the user enter his name in this text field.

3. Resize the text field width to 280 pixels.

4. Select the Attributes Inspector in Utilities area.

Make sure Attributes Inspector is selected.

Grab this right handle and adjust the width to 280 pixels.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Tue 18:19 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop Scheme: HelloWorld › iPhone 5.0 Simulator Breakpoints

Build Succeeded | 1/31/12 at 19:22 PM

Editor View Organizer

MainStoryboard.storyboard

View › Text Field

W: 280.0
H: 31.0

Automatic › HelloWorldViewController.h › No Selection

```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2011 MyCompanyName. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *greetingLabel;

@end
```

Text Field

Text: Text
Placeholder: Placeholder Text
Background: Background Image
Disabled: Disabled Background Image
Alignment
Border Style
Clear Button: Never appears
Clear when editing begins
Text Color: Default
Font: System 14.0
Min Font Size: 17
Adjust to Fit
Capitalization: None
Correction: Default
Keyboard: Default
Appearance: Default

Objects

Label

Text

Task 4

Task: Add a text field for the user's name. We are going to let the user enter his name in this text field.

5. Set the placeholder text to “type in your name” to let the user know we would like to know his name.
6. Set the text alignment to center.
7. Set the text field correction to “No” so that it will not try to correct the user's input.
8. Change the keyboard return key type to “Done”.

Notice the text field has a text property. We are going to use this property to get the user's input (that is his name).

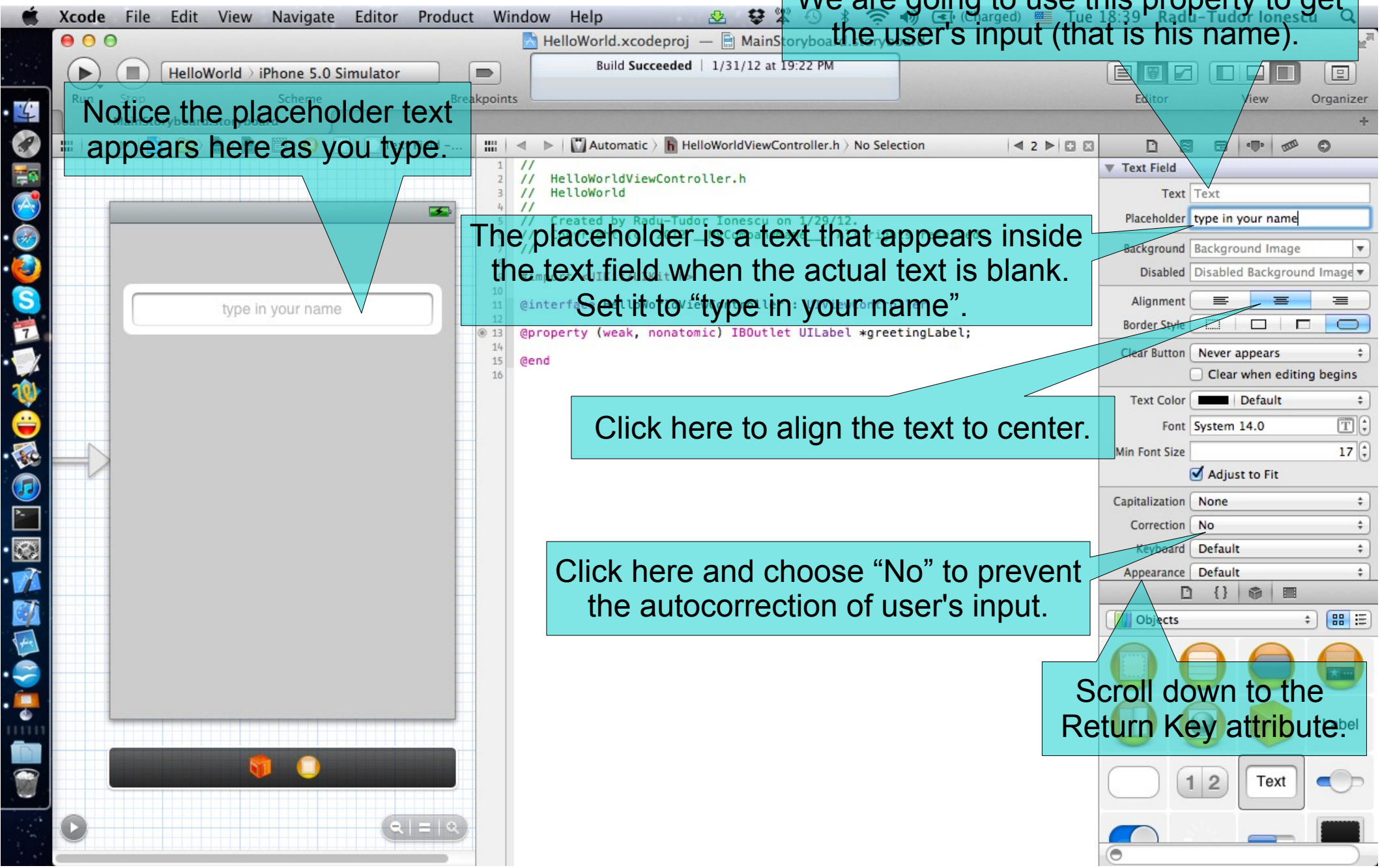
Notice the placeholder text appears here as you type.

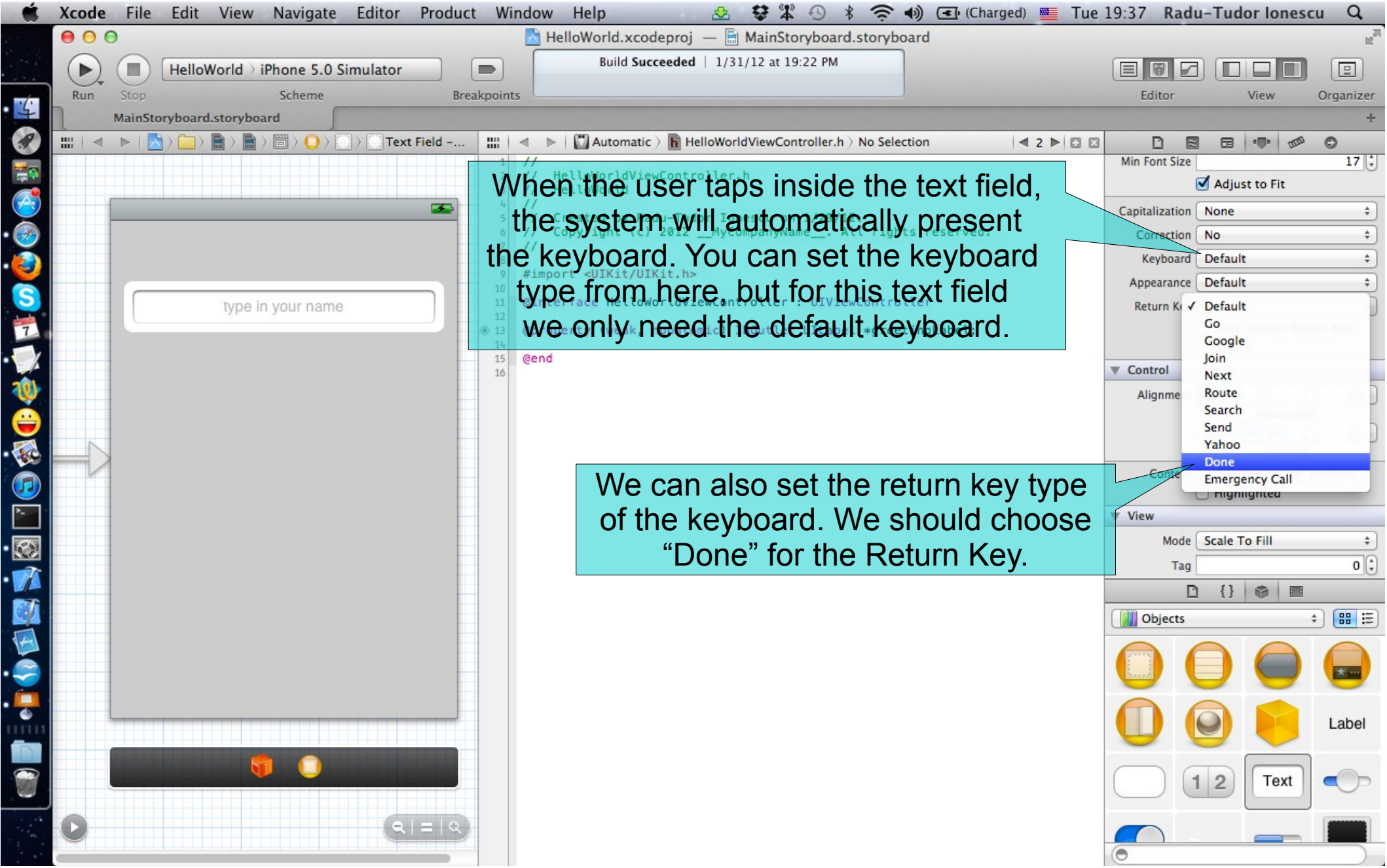
The placeholder is a text that appears inside the text field when the actual text is blank. Set it to “type in your name”.

Click here to align the text to center.

Click here and choose “No” to prevent the autocorrection of user's input.

Scroll down to the Return Key attribute.

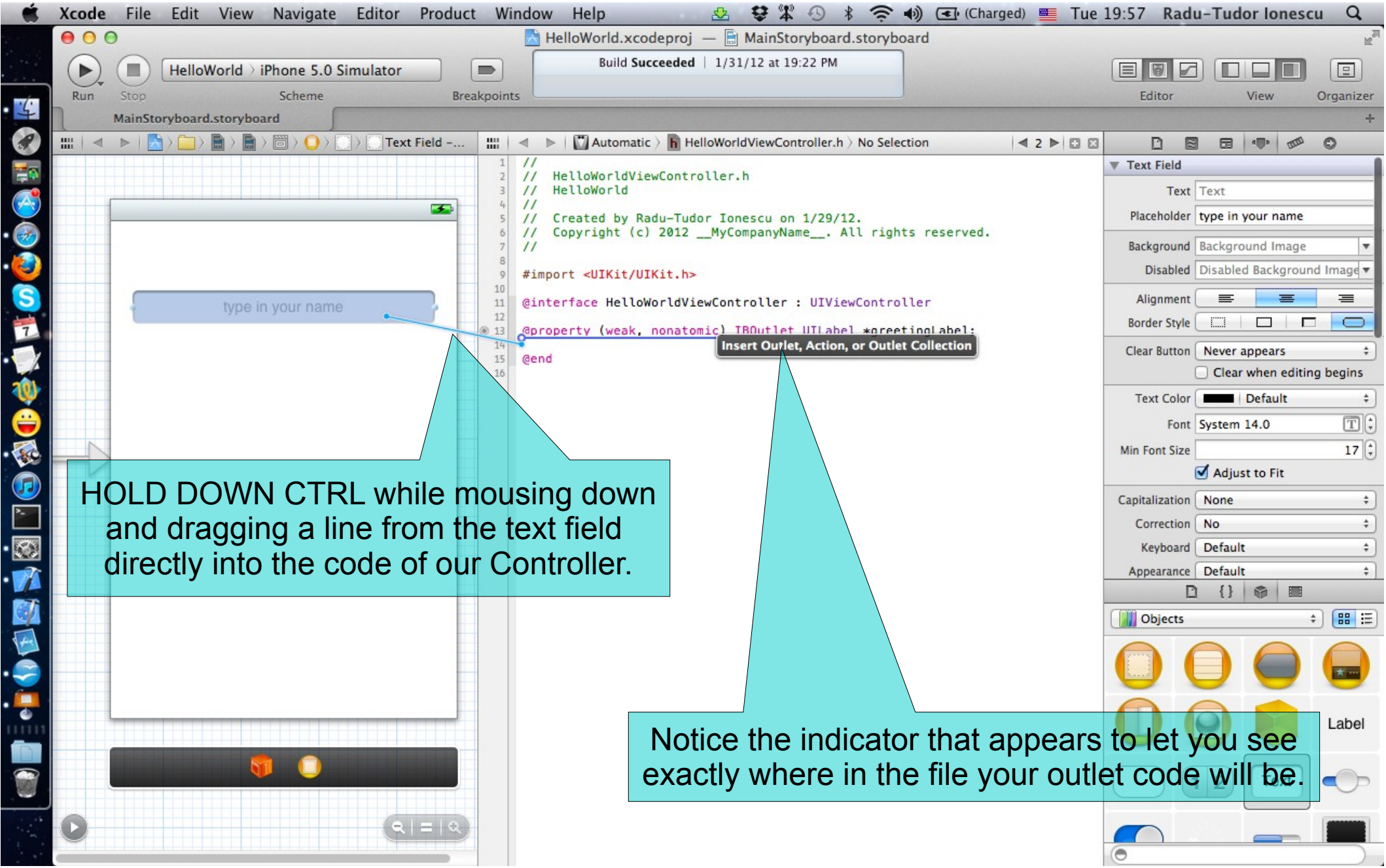



When the user taps inside the text field, the system will automatically present the keyboard. You can set the keyboard type from here, but for this text field we only need the default keyboard.
We can also set the return key type of the keyboard. We should choose “Done” for the Return Key.
Xcode File Edit View Navigate Editor Product Window Help
Tue 19:37 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Build Succeeded | 1/31/12 at 19:22 PM
HelloWorld iPhone 5.0 Simulator
Run Stop Scheme Breakpoints Editor View Organizer
MainStoryboard.storyboard
Text Field –...
Automatic HelloWorldViewController.h No Selection
type in your name
Min Font Size 17
Adjust to Fit
Capitalization None
Correction No
Keyboard Default
Appearance Default
Return Key
Default
Go
Google
Join
Next
Route
Search
Send
Yahoo
Done
Emergency Call
Control
View
Mode Scale To Fill
Tag 0
Objects
Label
Text

Task 4

Task: Add a text field for the user's name. We are going to let the user enter his name in this text field.

9. Declare a property in the Controller for the added text field. We need to be able to get the user's name from the text field and build the greetings message with his name (e.g. “Hello Steve!”). Make this connection between Controller and View directly with the mouse using CTRL-dragging.

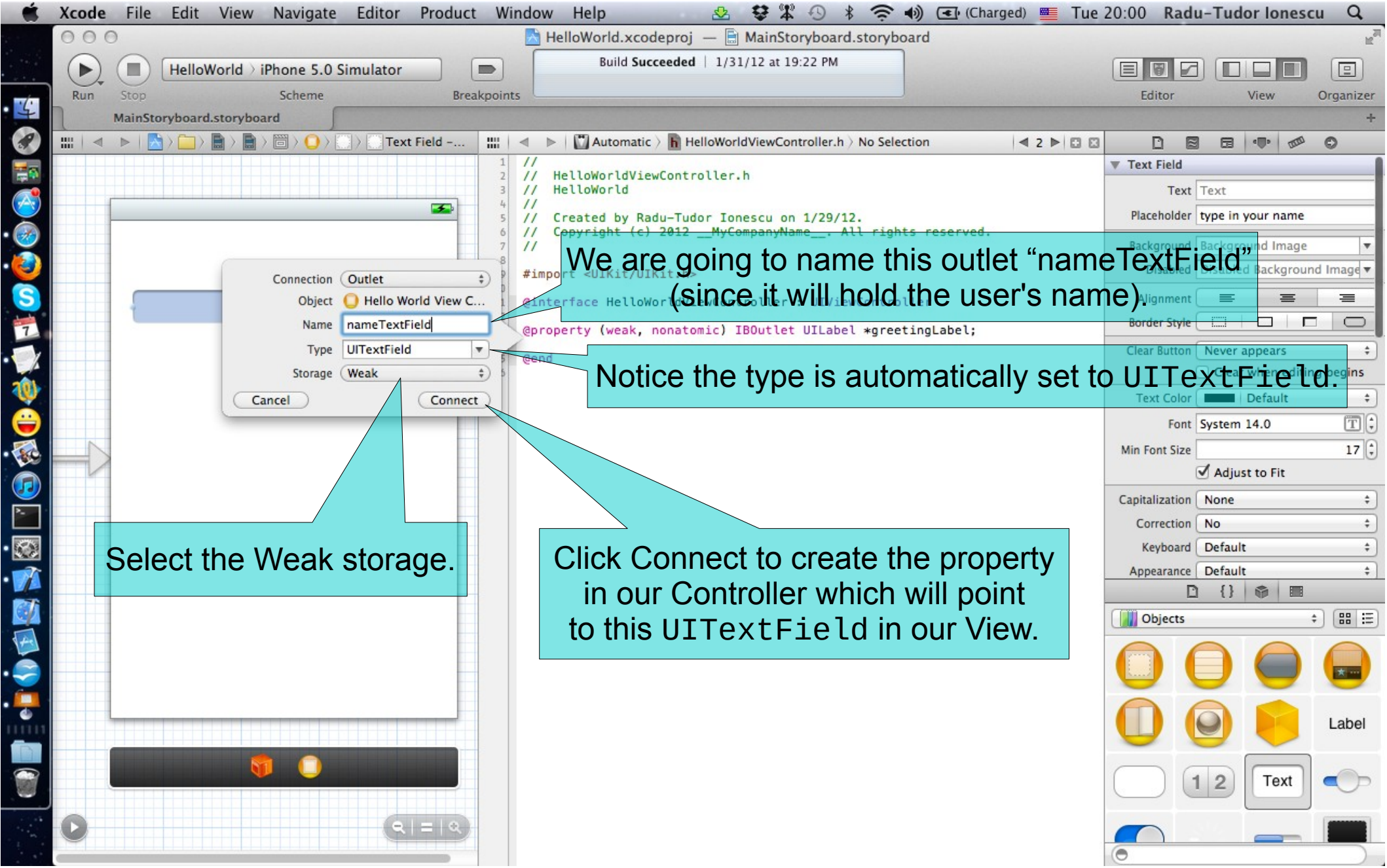

Xcode File Edit View Navigate Editor Product Window Help
Tue 19:57 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Run Stop Scheme Breakpoints
HelloWorld iPhone 5.0 Simulator
Build Succeeded 1/31/12 at 19:22 PM
Editor View Organizer
MainStoryboard.storyboard
Text Field -...
Automatic HelloWorldViewController.h No Selection
type in your name
//
// HelloWorldViewController.h
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface HelloWorldViewController : UIViewController
@property (weak, nonatomic) IBOutlet UILabel *greetingLabel;
@end
Insert Outlet, Action, or Outlet Collection
Text Field
Text
Placeholder type in your name
Background Background Image
Disabled Disabled Background Image
Alignment
Border Style
Clear Button Never appears
Clear when editing begins
Text Color Default
Font System 14.0
Min Font Size 17
Adjust to Fit
Capitalization None
Correction No
Keyboard Default
Appearance Default
Objects
Label
HOLD DOWN CTRL while mousing down and dragging a line from the text field directly into the code of our Controller.
Notice the indicator that appears to let you see exactly where in the file your outlet code will be.

Task 4

Task: Add a text field for the user's name. We are going to let the user enter his name in this text field.

10. Name the property “nameTextField”.

11. Declare it as a **weak** pointer.


Xcode File Edit View Navigate Editor Product Window Help
Tue 20:00 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Build Succeeded | 1/31/12 at 19:22 PM
HelloWorld iPhone 5.0 Simulator
Run Stop Scheme Breakpoints Editor View Organizer
MainStoryboard.storyboard
Text Field –...
Automatic HelloWorldViewController.h No Selection
//
// HelloWorldViewController.h
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
@interface HelloWorldViewController : UIViewController
@property (weak, nonatomic) IBOutlet UILabel *greetingLabel;
@end
Connection Outlet
Object Hello World View C...
Name nameTextField
Type UITextField
Storage Weak
Cancel
Connect
We are going to name this outlet "nameTextField" (since it will hold the user's name).
Notice the type is automatically set to UITextField.
Select the Weak storage.
Click Connect to create the property in our Controller which will point to this UITextField in our View.
Text Field
Text
Placeholder type in your name
Background Background Image
Disabled Disabled Background Image
Alignment
Border Style
Clear Button Never appears
Clear when editing begins
Text Color Default
Font System 14.0
Min Font Size 17
Adjust to Fit
Capitalization None
Correction No
Keyboard Default
Appearance Default
Objects
Label
Text

Highlighted text field.

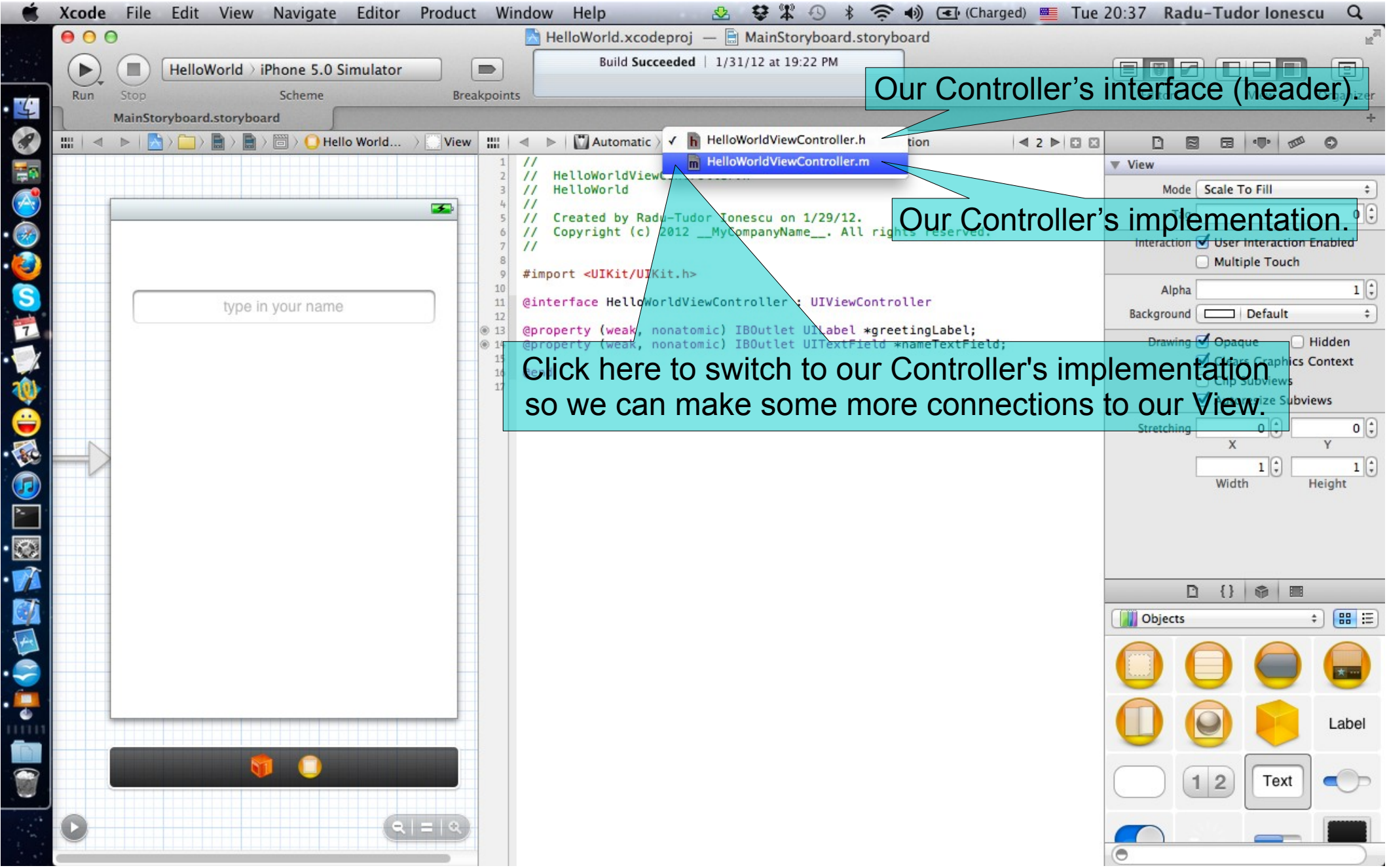
Mouse over this little icon to see where this `@property` is connected. Notice that the text field highlights.

`UITextField *` is the type of this `@property` (that means "pointer to a `UITextField` object").

Task 5

Task: Add a button that will trigger the greeting message.

1. Switch to the Controller's implementation.


Our Controller's interface (header).
Our Controller's implementation.
Click here to switch to our Controller's implementation so we can make some more connections to our View.
HelloWorldViewController.h
HelloWorldViewController.m
type in your name

Task 5

Task: Add a button that will trigger the greeting message.

2. Delete the code that we don't need which was automatically added by Xcode. Make sure **NOT** to remove the `@synthesize` declarations and the implementation of the `viewDidUnload` method.

The @synthesize declarations are very important. We'll cover them in a moment.

Notice that the `@implementation` here does not specify the superclass. Only the `@interface` in the .h (header) file does that.

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel;
@synthesize nameTextField;

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Release any cached data, images, etc that aren't in use.
}

#pragma mark - View lifecycle

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
}

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
}

- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
}

- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];
```

Xcode has added some extra code that we don't need for this application, so select everything from after the `@synthesize nameTextField` until the `viewDidUnload` method and delete it.

Select the code from after the `viewDidUnload` method implementation all the way down to (but not including) the @end. Then hit delete to delete it.

```
- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
}

- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
}

- (void)viewWillDisappear:(BOOL)animated
{
    [super viewWillDisappear:animated];
}

- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interf
{
    // Return YES for supported orientations
    return (interfaceOrientation != UIInterfaceOrientationPortraitUpsideDown)
}

@end
```

Task 5

Task: Add a button that will trigger the greeting message.

3. Study the `@synthesize` declarations and the `viewDidUnload` method implementation.

The `@synthesize` declarations were added by Xcode when we created the `greetingLabel` and `nameTextField` properties. The `@synthesize` declaration is used to generate accessor methods (setter and getter) for a certain property.

When a low-memory condition occurs and the current view controller's views are not needed, the system may opt to remove those views from memory. The `viewDidUnload` method is called after the view controller's view has been released and is your chance to perform any final cleanup. If your view controller stores references to the view or its subviews, you should use this method to release those references and set those references to `nil`.

Note the `@synthesize` that Xcode automatically added to our Controller's implementation when it created the `greetingLabel` `@property` (it did this when we CTRL-dragged to create the `greetingLabel` outlet). This `@synthesize` creates two methods (`greetingLabel` and `setGreetingLabel:`). The method `setGreetingLabel:` is used by iOS to hook the `UILabel` up to the `greetingLabel` `@property` at runtime (i.e. set the value of the pointer). The method `setGreetingLabel:` is also used by us to put the greeting message in this `UILabel`. `@synthesize` also creates an instance variable to store this pointer.

```
@synthesize greetingLabel;
@synthesize nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

@end
```

`void` is the return type of this method. More about methods in a few moments.

The `viewDidUnload` method sets the `greetingLabel` and `nameTextField` references to `nil` by calling the setters generated with `@synthesize`. This code was also automatically added to our implementation when we CTRL-dragged to create the outlets.

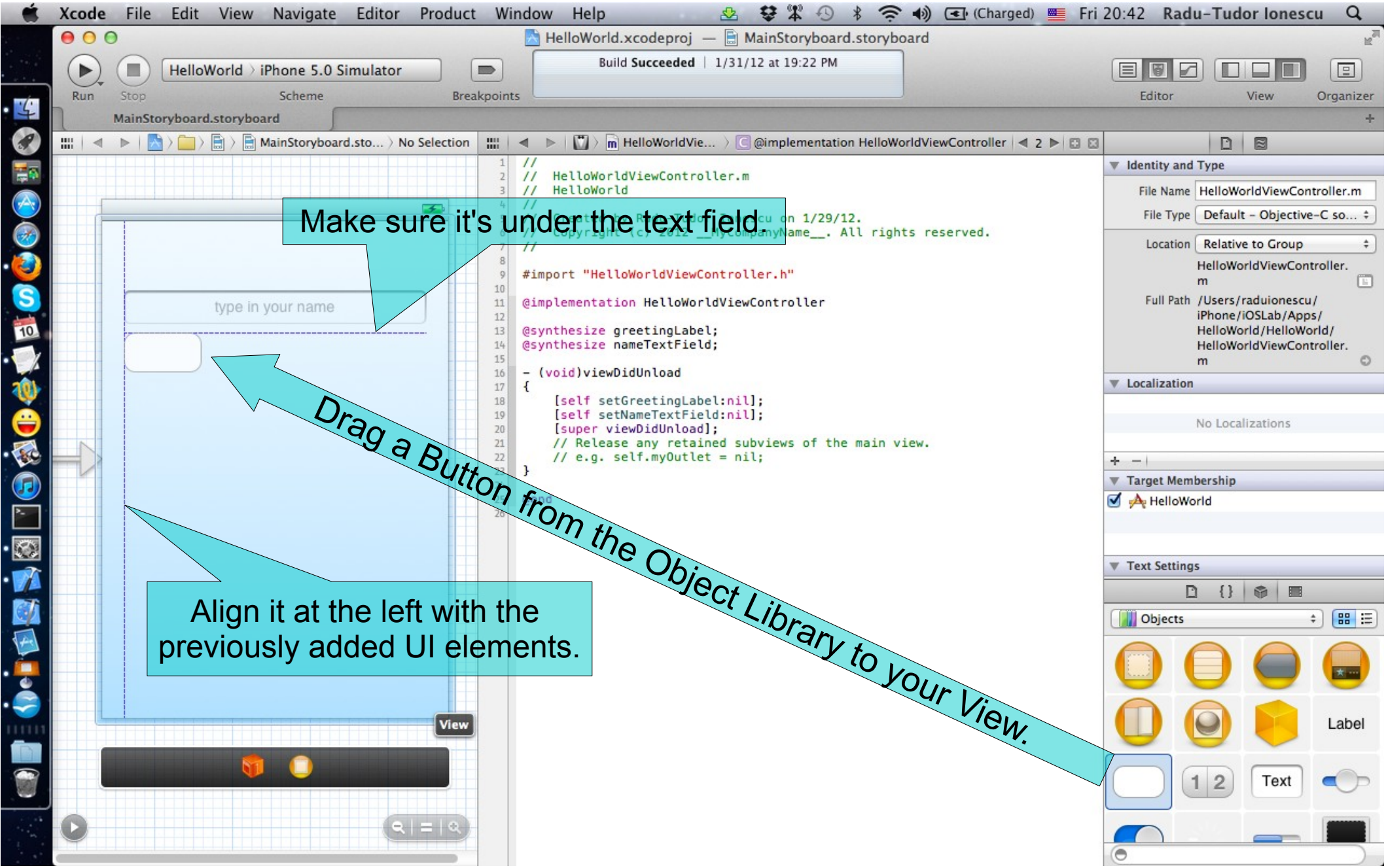
The same happens for this `@synthesize` which creates two methods (`nameTextField` and `setNameTextField:`). As for the `UILabel`, the method `setNameTextField:` is used by iOS to hook the `UITextField` up to the `nameTextField` `@propzerty` at runtime. The method `nameTextField` is used by us to get the user's name from the text field.

Task 5

Task: Add a button that will trigger the greeting message.

4. Drag a Round Rect Button from the Object Library to your View.

An instance of the `UIButton` class implements a button on the touch screen. A button intercepts touch events and sends an action message to a target object when tapped. This class provides methods for setting the title, image, and other appearance properties of a button. By using these accessors, you can specify a different appearance for each button state.

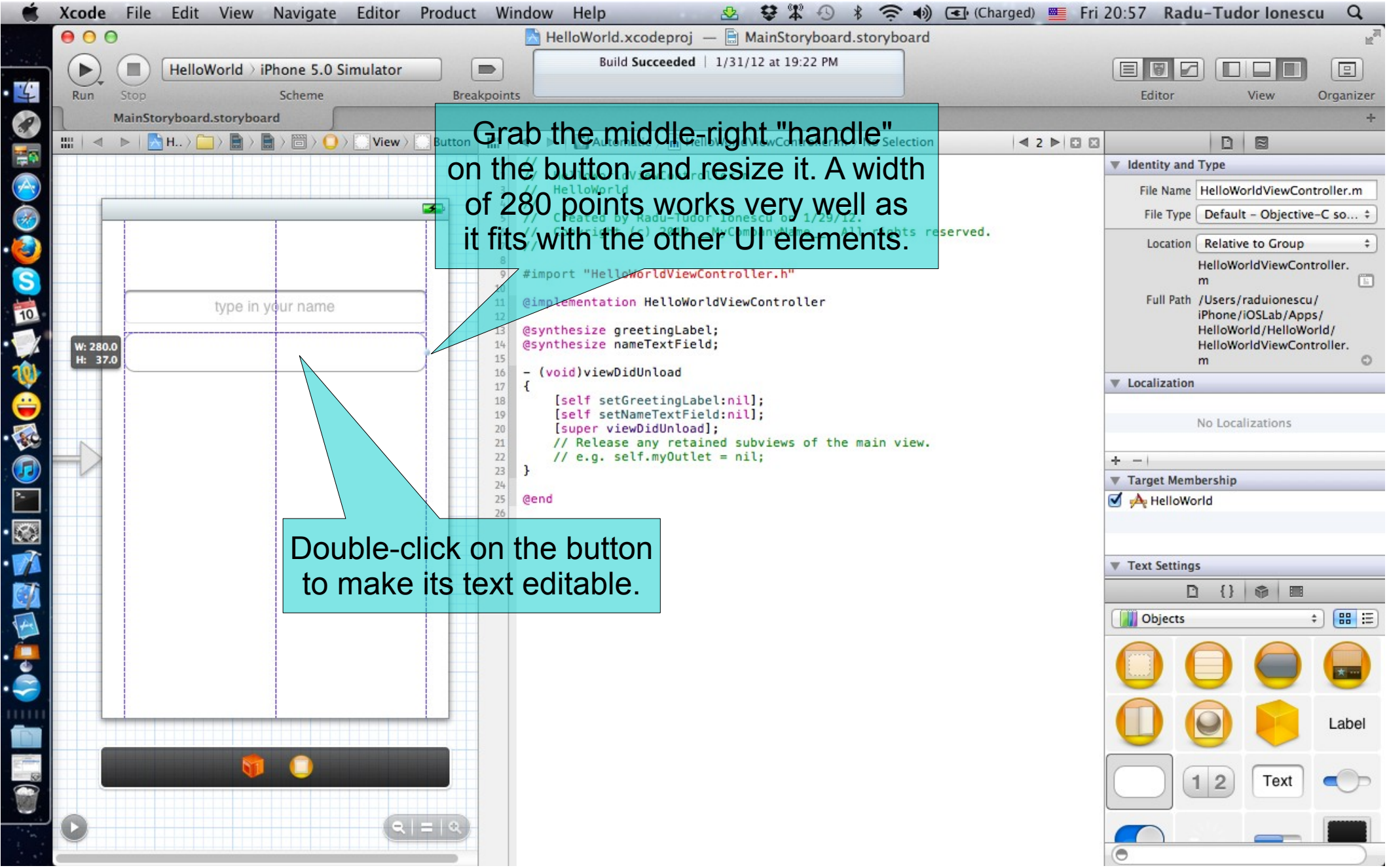

Xcode File Edit View Navigate Editor Product Window Help
Fri 20:42 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Build Succeeded 1/31/12 at 19:22 PM
Run Stop Scheme Breakpoints Editor View Organizer
MainStoryboard.storyboard
MainStoryboard.sto... No Selection
HelloWorldVie... @implementation HelloWorldViewController
//
// HelloWorldViewController.m
// HelloWorld
#import "HelloWorldViewController.h"
@implementation HelloWorldViewController
@synthesize greetingLabel;
@synthesize nameTextField;
- (void)viewDidUnload
{
[self setGreetingLabel:nil];
[self setNameTextField:nil];
[super viewDidUnload];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
}
type in your name
View
Make sure it's under the text field.
Drag a Button from the Object Library to your View.
Align it at the left with the previously added UI elements.
Identity and Type
File Name HelloWorldViewController.m
File Type Default - Objective-C so...
Location Relative to Group
HelloWorldViewController.m
Full Path /Users/raduionescu/iPhone/iOSLab/Apps/HelloWorld/HelloWorld/HelloWorldViewController.m
Localization
No Localizations
Target Membership
HelloWorld
Text Settings
Objects
Label
Text

Task 5

Task: Add a button that will trigger the greeting message.

5. Adjust the button width to 280 pixels.

6. Change the button's title to “Say Hello”.


Xcode File Edit View Navigate Editor Product Window Help
Fri 20:57 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Build Succeeded | 1/31/12 at 19:22 PM
HelloWorld iPhone 5.0 Simulator
Run Stop Scheme Breakpoints Editor View Organizer
MainStoryboard.storyboard
Grab the middle-right "handle" on the button and resize it. A width of 280 points works very well as it fits with the other UI elements.
type in your name
W: 280.0
H: 37.0
Double-click on the button to make its text editable.
#import "HelloWorldViewController.h"
@implementation HelloWorldViewController
@synthesize greetingLabel;
@synthesize nameTextField;
- (void)viewDidUnload
{
[self setGreetingLabel:nil];
[self setNameTextField:nil];
[super viewDidUnload];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
}
@end
Identity and Type
File Name HelloWorldViewController.m
File Type Default - Objective-C so...
Location Relative to Group
HelloWorldViewController.m
Full Path /Users/raduionescu/iPhone/iOSLab/Apps/HelloWorld/HelloWorld/HelloWorldViewController.m
Localization
No Localizations
Target Membership
HelloWorld
Text Settings
Objects
Label
Text

Notice the button is focused when you double-click on it.

Type-in “Say Hello” to set the button's title.

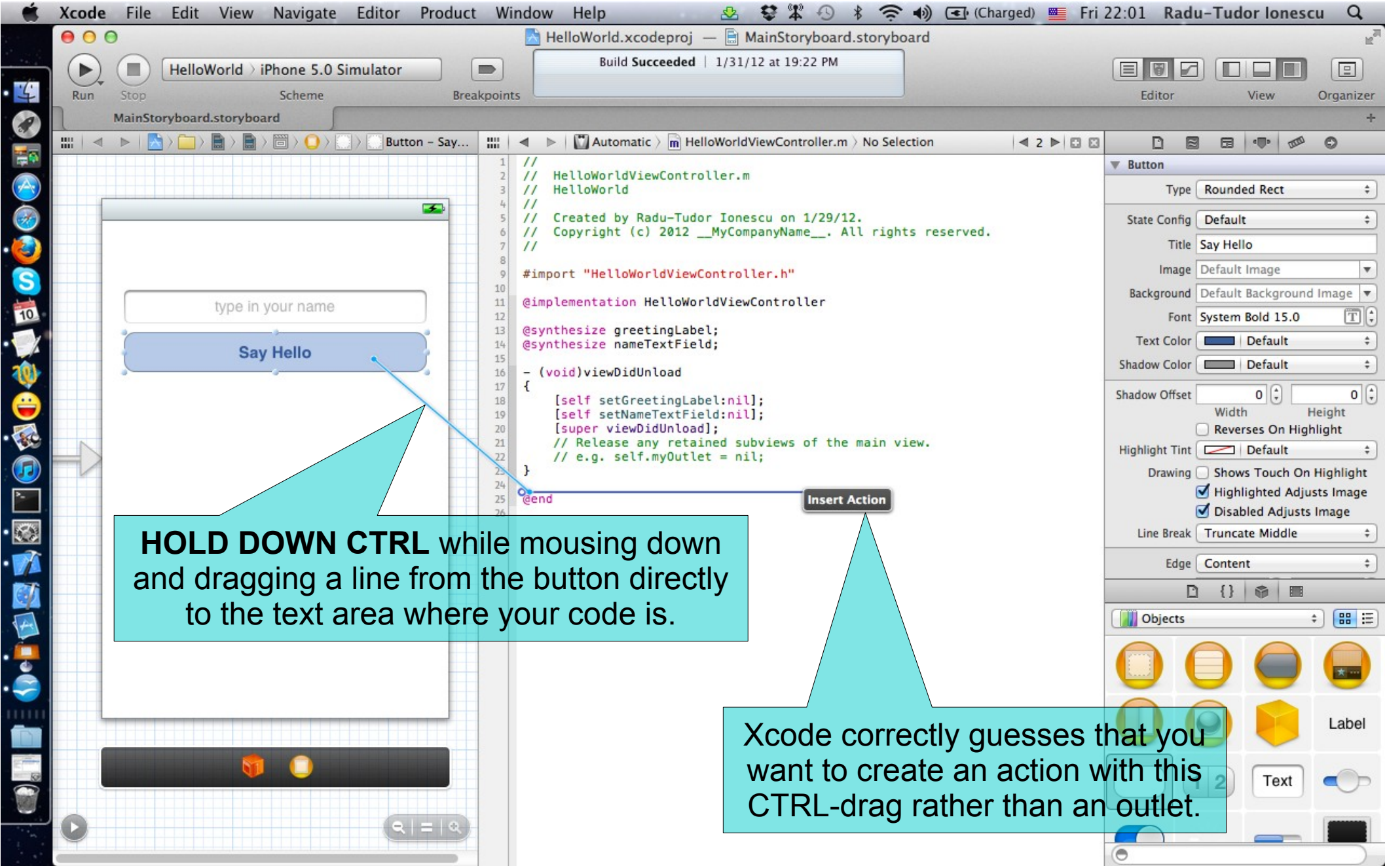
Task 5

Task: Add a button that will trigger the greeting message.

7. Specify the action that our `UIButton` is going to send to our Controller when the user touches it.

Remember that the term **outlet** refers to a `@property` through which we send messages to something in our View from our Controller (for example, `greetingLabel` is an outlet).

We use the term **action** to mean a method that is going to be sent from an object in our View to our Controller when something interesting happens in the user-interface.



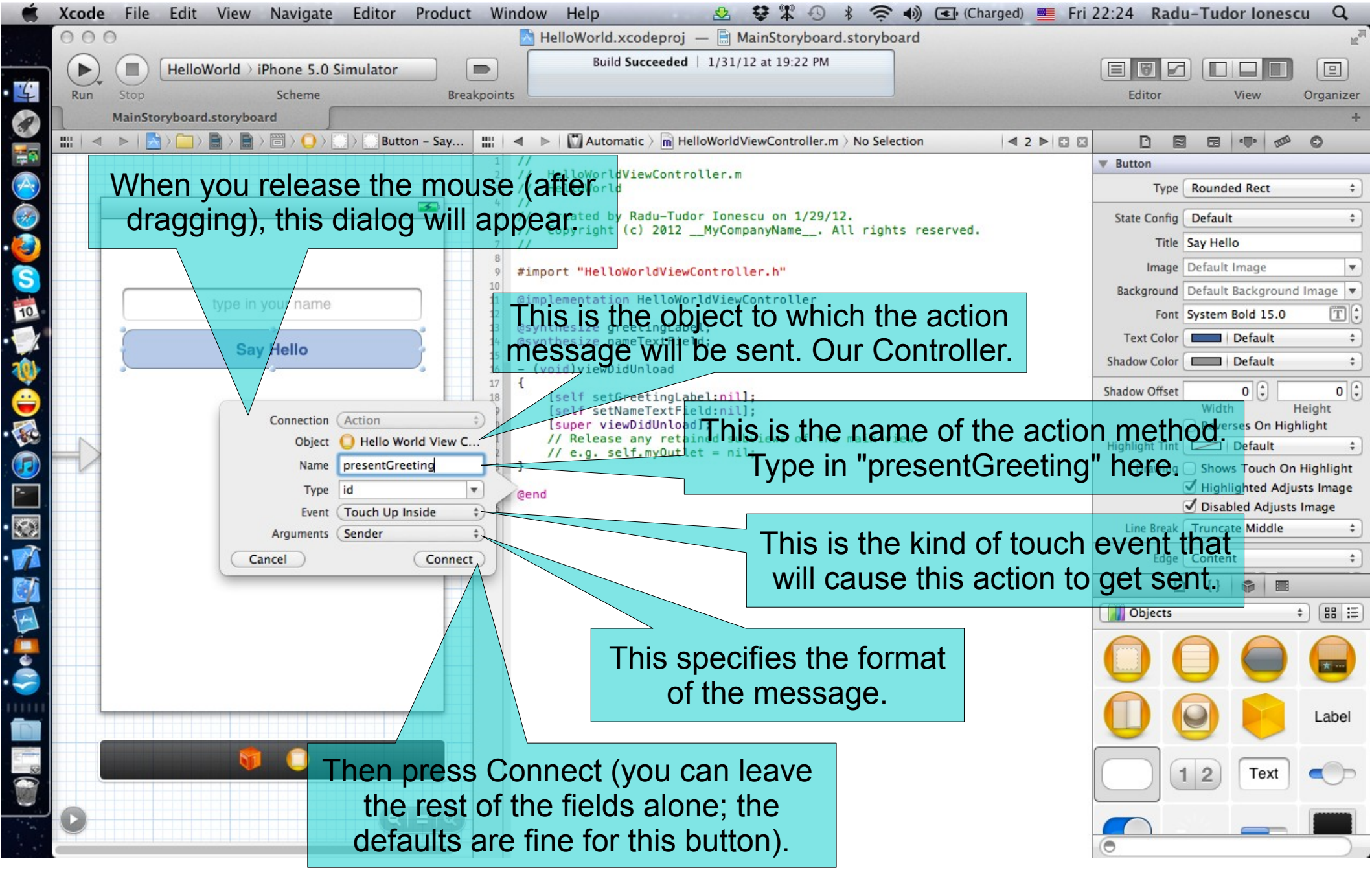
HOLD DOWN CTRL while mousing down and dragging a line from the button directly to the text area where your code is.

Xcode correctly guesses that you want to create an action with this CTRL-drag rather than an outlet.

Task 5

Task: Add a button that will trigger the greeting message.

8. Enter “presentGreeting” as the name of the action message (which makes sense since this button is going to be the button that triggers the presentation of the greetings message).


When you release the mouse (after dragging), this dialog will appear.
This is the object to which the action message will be sent. Our Controller.
This is the name of the action method. Type in "presentGreeting" here.
This is the kind of touch event that will cause this action to get sent.
This specifies the format of the message.
Then press Connect (you can leave the rest of the fields alone; the defaults are fine for this button).
Connection
Action
Object
Hello World View C...
Name
presentGreeting
Type
id
Event
Touch Up Inside
Arguments
Sender
Cancel
Connect
type in your name
Say Hello
Button
Type
Rounded Rect
State Config
Default
Title
Say Hello
Image
Default Image
Background
Default Background Image
Font
System Bold 15.0
Text Color
Default
Shadow Color
Default
Shadow Offset
Width
Height
Reverses On Highlight
Highlight Tint
Shows Touch On Highlight
Highlighted Adjusts Image
Disabled Adjusts Image
Line Break
Truncate Middle
Edge
Content
Objects
Label
Text

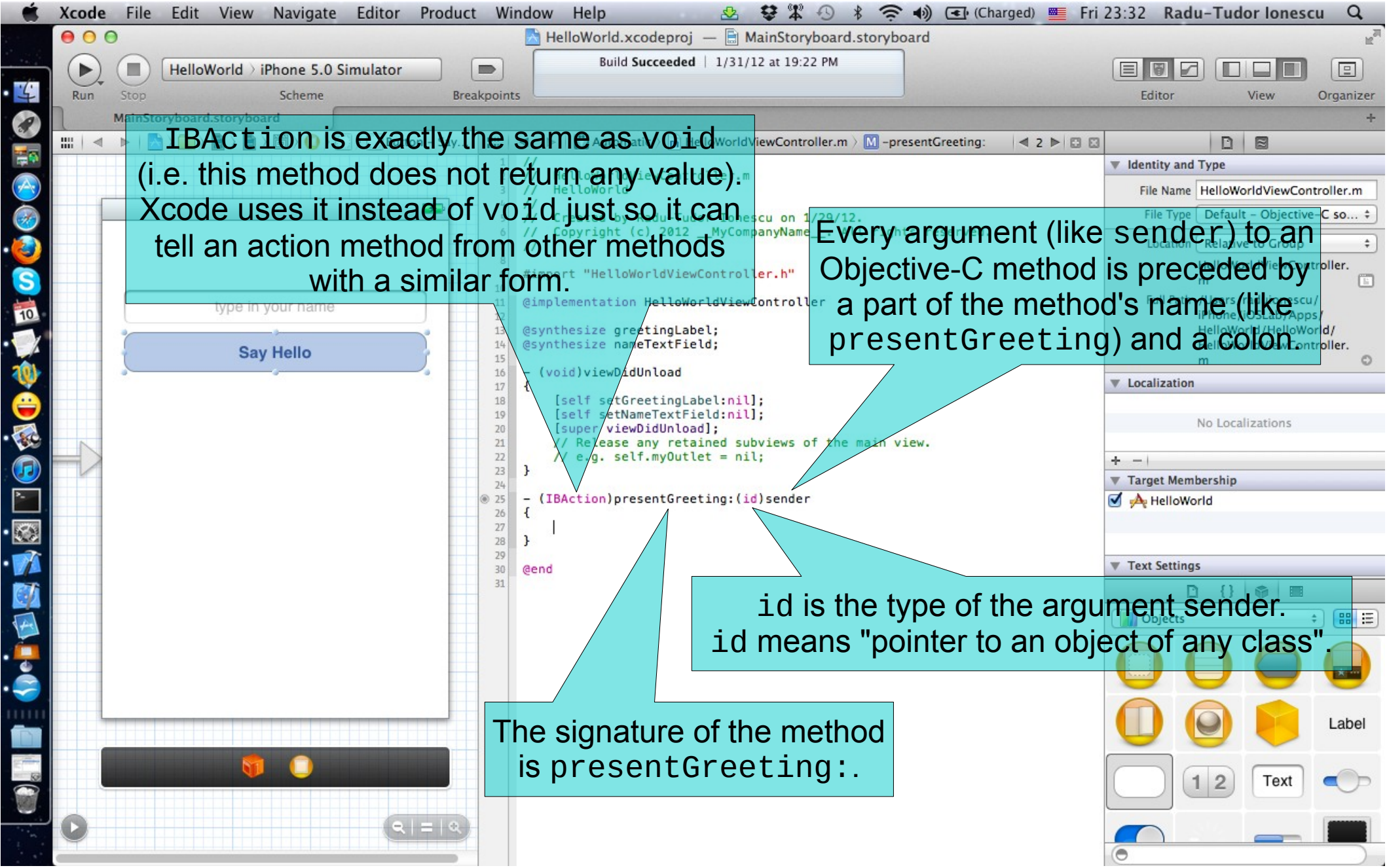
Task 5

Task: Add a button that will trigger the greeting message.

9. Study the `presentGreeting:` method added by CTRL-dragging to the Controller.

Notice the method has a parameter (called `sender`) with `id` type. You might be surprised that this does not read `id *`. But that would make no sense because the type `id` is already a pointer so `id *` would be a pointer to a pointer. The type `id` does not mean "object of any class", it means "pointer to an object of any class". Every time the “Say Hello” button is touched, `presentGreeting:` is going to be sent to our Controller with the UIButton itself as the message’s `sender` argument.

In general, we want to use the type `id` because either we want to allow any class of object to be passed into a method (uncommon) or because the class of the object is "opaque" (it’s like a "cookie").



IBAction is exactly the same as void (i.e. this method does not return any value). Xcode uses it instead of void just so it can tell an action method from other methods with a similar form.

Every argument (like sender) to an Objective-C method is preceded by a part of the method's name (like presentGreeting) and a colon.

id is the type of the argument sender. id means "pointer to an object of any class".

The signature of the method is presentGreeting:.

Task 5

Task: Add a button that will trigger the greeting message.

10. Highlight the button in Interface Builder.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Mon 18:56 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints

Build Succeeded | 1/31/12 at 19:22 PM

Editor View Organizer

MainStoryboard.storyboard

Hello World... › View

Highlighted button.

type in your name

Say Hello

Button – Say Hell

Automatic › HelloWorldViewController.m › -presentGreeting: 2

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel;
@synthesize nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    |
}

@end
```

Similar to an outlet, you can mouse over this little icon and see which object(s) in your View send(s) this message. Notice how the button highlights.

Identity and Type

File Name: HelloWorldViewController.m

File Type: Default – Objective-C so...

Location: Relative to Group

HelloWorldViewController.m

Full Path: /Users/raduionescu/iPhone/iOSLab/Apps/HelloWorld/HelloWorld/HelloWorldViewController.m

Localization

No Localizations

Target Membership

HelloWorld

Text Settings

Objects

Label

Text

Xcode File Edit View Navigate Editor Product Window Help (Charged) Mon 19:03 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Build Succeeded | 1/31/12 at 19:22 PM Editor View Organizer

MainStoryboard.storyboard

Hello World... › View

type in your name

Say Hello

Button – Say Hell

Automatic › HelloWorldViewController.m › -presentGreeting: 2

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel;
@synthesize nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

}

@end
```

MainStoryboard.storyboard — Button – Say Hello

Identity and Type

File Name HelloWorldViewController.m

File Type Default – Objective-C so...

Location Relative to Group

HelloWorldViewController.m

Full Path /Users/raduionescu/iPhone/iOSLab/Apps/HelloWorld/HelloWorld/HelloWorldViewController.m

Localization

No Localizations

Target Membership

HelloWorld

Text Settings

Label

Text

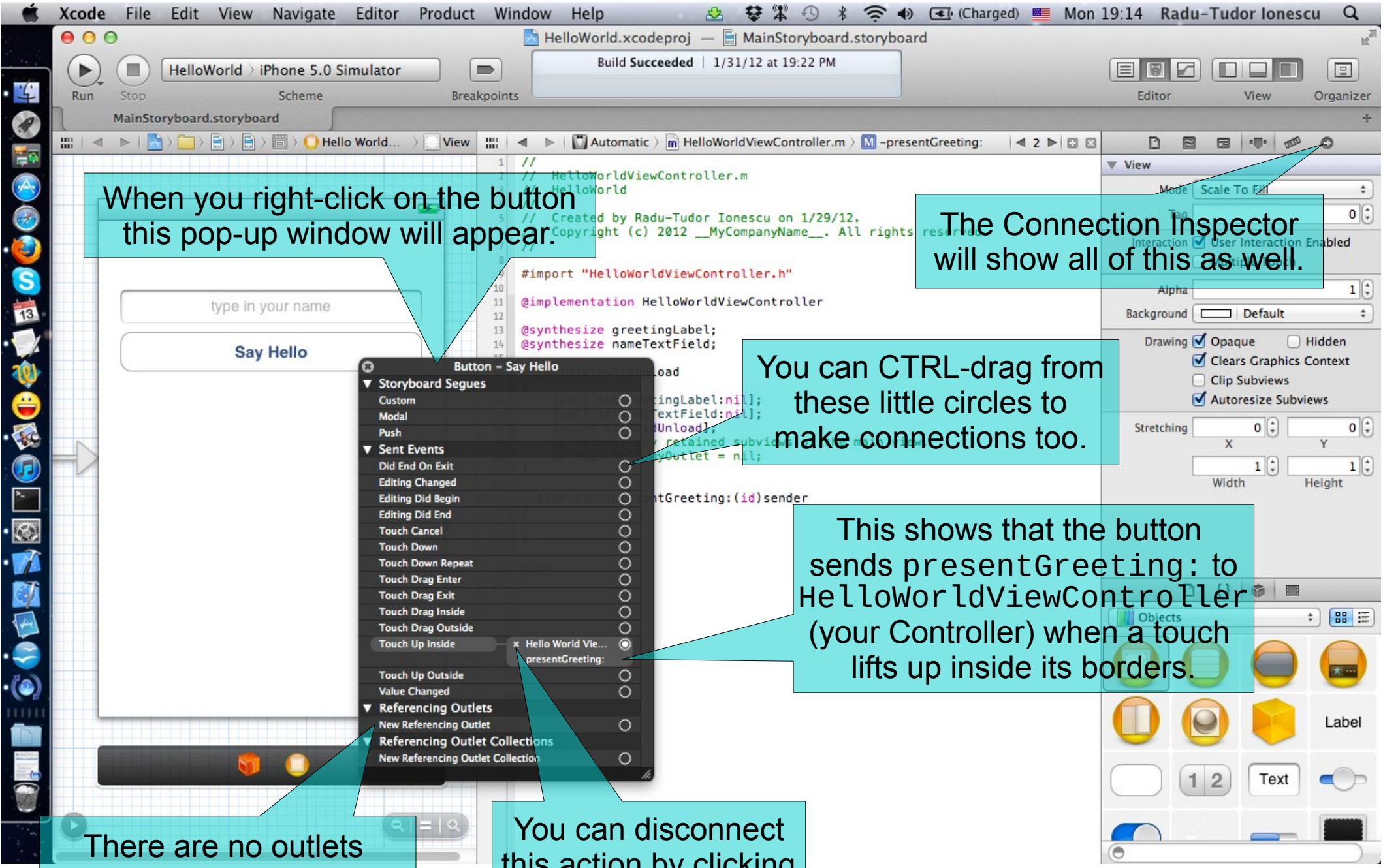
If you click this icon, you'll see all the objects in all storyboards which send this action to your Controller.

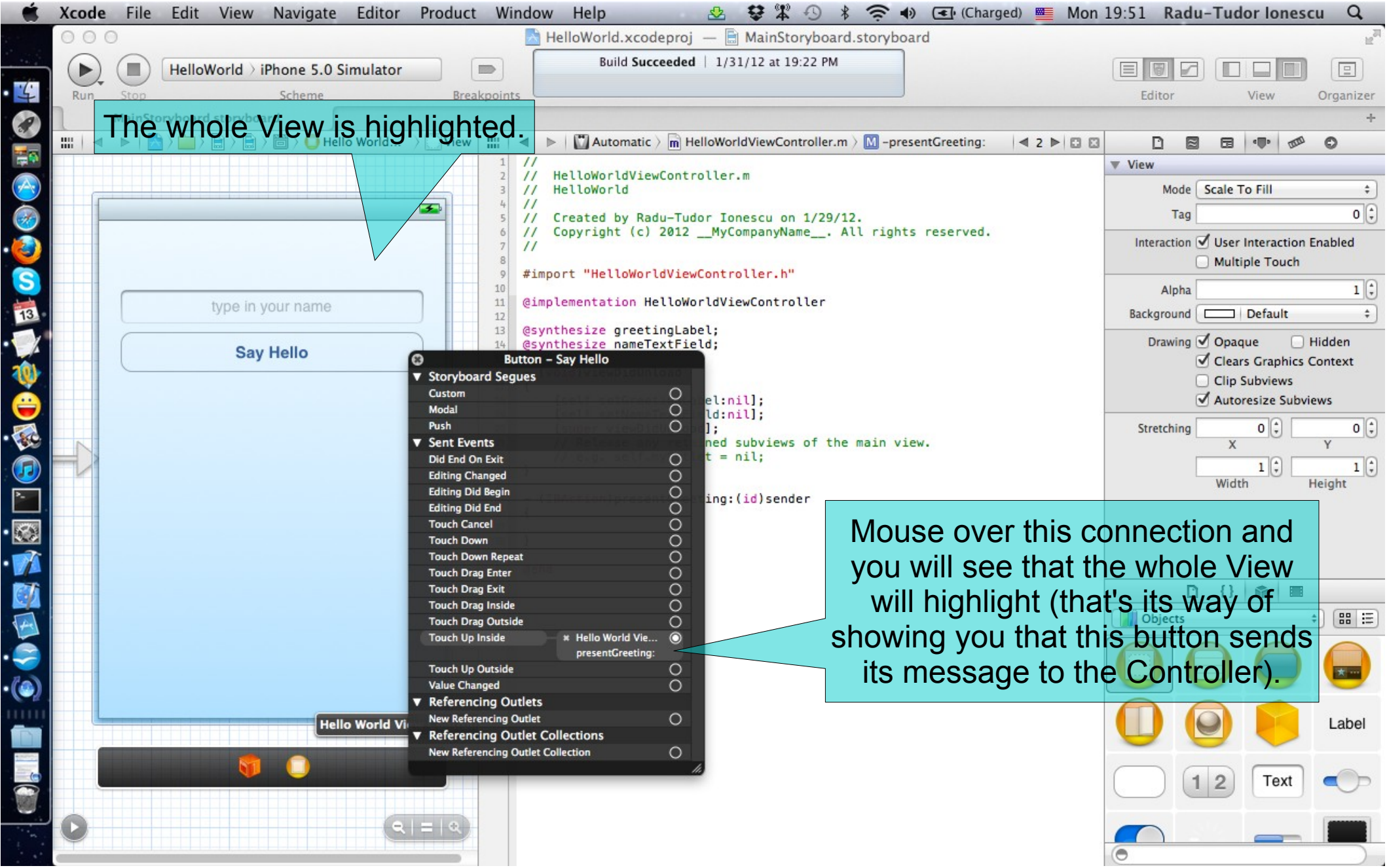
And mousing over one in the list will highlight it (just like mousing over the icon does).

Task 5

Task: Add a button that will trigger the greeting message.

10. Right-click on the button to see its connection. When you start building more complicated user-interfaces, it will be very important to be able to see your outlet and action connections. You can do this by right-clicking on any object in your MVC's View.

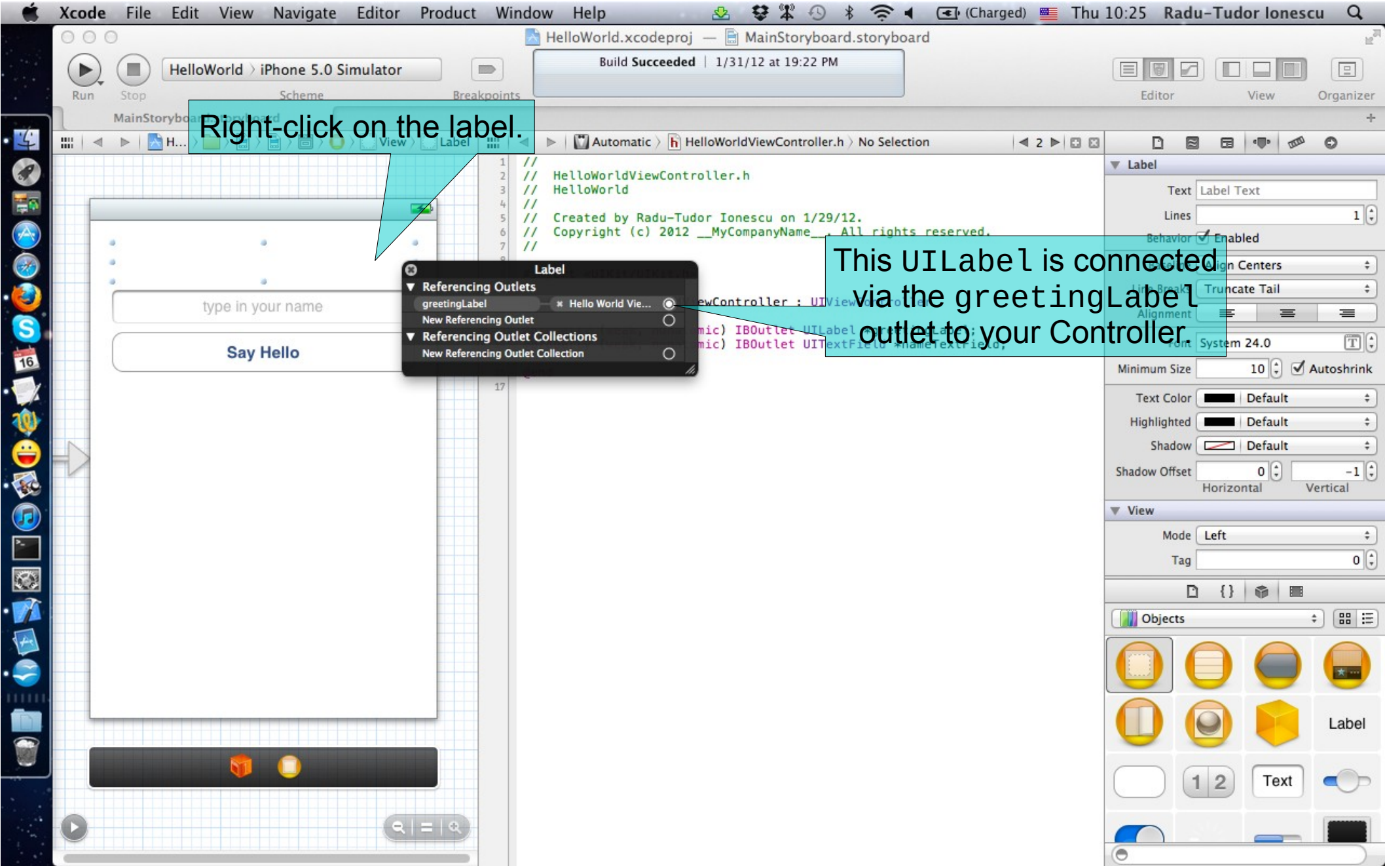

When you right-click on the button this pop-up window will appear.
The Connection Inspector will show all of this as well.
You can CTRL-drag from these little circles to make connections too.
This shows that the button sends presentGreeting: to HelloWorldViewController (your Controller) when a touch lifts up inside its borders.
There are no outlets which point to this button.
You can disconnect this action by clicking on this little x.
Button – Say Hello
Storyboard Segues
Custom
Modal
Push
Sent Events
Did End On Exit
Editing Changed
Editing Did Begin
Editing Did End
Touch Cancel
Touch Down
Touch Down Repeat
Touch Drag Enter
Touch Drag Exit
Touch Drag Inside
Touch Drag Outside
Touch Up Inside
Hello World Vie...
presentGreeting:
Touch Up Outside
Value Changed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
type in your name
Say Hello
HelloWorldViewController.m
-presentGreeting:
#import "HelloWorldViewController.h"
@implementation HelloWorldViewController
@synthesize greetingLabel;
@synthesize nameTextField;
Scale To Fill
User Interaction Enabled
Alpha
Background
Default
Drawing
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
Width
Height
Objects
Label
Text

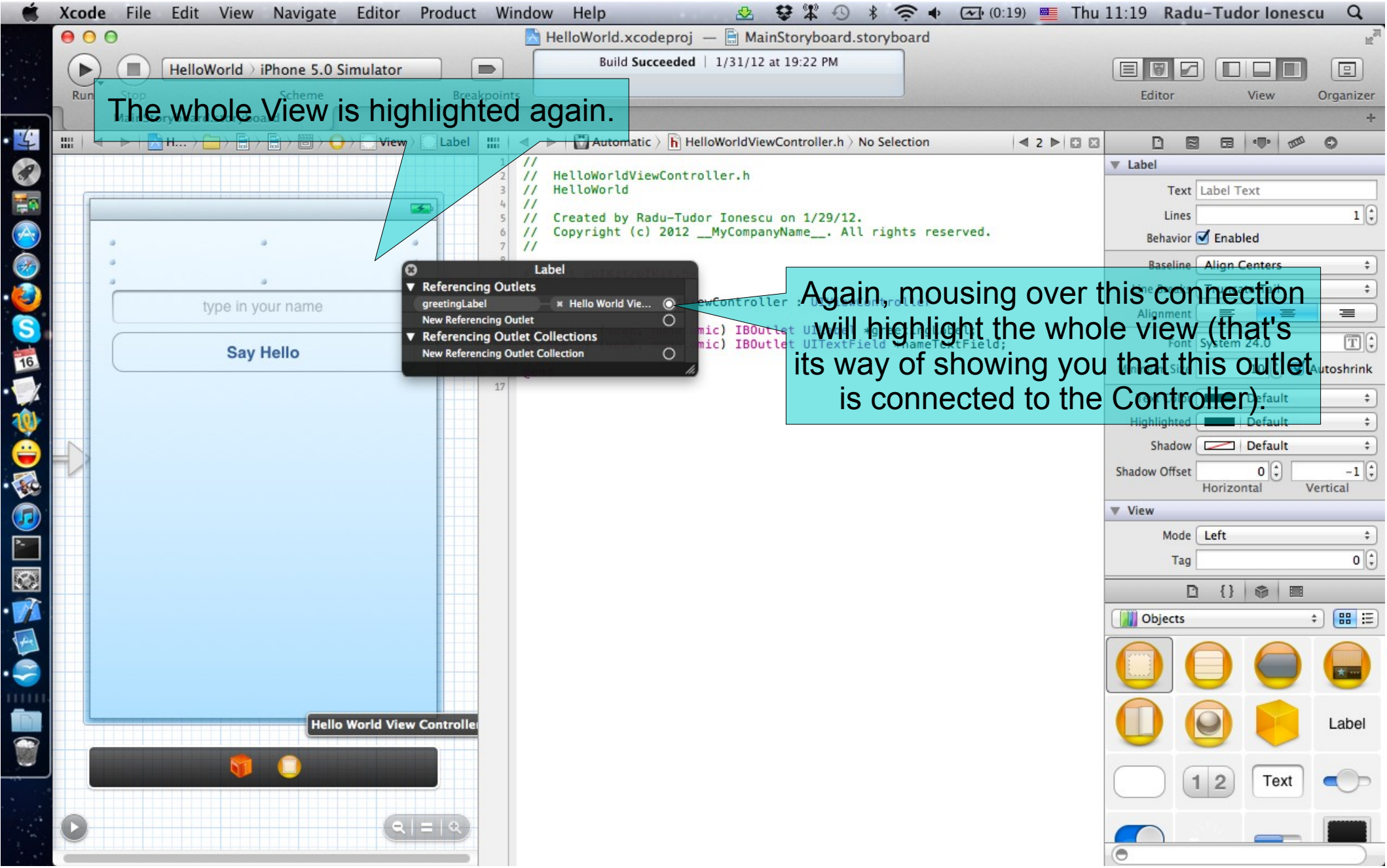

The whole View is highlighted.
Mouse over this connection and you will see that the whole View will highlight (that's its way of showing you that this button sends its message to the Controller).
Xcode File Edit View Navigate Editor Product Window Help
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Build Succeeded 1/31/12 at 19:22 PM
Run Stop Scheme Breakpoints
Editor View Organizer
Automatic HelloWorldViewController.m -presentGreeting:
type in your name
Say Hello
Button – Say Hello
Storyboard Segues
Custom
Modal
Push
Sent Events
Did End On Exit
Editing Changed
Editing Did Begin
Editing Did End
Touch Cancel
Touch Down
Touch Down Repeat
Touch Drag Enter
Touch Drag Exit
Touch Drag Inside
Touch Drag Outside
Touch Up Inside
Hello World Vie...
presentGreeting:
Touch Up Outside
Value Changed
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
View
Mode Scale To Fill
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
Objects
Label
Text

Task 5

Task: Add a button that will trigger the greeting message.

11. Right-click on the `greetingLabel` to see its connection.


Xcode File Edit View Navigate Editor Product Window Help
Thu 10:25 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Build Succeeded 1/31/12 at 19:22 PM
Run Stop Scheme Breakpoints Editor View Organizer
Right-click on the label.
type in your name
Say Hello
Label
Referencing Outlets
greetingLabel
Hello World Vie...
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
This UILabel is connected via the greetingLabel outlet to your Controller.
Automatic HelloWorldViewController.h No Selection
//
// HelloWorldViewController.h
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
Label
Text Label Text
Lines 1
Behavior Enabled
Alignment Centers
Line Break Truncate Tail
Font System 24.0
Minimum Size 10 Autoshrink
Text Color Default
Highlighted Default
Shadow Default
Shadow Offset 0 -1
Horizontal Vertical
View
Mode Left
Tag 0
Objects
Label
Text

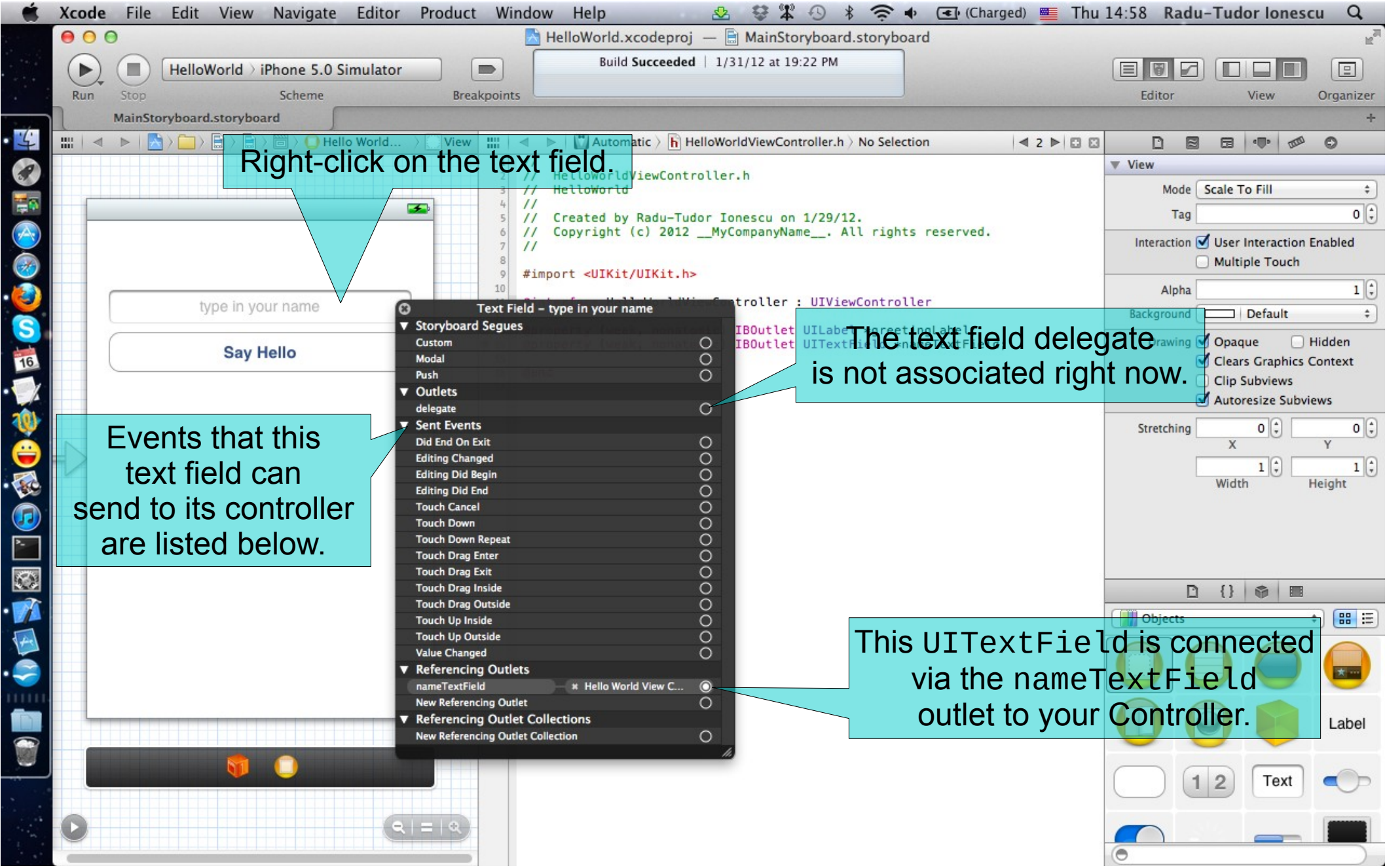

The whole View is highlighted again.
Again, mousing over this connection will highlight the whole view (that's its way of showing you that this outlet is connected to the Controller).
Xcode File Edit View Navigate Editor Product Window Help
Thu 11:19 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Build Succeeded 1/31/12 at 19:22 PM
Run Stop Scheme Breakpoints
Editor View Organizer
HelloWorldViewController.h No Selection
//
// HelloWorldViewController.h
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
Label
Referencing Outlets
greetingLabel
Hello World Vie...
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
type in your name
Say Hello
Hello World View Controlle
Label
Text Label Text
Lines 1
Behavior Enabled
Baseline Align Centers
Alignment
Font System 24.0
Autoshrink
Highlighted Default
Shadow Default
Shadow Offset 0 -1
Horizontal Vertical
View
Mode Left
Tag 0
Objects
Label
Text

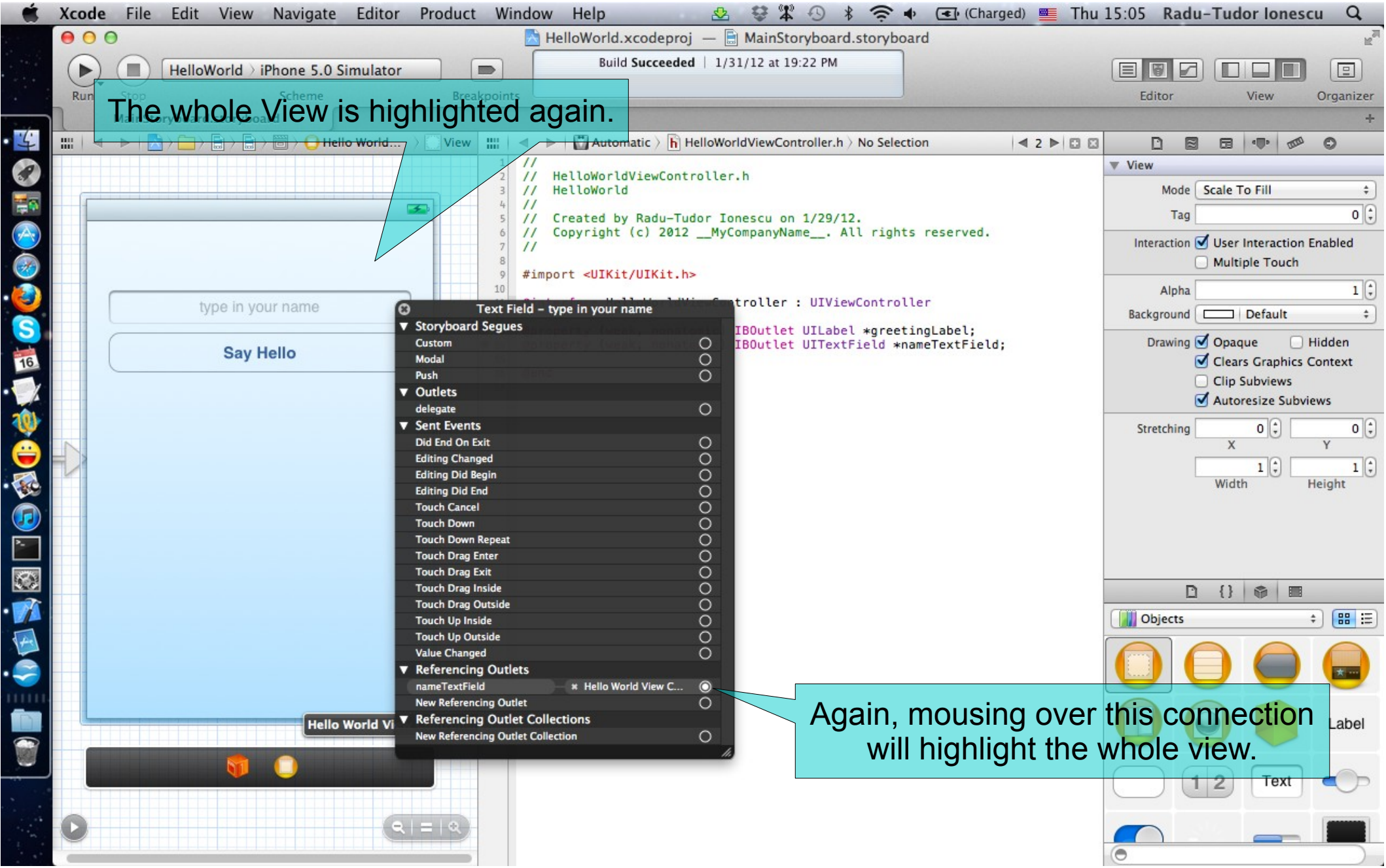
Task 5

Task: Add a button that will trigger the greeting message.

12. Right-click on the `nameTextField` to see its connection.

13. Notice the **delegate** outlet. A text field delegate responds to editing-related messages from the text field. You can use the delegate to respond to the text entered by the user and to some special commands, such as when the return button is pressed.


Right-click on the text field.
Events that this text field can send to its controller are listed below.
The text field delegate is not associated right now.
This UITextField is connected via the nameTextField outlet to your Controller.
Text Field – type in your name
Storyboard Segues
Custom
Modal
Push
Outlets
delegate
Sent Events
Did End On Exit
Editing Changed
Editing Did Begin
Editing Did End
Touch Cancel
Touch Down
Touch Down Repeat
Touch Drag Enter
Touch Drag Exit
Touch Drag Inside
Touch Drag Outside
Touch Up Inside
Touch Up Outside
Value Changed
Referencing Outlets
nameTextField
Hello World View C...
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
type in your name
Say Hello
HelloWorldViewController.h
Mode
Scale To Fill
Tag
Interaction
User Interaction Enabled
Multiple Touch
Alpha
Background
Default
Opaque
Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching
Width
Height
Objects
Label
Text


Xcode File Edit View Navigate Editor Product Window Help
Thu 15:05 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Build Succeeded 1/31/12 at 19:22 PM
Run Stop Scheme Breakpoints
Editor View Organizer
The whole View is highlighted again.
Hello World... View
Automatic HelloWorldViewController.h No Selection
//
// HelloWorldViewController.h
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//
#import <UIKit/UIKit.h>
IBOutlet UILabel *greetingLabel;
IBOutlet UITextField *nameTextField;
type in your name
Say Hello
Hello World Vi
Text Field – type in your name
Storyboard Segues
Custom
Modal
Push
Outlets
delegate
Sent Events
Did End On Exit
Editing Changed
Editing Did Begin
Editing Did End
Touch Cancel
Touch Down
Touch Down Repeat
Touch Drag Enter
Touch Drag Exit
Touch Drag Inside
Touch Drag Outside
Touch Up Inside
Touch Up Outside
Value Changed
Referencing Outlets
nameTextField
Hello World View C...
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
View
Mode Scale To Fill
Tag 0
Interaction User Interaction Enabled
Multiple Touch
Alpha 1
Background Default
Drawing Opaque Hidden
Clears Graphics Context
Clip Subviews
Autoresize Subviews
Stretching 0 0 X Y
1 1 Width Height
Objects
Label
Text
Again, mousing over this connection will highlight the whole view.

Task 5

Task: Add a button that will trigger the greeting message.

14. Right-click on the icon that represents the Controller to see its connection.
15. Mouse over the “Say Hello” button entry in the pop-up window.
16. Mouse over the `greetingLabel` outlet in the pop-up window.

Notice that the button highlights.

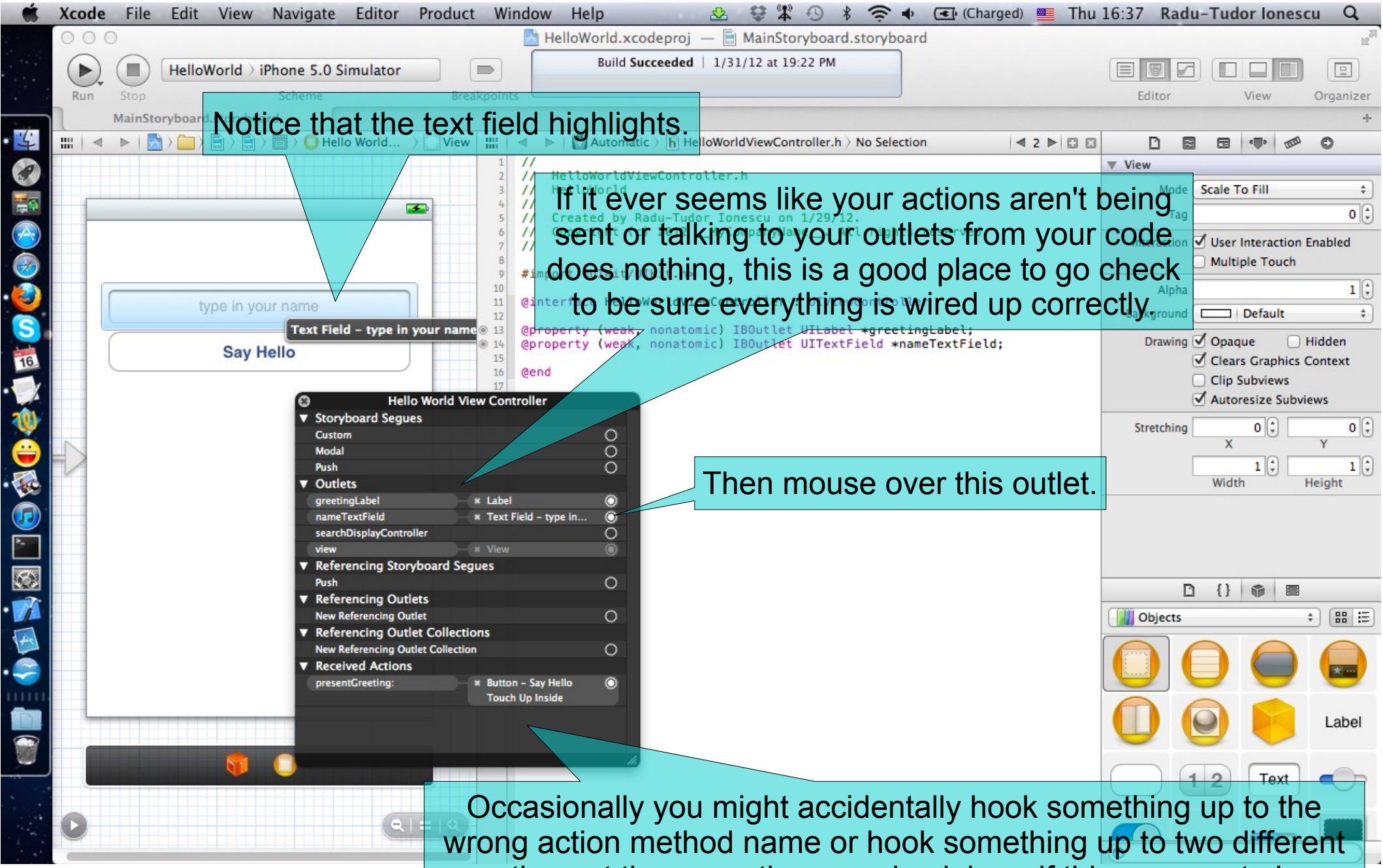
Then mouse over this "Button - Say Hello" entry.

Right-click on this icon that represents your Controller.

Notice that the `greetingsLabel` highlights.

Then mouse over this outlet.

Don't worry about this view outlet, we'll explain it later in this course.


Notice that the text field highlights.
If it ever seems like your actions aren't being sent or talking to your outlets from your code does nothing, this is a good place to go check to be sure everything is wired up correctly.
Then mouse over this outlet.
Occasionally you might accidentally hook something up to the wrong action method name or hook something up to two different actions at the same time, so check here if things seem to be acting sort of messed up when it comes to outlets and actions.
Text Field – type in your name
Hello World View Controller
Storyboard Segues
Custom
Modal
Push
Outlets
greetingLabel
Label
nameTextField
Text Field – type in...
searchDisplayController
view
View
Referencing Storyboard Segues
Referencing Outlets
New Referencing Outlet
Referencing Outlet Collections
New Referencing Outlet Collection
Received Actions
presentGreeting:
Button – Say Hello
Touch Up Inside

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

1. Switch back to HelloWorldViewController.m in Assitant Editor.

2. Hide the Utilities area.

Switch to the Controller's implementation.

We won't need the Utilities area for a while, so close it.

```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

@interface HelloWorldViewController : UIViewController

@property (weak, nonatomic) IBOutlet UILabel *greetingLabel;
@property (weak, nonatomic) IBOutlet UITextField *nameTextField;

@end
```

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

3. It is time to write the code inside `presentGreeting:` that will get executed whenever the "Say Hello" button gets touched. Let's start by declaring a local variable that will store the user's name.

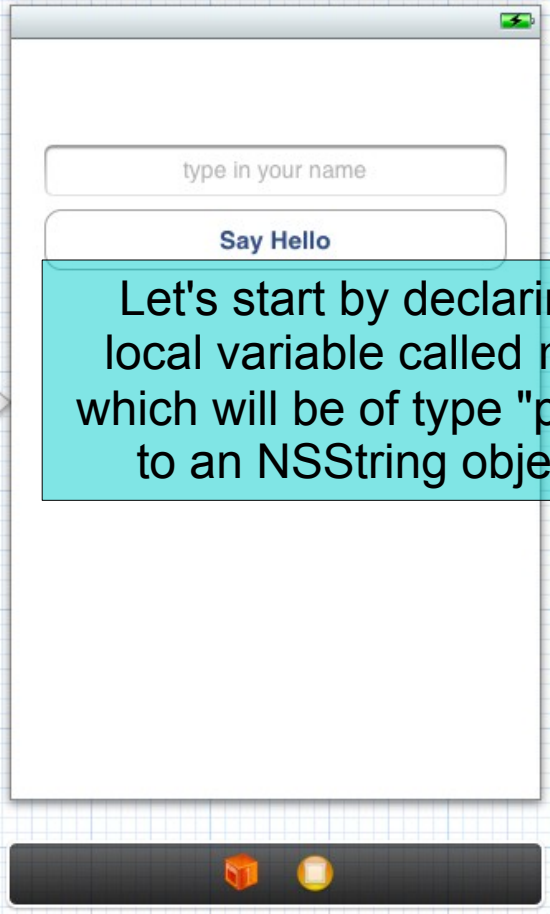
Xcode File Edit View Navigate Editor Product Window Help (Charged) Thu 17:08 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Build Succeeded | 1/31/12 at 19:22 PM Project 1 Editor View Organizer

MainStoryboard.storyboard

Hello World View Contr... › View



Automatic › HelloWorldViewController.m › -presentGreeting:

```
1  //
2  //  HelloWorldViewController.m
3  //  HelloWorld
4  //
5  //  Created by Radu-Tudor Ionescu on 1/29/12.
6  //  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
7  //
8
9  #import "HelloWorldViewController.h"
10
11 @implementation HelloWorldViewController
12
13 @synthesize greetingLabel;
14 @synthesize nameTextField;
15
16 - (void)viewDidUnload
17 {
18     [self setGreetingLabel:nil];
19     [self setNameTextField:nil];
20     [super viewDidUnload];
21     // Release any retained subviews of the main view.
22     // e.g. self.myOutlet = nil;
23 }
24
25 - (IBAction)presentGreeting:(id)sender
26 {
27     NSString *name = |
28 }
29
30 @end
31
```

Let's start by declaring a local variable called name which will be of type "pointer to an NSString object".

Remember that **all** Objective-C objects are allocated in the heap and we keep pointers to them. It would never be right to say "`NSString name`" (without the `*`).

Yes, we could say `id name` = here. But we should use static typing whenever possible.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

4. To store the user's name in the local variable you need to get the text inside the text field.

To send a message to an Objective-C object we use a syntax that starts with an open square bracket `[`, then a pointer to the object we want to send the message to (`nameTextField`) then a space, then the name of the message to send (`text`). The message sending syntax ends with a `]` to match the `[` it started with.

Start with an open square bracket.

When you agree with Xcode suggestion you can hit Enter to autocomplete.

Xcode tries diligently to help you as you type.
It is smart about sensing what's going on in your code.
Xcode is suggesting variables that start with "nam…".

Xcode File Edit View Navigate Editor Product Window Help (Charged) Thu 17:48 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Build Succeeded | 1/31/12 at 19:22 PM Project 1 Editor View Organizer

MainStoryboard.storyboard

Hello World View Contr... › View

type in your name

Say Hello

Automatic › HelloWorldViewController.m › -presentGreeting:

```
1  //
2  //  HelloWorldViewController.m
3  //  HelloWorld
4  //
5  //  Created by Radu-Tudor Ionescu on 1/29/12.
6  //  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
7  //
8
9  #import "HelloWorldViewController.h"
10
11 @implementation HelloWorldViewController
12
13 @synthesize greetingLabel;
14 @synthesize nameTextField;
15
16 - (void)viewDidUnload
17 {
18     [self setGreetingLabel:nil];
19     [self setNameTextField:nil];
20     [super viewDidUnload];
21     // Release any retained subviews of the main view.
22     // e.g. self.myOutlet = nil;
23 }
24
25 - (IBAction)presentGreeting:(id)sender
26 {
27     NSString *name = [nameTextField text];
28 }
29
30 @end
31
```

This method has no arguments. We'll see a method with an argument later.

It seems that we have a problem! This little triangle is a warning that there's a problem with this line of code. A red dot here would mean an error in the code (which won't compile).

`UITextField` objects respond to the message `text` which returns an `NSString` containing the text that was typed in inside the text field.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

5. Click on the triangle to find out what warning is generated by our line of code. Warnings and errors show up in Xcode as you write your code. Although Xcode projects can compile even with warnings, it is recommended to eliminate all warnings from your project.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Fri 11:37 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Build Succeeded | 1/31/12 at 19:22 PM Project 1 Editor View Organizer

MainStoryboard.storyboard

Hello World View Contr... › View

Warning triangle appears here too.

Another place the warning triangle appears.

type in your name

Say Hello

Click on the triangle to find out what the warning is.

Automatic › HelloWorldViewController.m › -presentGreeting:

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel;
@synthesize nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *name = [nameTextField text];
}

@end
```

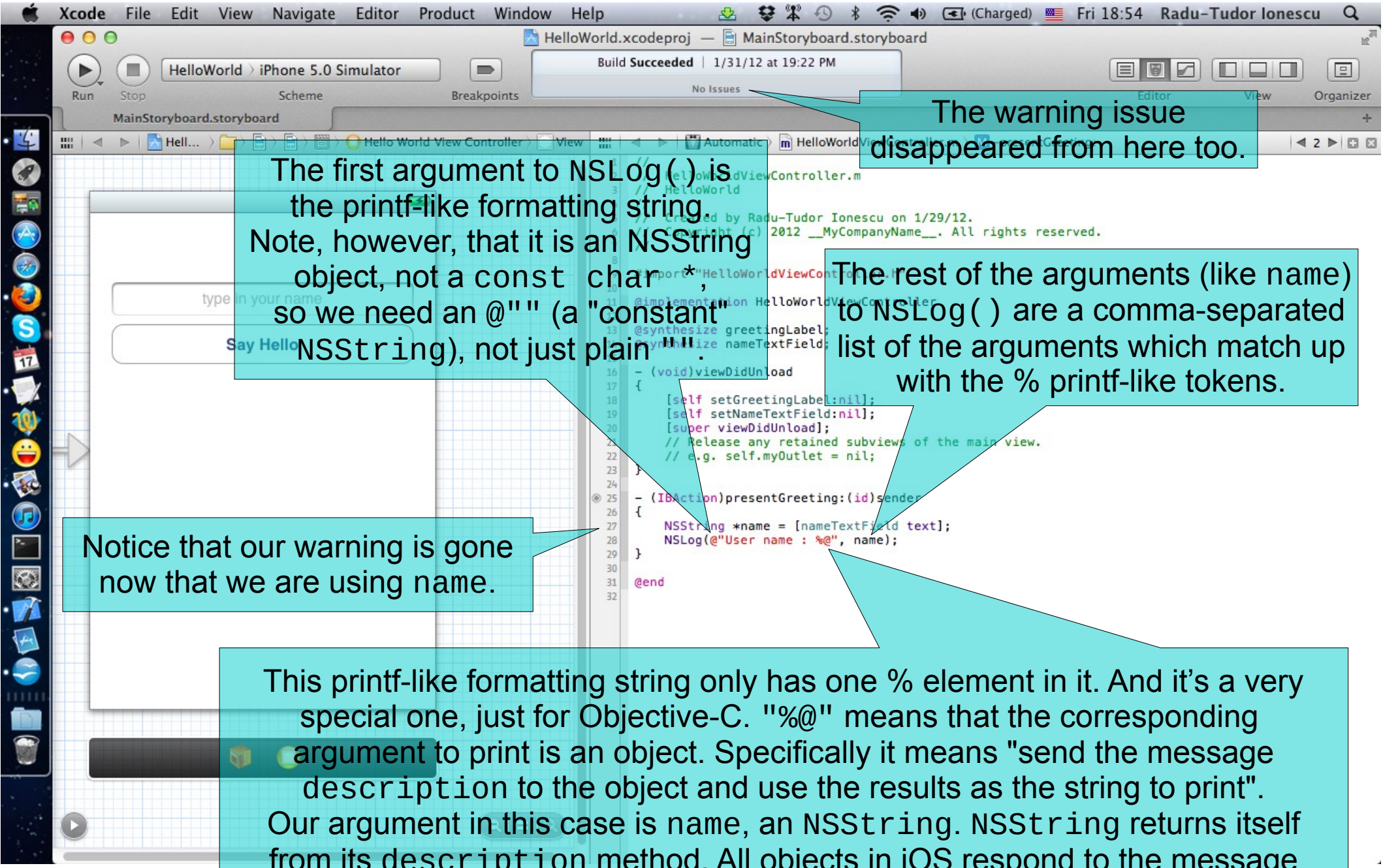
Unused variable 'name'

This warning appears to be correct. We do not (yet) use the local variable name in this method.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

6. Add an `NSLog()` to print the name typed in by the user. A very simple debugging technique is to log information to the console. This is very easy to do in Xcode. There is a printf-like function whose output goes to the console called `NSLog()`.



The warning issue disappeared from here too.

The first argument to `NSLog()` is the printf-like formatting string. Note, however, that it is an NSString object, not a `const char *`, so we need an `@""` (a "constant" `NSString`), not just plain `""`.

The rest of the arguments (like `name`) to `NSLog()` are a comma-separated list of the arguments which match up with the % printf-like tokens.

Notice that our warning is gone now that we are using `name`.

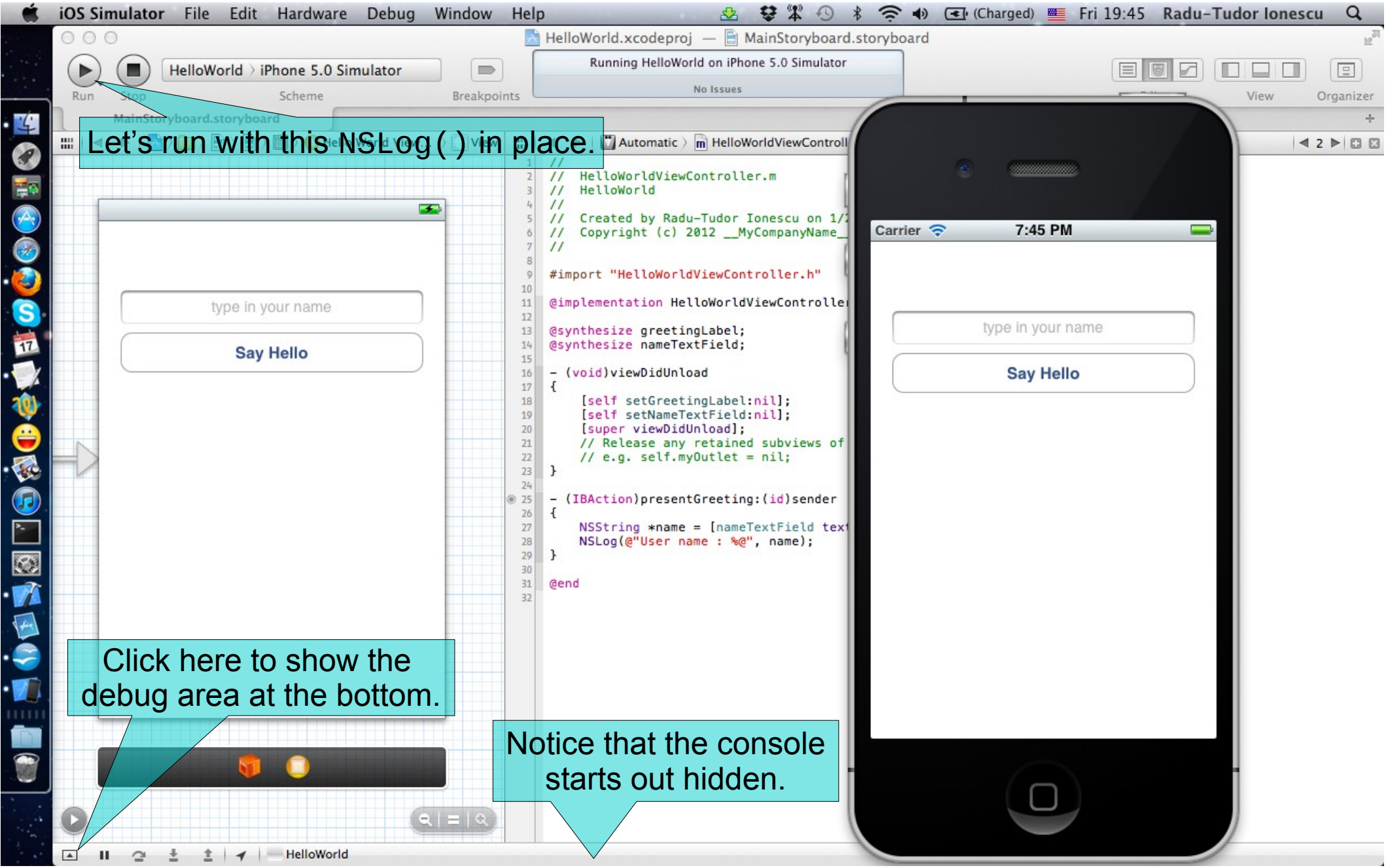
This printf-like formatting string only has one % element in it. And it's a very special one, just for Objective-C. "%@" means that the corresponding argument to print is an object. Specifically it means "send the message `description` to the object and use the results as the string to print". Our argument in this case is `name`, an `NSString`. `NSString` returns itself from its `description` method. All objects in iOS respond to the message `description` because `NSObject`, the root of all classes in iOS, implements it.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

7. Run the application in the Simulator with `NSLog()` in place.

8. Show up the debug area.


iOS Simulator File Edit Hardware Debug Window Help
(Charged) Fri 19:45 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Running HelloWorld on iPhone 5.0 Simulator
No Issues
Run Stop Scheme Breakpoints View Organizer
Let's run with this NSLog( ) in place.
Automatic HelloWorldViewControll
//
// HelloWorldViewController.m
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/2
// Copyright (c) 2012 __MyCompanyName__
//
#import "HelloWorldViewController.h"
@implementation HelloWorldViewControlle
@synthesize greetingLabel;
@synthesize nameTextField;
- (void)viewDidUnload
{
[self setGreetingLabel:nil];
[self setNameTextField:nil];
[super viewDidUnload];
// Release any retained subviews of
// e.g. self.myOutlet = nil;
}
- (IBAction)presentGreeting:(id)sender
{
NSString *name = [nameTextField text
NSLog(@"User name : %@", name);
}
@end
type in your name
Say Hello
Click here to show the debug area at the bottom.
Notice that the console starts out hidden.
Carrier 7:45 PM
type in your name
Say Hello
HelloWorld

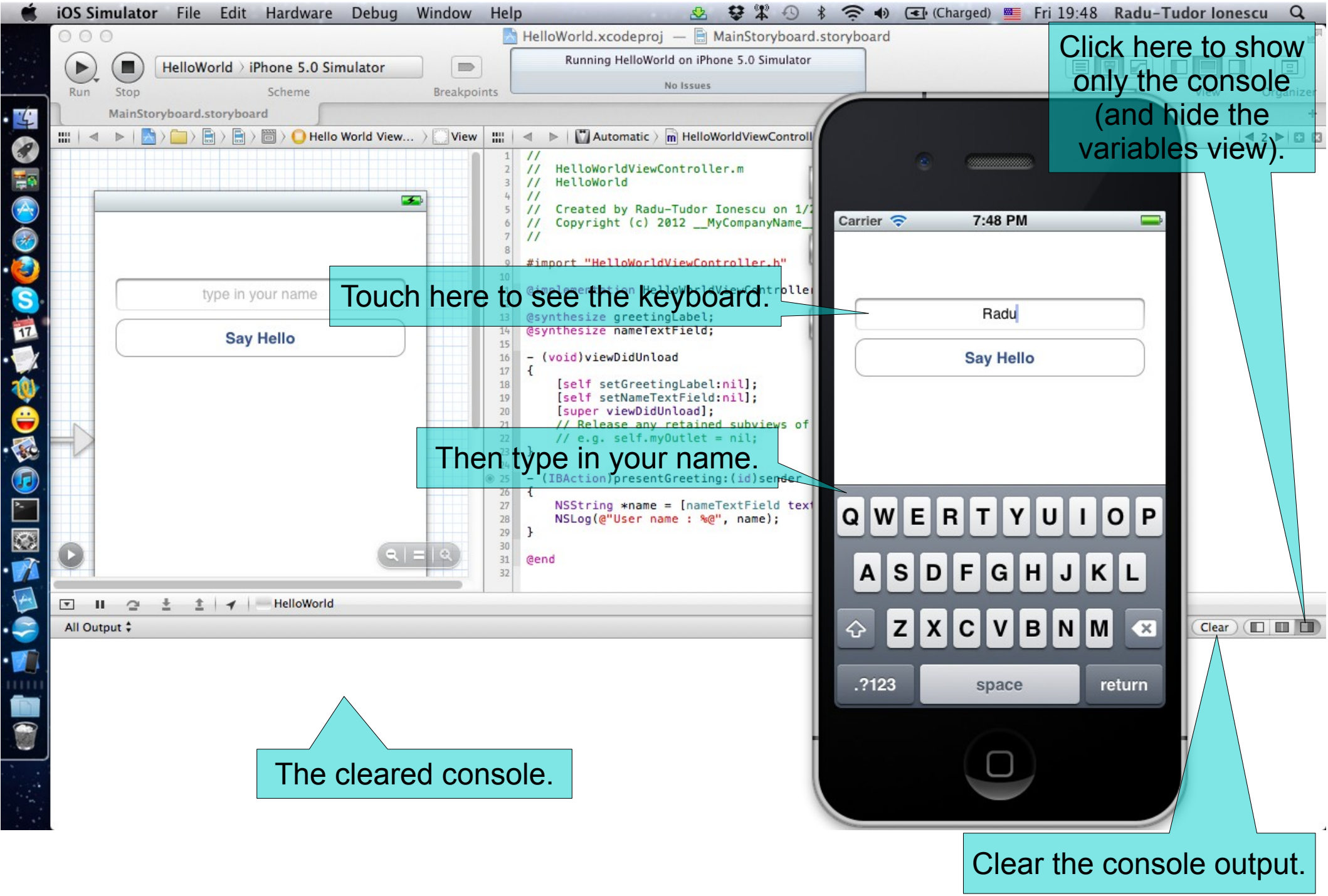
Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

9. Hide the variables view in the debug area.

10. Clear the console.

11. Type in your name in the text field.

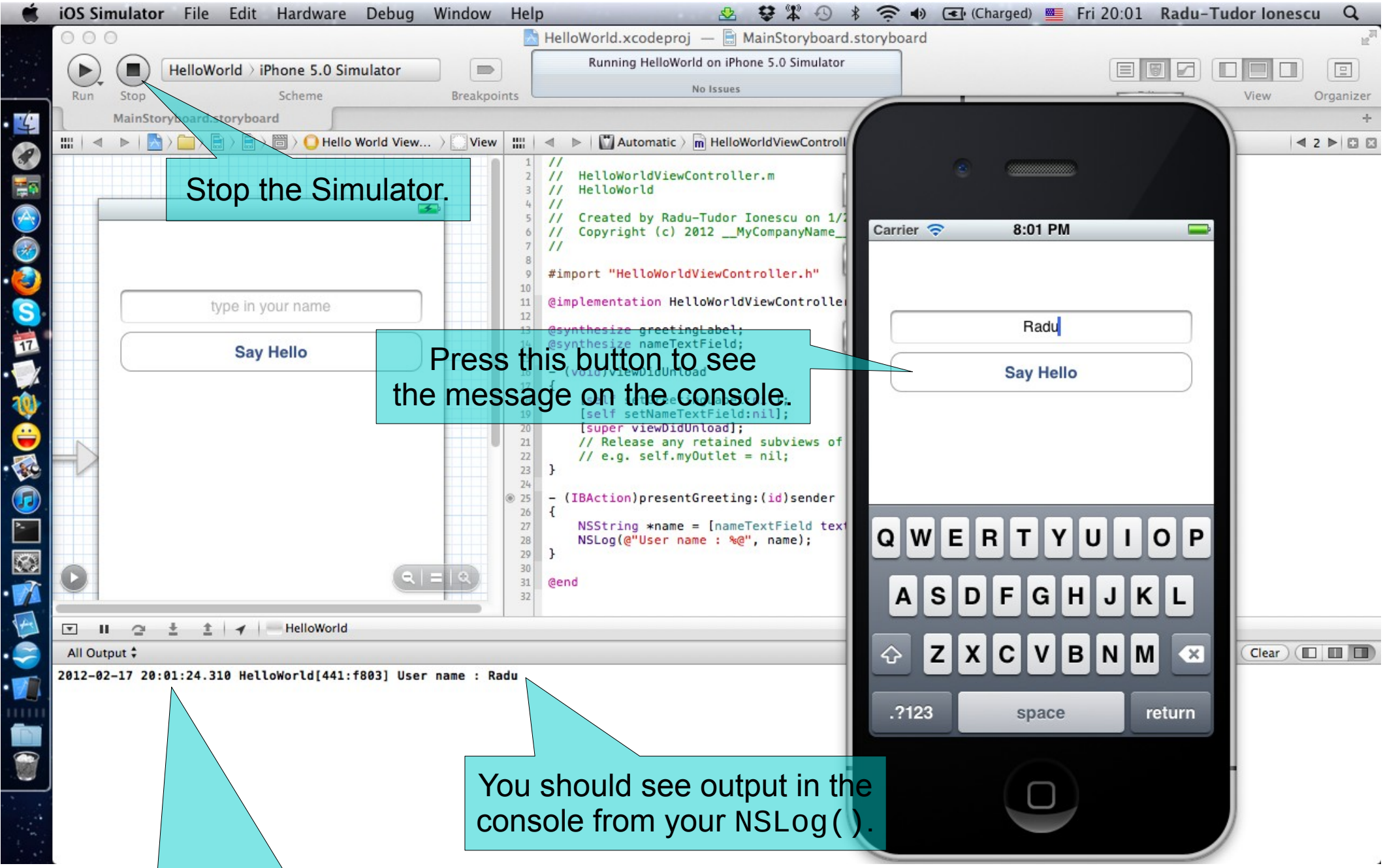

iOS Simulator File Edit Hardware Debug Window Help
Fri 19:48 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Run Stop Scheme Breakpoints
Running HelloWorld on iPhone 5.0 Simulator
No Issues
MainStoryboard.storyboard
Hello World View... View
Automatic HelloWorldViewControll
type in your name
Say Hello
//
// HelloWorldViewController.m
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/2
// Copyright (c) 2012 __MyCompanyName__
//
#import "HelloWorldViewController.h"
@implementation HelloWorldViewController
@synthesize greetingLabel;
@synthesize nameTextField;
- (void)viewDidUnload
{
[self setGreetingLabel:nil];
[self setNameTextField:nil];
[super viewDidUnload];
// Release any retained subviews of
// e.g. self.myOutlet = nil;
}
- (IBAction)presentGreeting:(id)sender
{
NSString *name = [nameTextField text
NSLog(@"User name : %@", name);
}
@end
HelloWorld
All Output
Clear
Carrier 7:48 PM
Radu
Say Hello
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
.?123 space return
Click here to show only the console (and hide the variables view).
Touch here to see the keyboard.
Then type in your name.
The cleared console.
Clear the console output.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

12. Press the “Say Hello” button. Notice the console output.

13. Stop the simulator.


iOS Simulator File Edit Hardware Debug Window Help
Fri 20:01 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Run Stop Scheme Breakpoints
Running HelloWorld on iPhone 5.0 Simulator
No Issues
View Organizer
Stop the Simulator.
type in your name
Say Hello
Press this button to see the message on the console.
//
// HelloWorldViewController.m
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/2
// Copyright (c) 2012 __MyCompanyName__
//
#import "HelloWorldViewController.h"
@implementation HelloWorldViewController
@synthesize greetingLabel;
@synthesize nameTextField;
[self setNameTextField:nil];
[super viewDidUnload];
// Release any retained subviews of
// e.g. self.myOutlet = nil;
}
- (IBAction)presentGreeting:(id)sender
{
NSString *name = [nameTextField text
NSLog(@"User name : %@", name);
}
@end
Carrier 8:01 PM
Radu
Say Hello
All Output
2012-02-17 20:01:24.310 HelloWorld[441:f803] User name : Radu
Clear
You should see output in the console from your NSLog( ).
All console logs are timestamped.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

14. Get quick help for the `text` method of the `UITextField`. Xcode contains extensive reference documentation for all methods/classes. A quick lookup can be done simply by selecting a method or class name and choosing "Quick Help for Selected Item" from the Help menu.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Fri 20:30 Radu-Tudor Ionescu

Search
Xcode Help
Xcode User Guide
Documentation and API Reference ⌥⌘?
Release Notes
Quick Help for Selected Item ^⌘?
Search Documentation for Selected Text ^⌥⌘/

HelloWorld › iPhone 5.0 Simulator
Run Stop Scheme Breakpoints Editor View Organizer

MainStoryboard.storyboard

Hello World View... › View

type in your name
Say Hello

```
//
//
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController
@synthesize nameTextField;
- (void)viewDidUnload
{
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *name = [nameTextField text];
    NSLog(@"User name : %@", name);
}

@end
```

Useful key-binding here. All key-bindings are settable in Xcode's Preferences.

Or you can hold down the option key and click on a term to get quick help. Hold down option and click on on the text method to get quick help for it.

text

Name: text
Declaration: @property(nonatomic, copy) NSString *text
Availability: iOS (2.0 and later)
Abstract: The text displayed by the text field.
Declared In: UITextField.h
Reference: UITextField Class Reference

Here you can see that the `text` method in `UILabel` is actually a `@property` (it's the getter of the text property).

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

15. Use dot notation for the `text` `@property`. It turns out that `@propertys` are so important that there is a special Objective-C syntax just for `@property` setters and getters. It's called "dot notation". We can express calling the getter of our `@property` using dot notation instead. These are two syntactically different expressions of **exactly** the same thing.

16. It's a good time to save the implementation file.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Fri 20:48 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Finished running HelloWorld on iPhone 5.0 Simulato No Issues Editor View Organizer

MainStoryboard.storyboard

Hello World View... › View

type in your name

Say Hello

Automatic › HelloWorldViewController.m › -presentGreeting:

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 10/12/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel;
@synthesize nameTextField;
- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *name = nameTextField.text;
    NSLog(@"User name : %@", name);
}

@end
```

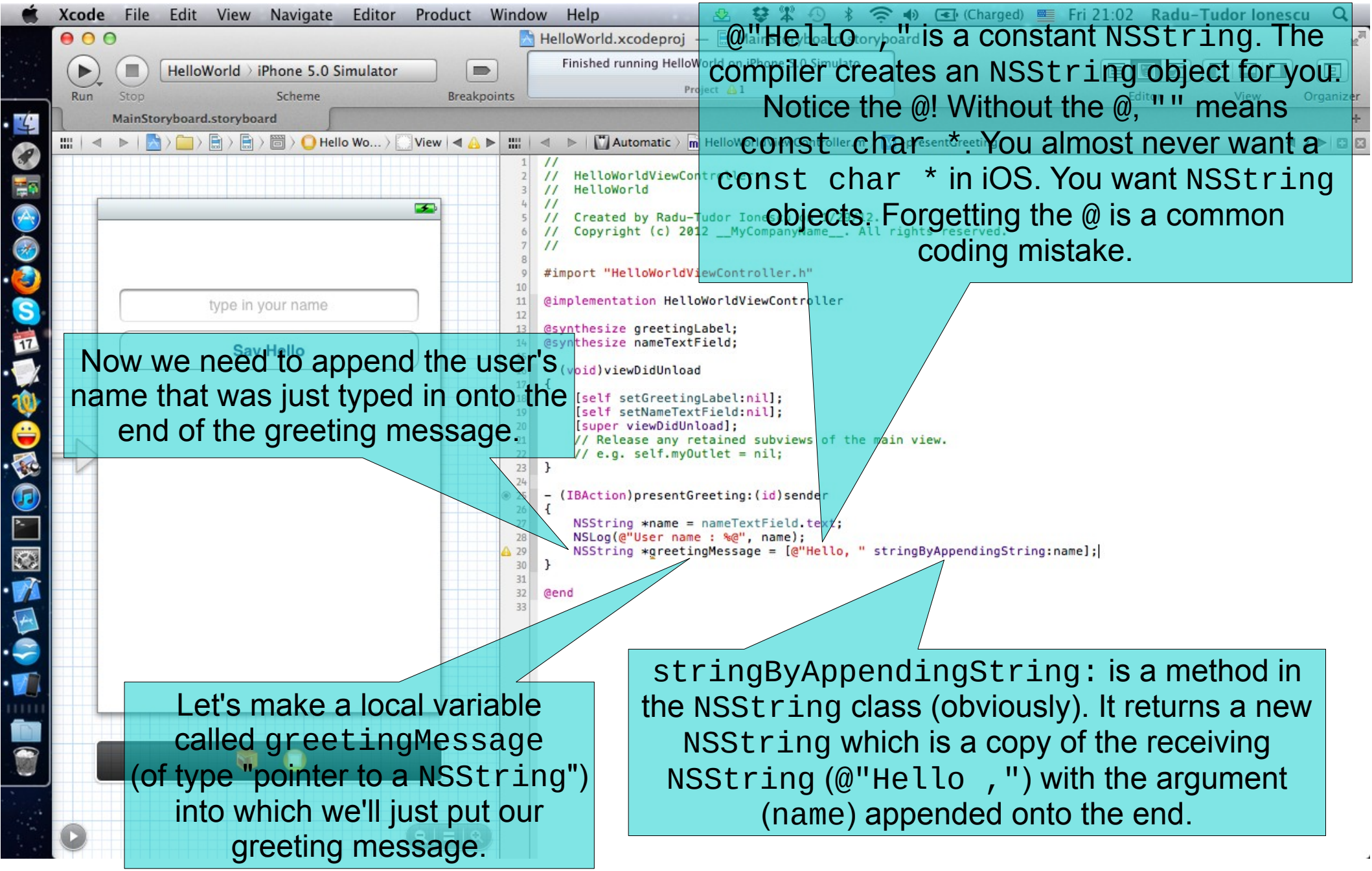
This grayed out icon means this file is not saved. Click anywhere in this file and choose Save from the File menu or use the CMD + S shortcut to save.

Now that we know that `text` is actually a `@property`, let's use dot notation.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

17. Build a local `NSString` variable with the greeting message. Now that we have the user's name, we need to update our display with the greeting message. This actually only takes one line of code, but we'll break it down into steps. The first step is to append the user's name to the greeting message.



@"Hello, " is a constant `NSString`. The compiler creates an `NSString` object for you. Notice the @! Without the @, `""` means `const char *`. You almost never want a `const char *` in iOS. You want `NSString` objects. Forgetting the @ is a common coding mistake.

Now we need to append the user's name that was just typed in onto the end of the greeting message.

Let's make a local variable called `greetingMessage` (of type "pointer to a `NSString`") into which we'll just put our greeting message.

`stringByAppendingString:` is a method in the `NSString` class (obviously). It returns a new `NSString` which is a copy of the receiving `NSString` (`@"Hello, "`) with the argument (`name`) appended onto the end.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

18. Build the greeting message without using the local variable `name` (note that we actually don't need it).

19. Delete the first two lines of code in `presentGreeting:` method.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Mon 18:49 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints

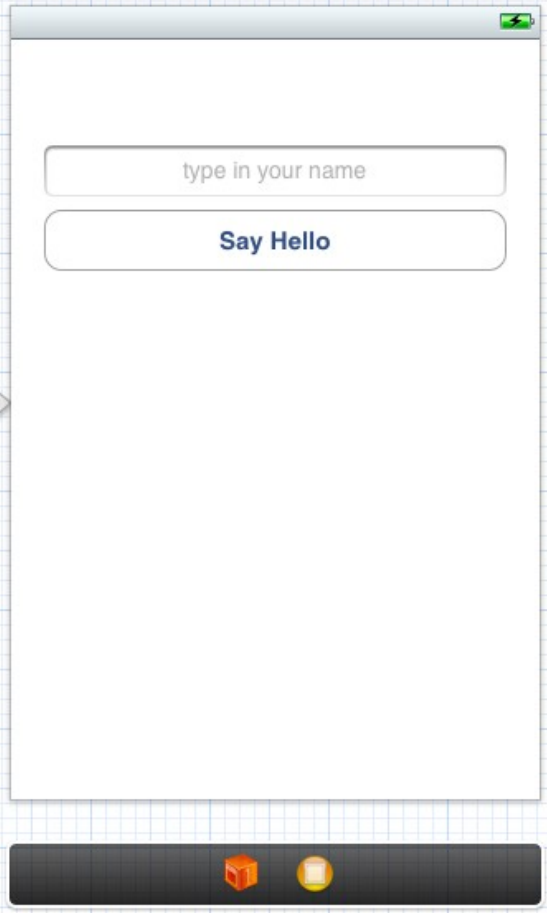
Build Succeeded | 2/17/12 at 19:34 PM

Project ⚠1

Editor View Organizer

MainStoryboard.storyboard

MainStoryboar... › No Selection



Automatic › HelloWorldViewController.m › -presentGreeting:

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel;
@synthesize nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *name = nameTextField.text;
    NSLog(@"User name : %@", name);
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:name];
}

@end
```

There's really no need for this local variable name. So let's select and copy its value (`nameTextField.text`).

Then select where we use it and paste.

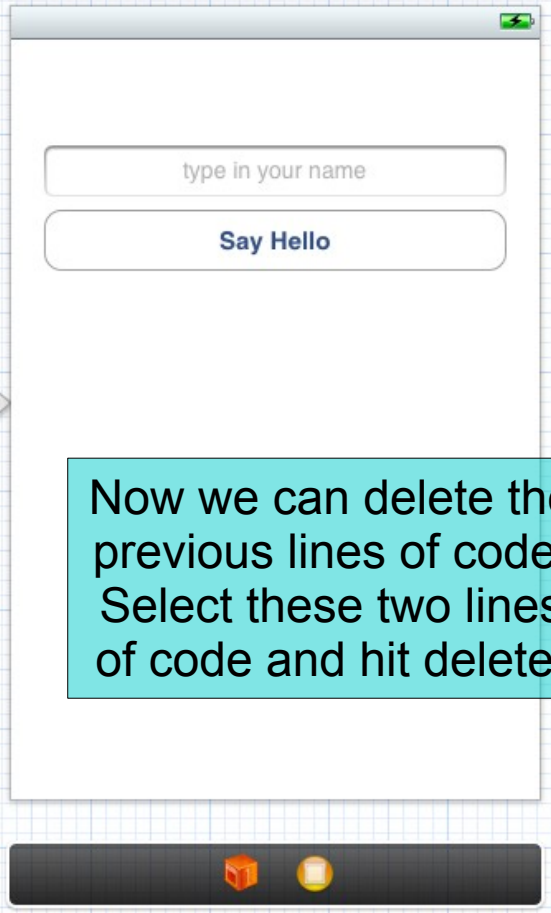
Xcode File Edit View Navigate Editor Product Window Help (Charged) Mon 18:49 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Build Succeeded | 2/17/12 at 19:34 PM Project 1 Editor View Organizer

MainStoryboard.storyboard

MainStoryboar... › No Selection



Now we can delete the previous lines of code. Select these two lines of code and hit delete.

Automatic › HelloWorldViewController.m › -presentGreeting:

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel;
@synthesize nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *name = nameTextField.text;
    NSLog(@"User name : %@", name);
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:nameTextField.text];
}

@end
```

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

20. Rename the instance variable of the `nameTextField` `@property` by adding an underscore. Accessing the synthesized instance variable directly (and thus not calling the getter) is bad. There is no compiler warning to help us notice that, but changing the name used by `@synthesize` to create its instance variable will make it very clear when we accidentally forget to use the getter.

21. Also rename the instance variable of the `greetingLabel` `@property`.

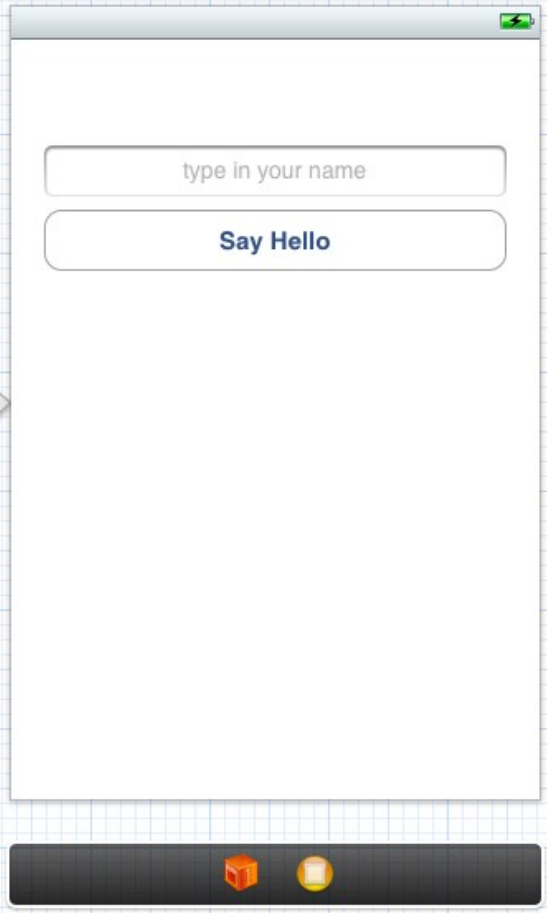
Xcode File Edit View Navigate Editor Product Window Help (Charged) Mon 20:28 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Build Succeeded | 2/17/12 at 19:34 PM Project 1 Editor View Organizer

MainStoryboard.storyboard

MainStoryboar... › No Selection



Automatic › HelloWorldViewController.m › greetingLabel

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel;
@synthesize nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:nameTextField.text];
}

@end
```

There is no compiler warning to help us notice the problem. It will compile and run, but there is danger here if we don't use “`self.`”!

We can avoid this potential accident by having `@synthesize` use a different name for its instance variable than the name of the property. We do that using this equals-sign syntax.

```
@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;
```

Prefixing the property name with an underscore is the most common naming convention for an instance variable created by `@synthesize`.

Click here to see the error details.

Notice that there are errors now when we access the instance variable directly.

```
- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:nameTextField.text];
}

@end
```

Xcode has a suggestion to replace `nameTextField` with `_nameTextField` which will cause the error to disappear, but it will not solve the problem. **Only** setters and getters should access the instance variable directly! There are rare exceptions, but for now, stick to this rule. So we actually want to use the getter instead.

```
@implementation HelloWorldViewController

@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sende
{
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:_nameTextField.text];
}

@end
```

Issue: Use of undeclared identifier 'nameTextField'; did you mean '_nameTextField'?

Fix-it: Replace "nameTextField" with "_nameTextField"

The `nameTextField` instance variable is not recognized anymore.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

22. Fix the error by using the getter to access the `nameTextField` `@property`. A `@property` is nothing more than a setter method and a getter method. The setter method will be called by the system at run-time to wire this outlet up.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Mon 21:05 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Build Succeeded | 2/17/12 at 19:34 PM Project 1 Editor View Organizer

MainStoryboard.storyboard

MainStoryboar... › No Selection

type in your name

Say Hello

```
//
//  HelloWorldViewController.m
//  HelloWorld
//  Created by Radu-Tudor Ionescu on 2/9/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:self.nameTextField.text];
}

@end
```

Note that `@synthesize` also creates some storage. Our storage in this case is the `_nameTextField` instance variable.

Fix the error by putting the `self` in front of `nameTextField`. This is calling the getter method using dot notation.

It is perfectly legal to have multiple dots in an expression like this.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

23. Set the greeting message as the label's text using normal method notation.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Tue 18:54 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Build Succeeded | 2/17/12 at 19:34 PM Project 2 Editor View Organizer

MainStoryboard.storyboard

MainStoryboar... › No Selection

type in your name

Say Hello

Hello World View Controller

Automatic › HelloWorldViewController.m › -presentGreeting:

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyNameHere__.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:self.nameTextField.text];
    [self.greetingLabel setText:(NSString *)
}

@end
```

- void setTag:(NSInteger)
- void setText:(NSString *)
- void setTextAlignment:(UITextAlignment)
- void setTextColor:(UIColor *)
- void setTransform:(CGAffineTransform)

We should be using dot notation here, but we will (briefly) look at it using method notation just to be clear what we are doing.

We will use `UILabel`'s text `@property`'s setter (`setText:`) to set our `greetingLabel`'s text to the `greetingMessage`.

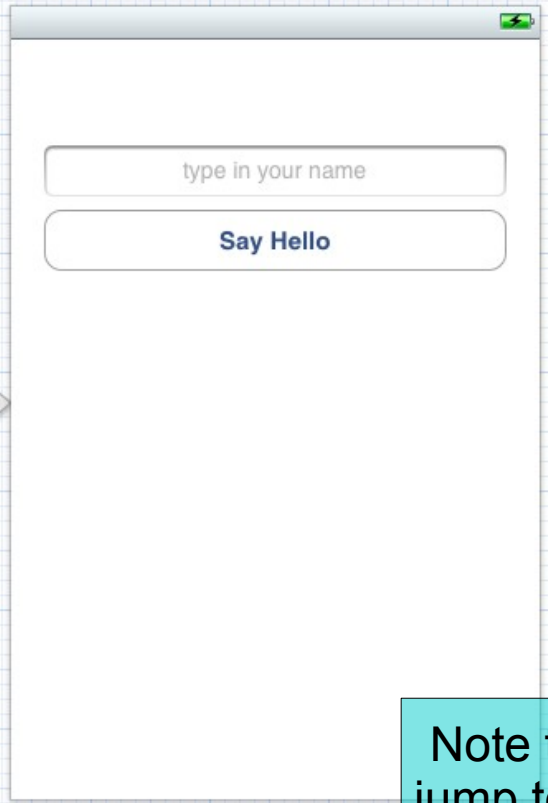
Xcode File Edit View Navigate Editor Product Window Help (Charged) Tue 19:05 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Build Succeeded | 2/17/12 at 19:34 PM Project 1 Editor View Organizer

MainStoryboard.storyboard

MainStoryboar... › No Selection



Automatic › HelloWorldViewController.m › -presentGreeting:

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:self.nameTextField.text];
    [self.greetingLabel setText:(NSString *)
}

@end
```

Note that the TAB key can be used to jump to the next argument to a method. This is what an argument looks like after having been tabbed to. Now you just type the argument you want (`greetingMessage` in this case) and it will replace the `(NSString *)`.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Tue 19:11 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints

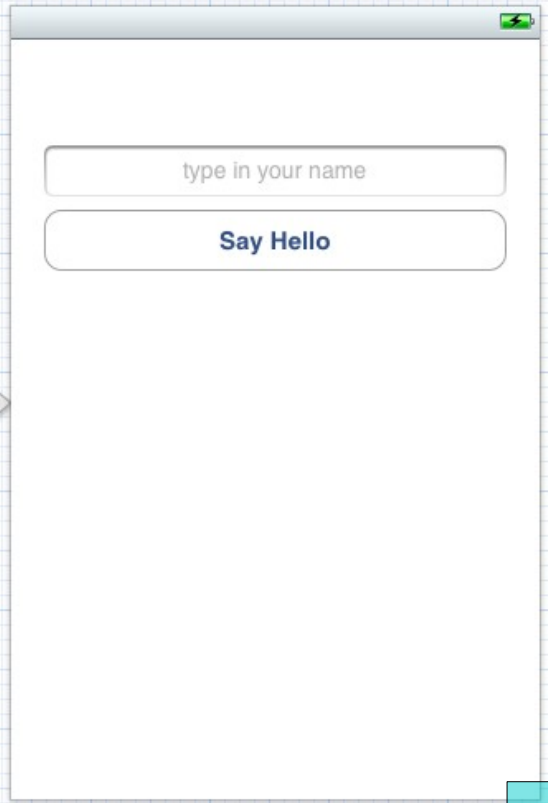
Build Succeeded | 2/17/12 at 19:34 PM

No Issues

Editor View Organizer

MainStoryboard.storyboard

MainStoryboard.storybo... › No Selection



Hello World View Controller

Automatic › HelloWorldViewController.m › -presentGreeting:

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:self.nameTextField.text];
    [self.greetingLabel setText:greetingMessage];
}

@end
```

Again, we would probably never use this normal method syntax to set a `@property` like this. It's just for illustrative purposes.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

24. Switch to using dot notation to set the `UILabel`'s text `@property`.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Tue 19:23 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints

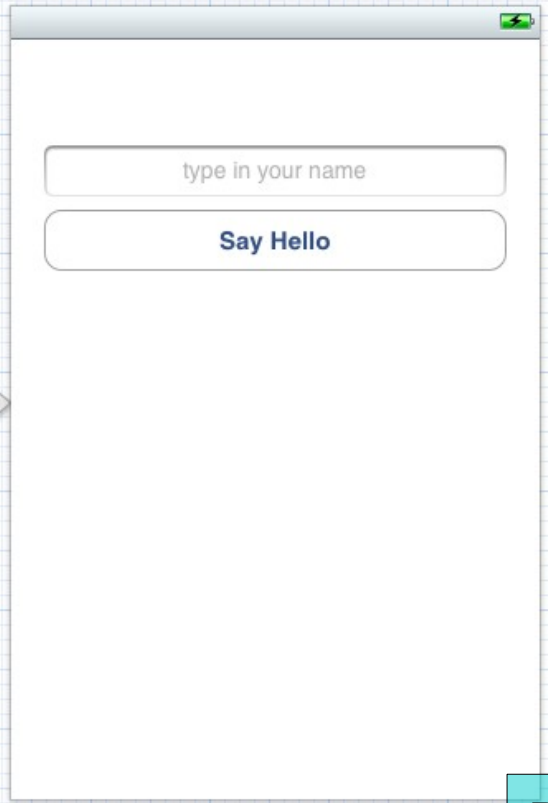
Build Succeeded | 2/17/12 at 19:34 PM

No Issues

Editor View Organizer

MainStoryboard.storyboard

MainStoryboard.storybo... › No Selection



Hello World View Controller

Automatic › HelloWorldViewController.m › -presentGreeting:

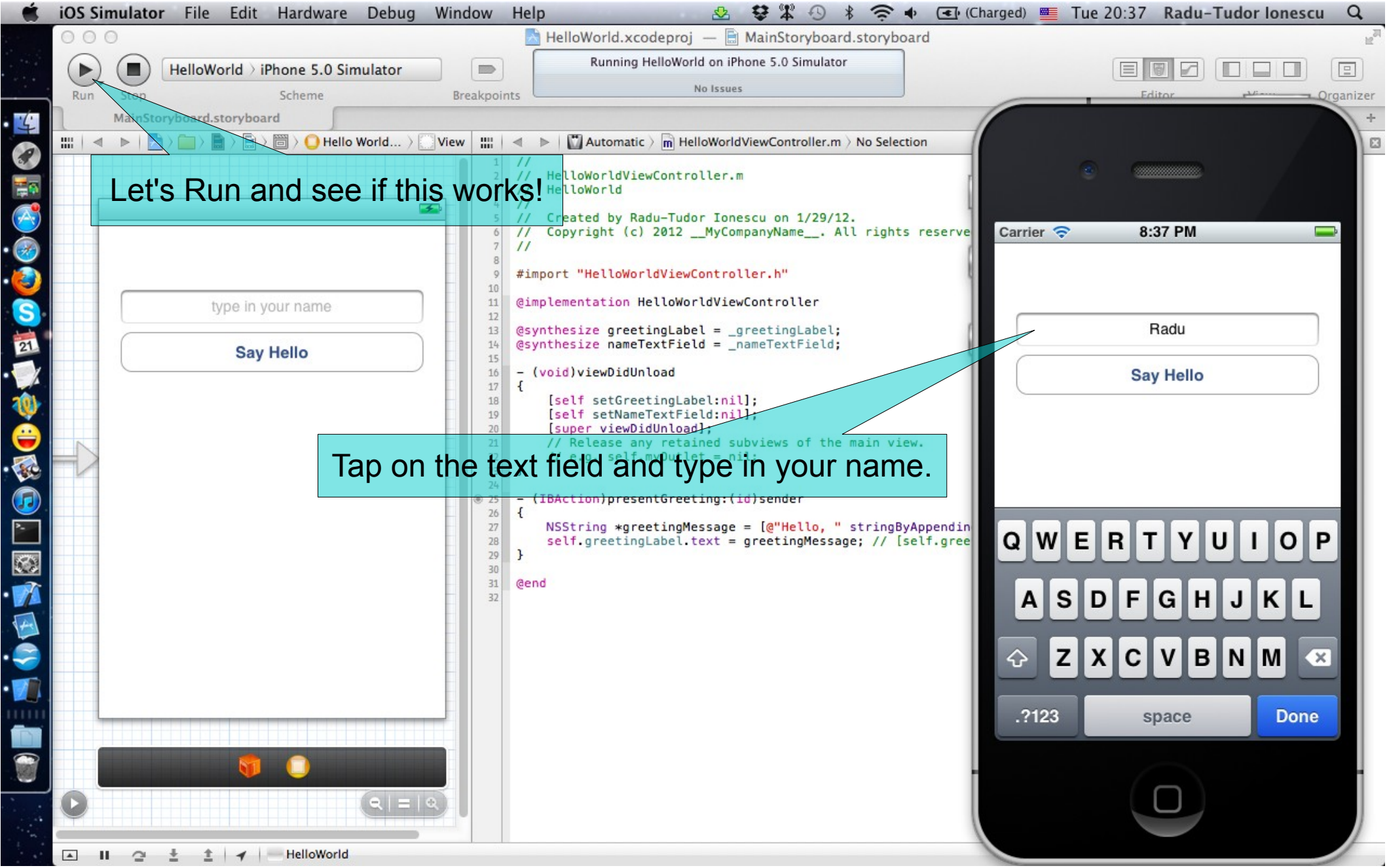
```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:self.nameTextField.text];
    self.greetingLabel.text = greetingMessage; // [self.greetingLabel setText:greetingMessage];
}

@end
```

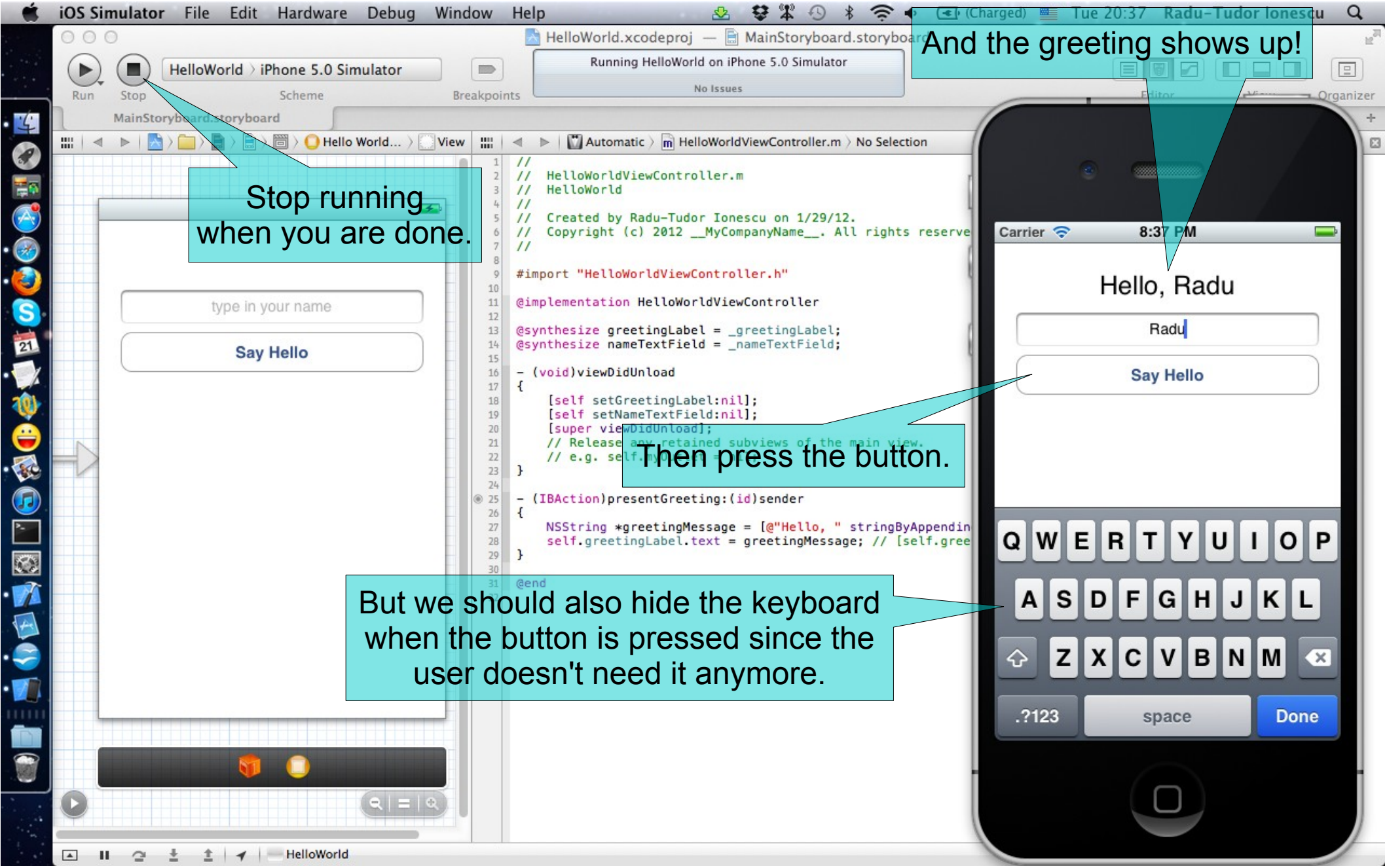
Dot notation for setters is exactly the same as dot notation for getters, it's just that they appear on the left-hand side of equals signs rather than the right-hand side.

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

25. Run the application in iOS Simulator to test it.


iOS Simulator File Edit Hardware Debug Window Help
(Charged) Tue 20:37 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Run Stop Scheme Breakpoints
Running HelloWorld on iPhone 5.0 Simulator
No Issues
Editor Organizer
MainStoryboard.storyboard
Hello World... View
Automatic HelloWorldViewController.m No Selection
Let's Run and see if this works!
type in your name
Say Hello
//
// HelloWorldViewController.m
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserve
//
#import "HelloWorldViewController.h"
@implementation HelloWorldViewController
@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;
- (void)viewDidUnload
{
[self setGreetingLabel:nil];
[self setNameTextField:nil];
[super viewDidUnload];
// Release any retained subviews of the main view.
- (IBAction)presentGreeting:(id)sender
{
NSString *greetingMessage = [@"Hello, " stringByAppendin
self.greetingLabel.text = greetingMessage; // [self.gree
}
@end
Tap on the text field and type in your name.
Carrier 8:37 PM
Radu
Say Hello
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
.?123 space Done
HelloWorld


iOS Simulator File Edit Hardware Debug Window Help
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld 〉 iPhone 5.0 Simulator
Run Stop Scheme Breakpoints
Running HelloWorld on iPhone 5.0 Simulator
No Issues
Editor Organizer
MainStoryboard.storyboard
Hello World... 〉 View
Automatic 〉 HelloWorldViewController.m 〉 No Selection
//
// HelloWorldViewController.m
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserve
//
#import "HelloWorldViewController.h"
@implementation HelloWorldViewController
@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;
- (void)viewDidUnload
{
[self setGreetingLabel:nil];
[self setNameTextField:nil];
[super viewDidUnload];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
}
- (IBAction)presentGreeting:(id)sender
{
NSString *greetingMessage = [@"Hello, " stringByAppendin
self.greetingLabel.text = greetingMessage; // [self.gree
}
@end
type in your name
Say Hello
Carrier 8:37 PM
Hello, Radu
Radu
Say Hello
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
.?123 space Done
HelloWorld
Stop running when you are done.
Then press the button.
But we should also hide the keyboard when the button is pressed since the user doesn't need it anymore.
And the greeting shows up!

Task 6

Task: Implement the button action so that our application displays the greeting message on the screen.

25. Dismiss the keyboard when the button is pressed. When the user taps in a text field, that text field becomes the first responder and automatically asks the system to display the associated keyboard. It is your application's responsibility to dismiss the keyboard at the time of your choosing.

In general, you might dismiss the keyboard in response to a specific user action, such as the user tapping a particular button in your user interface. You might also configure your text field delegate to dismiss the keyboard when the user presses the "return" key on the keyboard itself. To dismiss the keyboard, you must send the `resignFirstResponder` message to the text field that is currently the first responder. Doing so causes the text field object to end the current editing session and hide the keyboard.

26. Test if the keyboard hides when the button is pressed.

Xcode File Edit View Navigate Editor Product Window Help (Charged) Tue 20:41 Radu-Tudor Ionescu

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld › iPhone 5.0 Simulator Scheme Breakpoints Finished running HelloWorld on iPhone 5.0 Simulato No Issues Editor View Organizer

MainStoryboard.storyboard

Hello World... › View

type in your name

Say Hello

Automatic › HelloWorldViewController.m › -presentGreeting: 2

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *greetingMessage = [@"Hello, " stringByAppendingString:self.nameTextField.text];
    self.greetingLabel.text = greetingMessage; // [self.greetingLabel setText:greetingMessage];
    [self.nameTextField resignFirstResponder];
}

@end
```

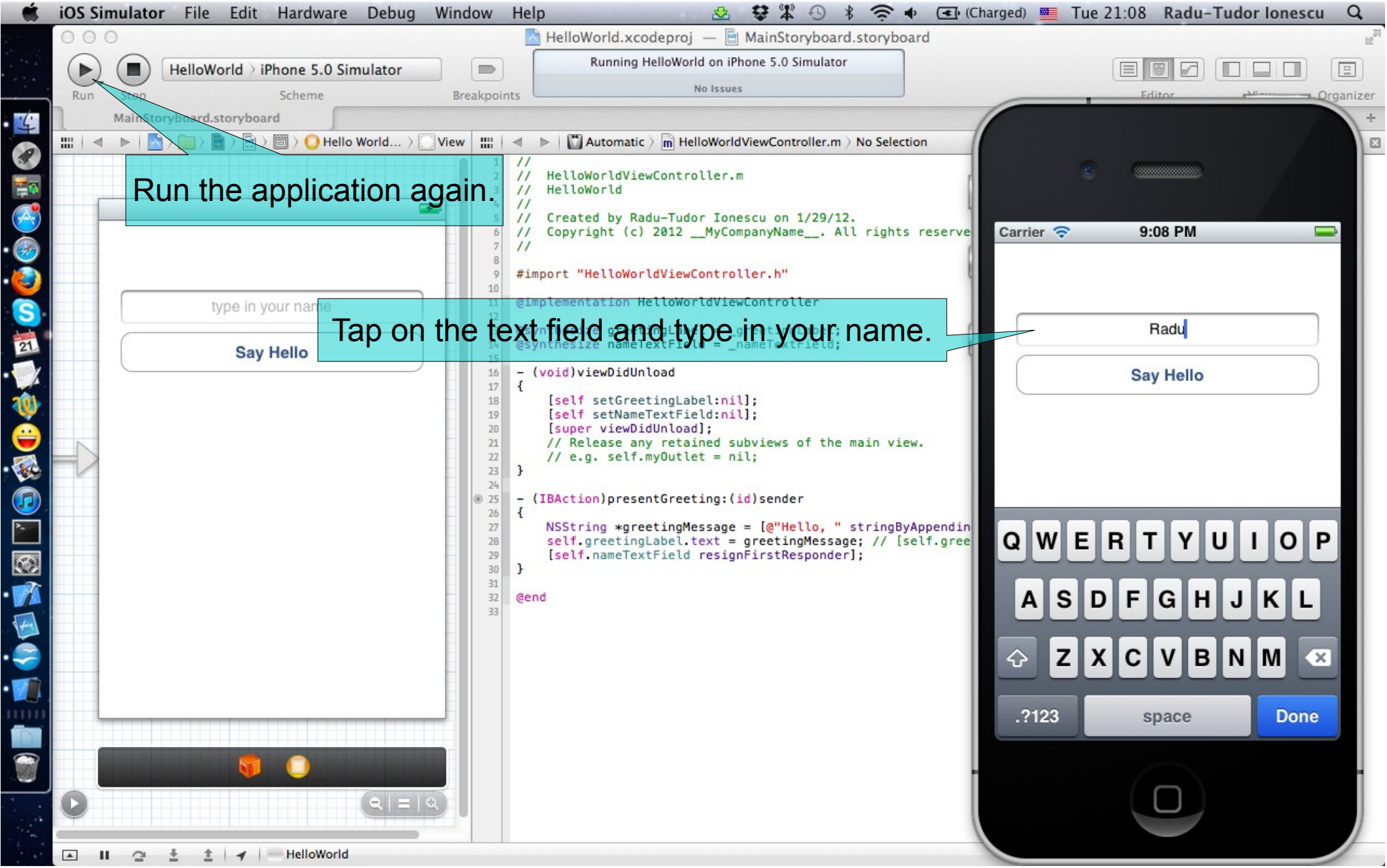
Send the `resignFirstResponder` message to the text field.

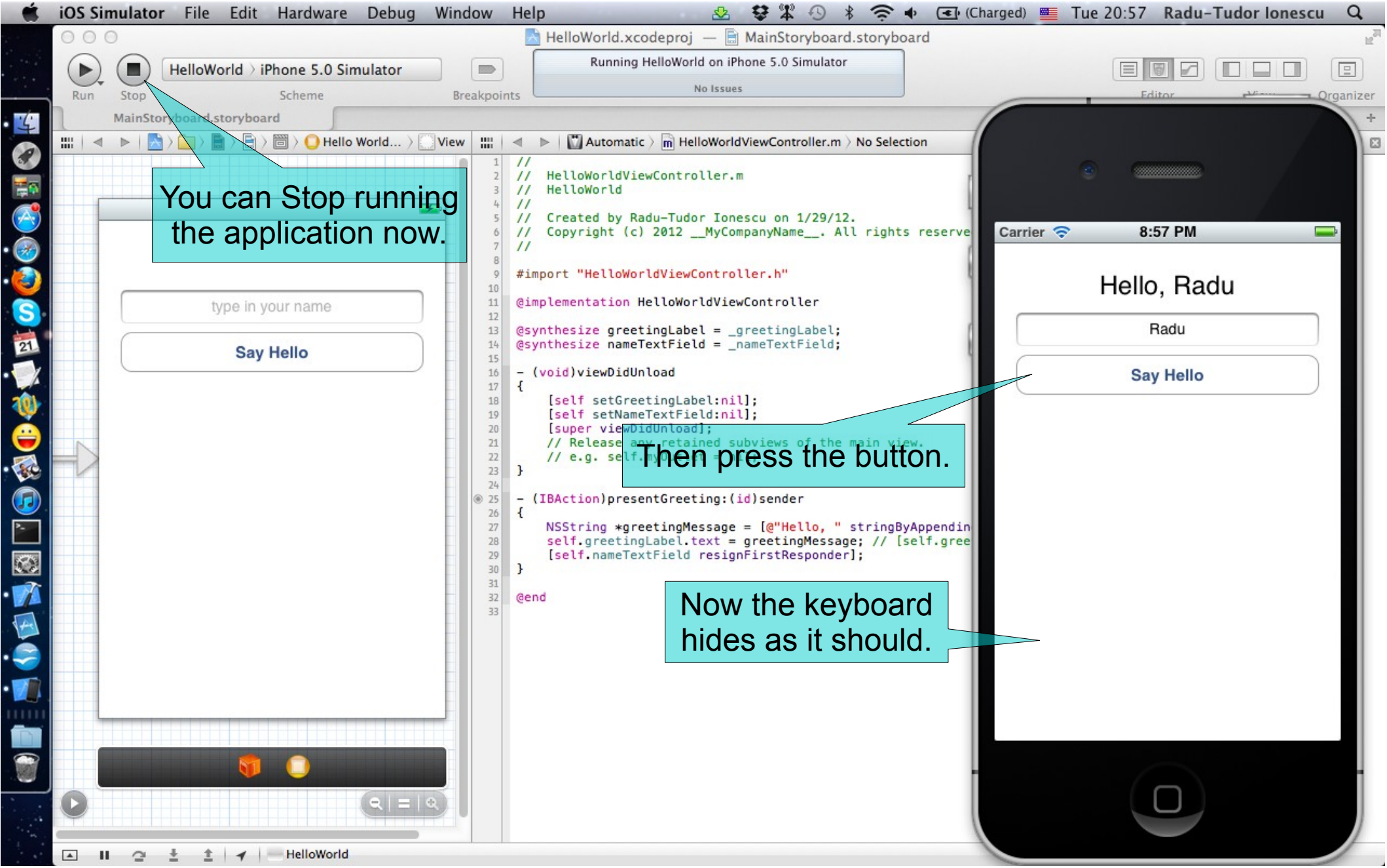
Notice the method has no parameters.

resignFirstResponder

Name: resignFirstResponder
Declaration: - (BOOL)resignFirstResponder
Availability: iOS (2.0 and later)
Abstract: Notifies the receiver that it has been asked to relinquish its status as first responder in its window.
Declared In: UIResponder.h
Reference: UIResponder Class Reference
Related Documents: Event Handling Guide for iOS

Hold down the option key and click on the method to get details.


iOS Simulator File Edit Hardware Debug Window Help
Tue 21:08 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
HelloWorld iPhone 5.0 Simulator
Run
Stop
Scheme
Breakpoints
Running HelloWorld on iPhone 5.0 Simulator
No Issues
Editor
Organizer
MainStoryboard.storyboard
Hello World... View
Automatic HelloWorldViewController.m No Selection
Run the application again.
type in your name
Say Hello
Tap on the text field and type in your name.
//
// HelloWorldViewController.m
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserve
//
#import "HelloWorldViewController.h"
@implementation HelloWorldViewController
@synthesize nameTextField = _nameTextField;
- (void)viewDidUnload
{
[self setGreetingLabel:nil];
[self setNameTextField:nil];
[super viewDidUnload];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
}
- (IBAction)presentGreeting:(id)sender
{
NSString *greetingMessage = [@"Hello, " stringByAppendin
self.greetingLabel.text = greetingMessage; // [self.gree
[self.nameTextField resignFirstResponder];
}
@end
Carrier 9:08 PM
Radu
Say Hello
Q W E R T Y U I O P
A S D F G H J K L
Z X C V B N M
.?123 space Done
HelloWorld


iOS Simulator File Edit Hardware Debug Window Help
Tue 20:57 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Run Stop
HelloWorld 〉 iPhone 5.0 Simulator
Scheme
Breakpoints
Running HelloWorld on iPhone 5.0 Simulator
No Issues
Editor
Organizer
MainStoryboard.storyboard
Hello World... 〉 View
Automatic 〉 HelloWorldViewController.m 〉 No Selection
You can Stop running the application now.
type in your name
Say Hello
//
// HelloWorldViewController.m
// HelloWorld
//
// Created by Radu-Tudor Ionescu on 1/29/12.
// Copyright (c) 2012 __MyCompanyName__. All rights reserve
//
#import "HelloWorldViewController.h"
@implementation HelloWorldViewController
@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;
- (void)viewDidUnload
{
[self setGreetingLabel:nil];
[self setNameTextField:nil];
[super viewDidUnload];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
}
- (IBAction)presentGreeting:(id)sender
{
NSString *greetingMessage = [@"Hello, " stringByAppendin
self.greetingLabel.text = greetingMessage; // [self.gree
[self.nameTextField resignFirstResponder];
}
@end
Then press the button.
Now the keyboard hides as it should.
Carrier 8:57 PM
Hello, Radu
Radu
Say Hello
HelloWorld

Task 7

Task: Find out if the application has any problems that should be fixed.

1. Notice we can't dismiss (hide back) the keyboard when the return key (“Done”) is pressed. The normal behavior is to dismiss the keyboard when the return button is pressed.
2. The greeting message would look nice with an exclamation mark at the end. At this moment the application doesn't look finished.

It would be nice to have an exclamation mark at the end.

If you press the return button the keyboard should hide. This doesn't happen!

iOS Simulator File Edit Hardware Debug Window Help

HelloWorld.xcodeproj — MainStoryboard.storyboard

Run Stop HelloWorld 〉 iPhone 5.0 Simulator Scheme Breakpoints

Running HelloWorld on iPhone 5.0 Simulator

No Issues

MainStoryboard.storyboard

Hello World... 〉 View

type in your name

Say Hello

Automatic 〉 HelloWorldViewController.m 〉 No Selection

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by Radu-Tudor Ionescu on 1/29/12.
//  Copyright (c) 2012 __MyCompanyName__. All rights reserve
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;

- (void)viewDidUnload
{
    [self setGreetingLabel:nil];
    [self setNameTextField:nil];
    [super viewDidUnload];
    // Release any retained subviews of the main view.
    // e.g. self.myOutlet = nil;
}

- (IBAction)presentGreeting:(id)sender
{
    NSString *greetingMessage = [@"Hello, " stringByAppendin
    self.greetingLabel.text = greetingMessage; // [self.gree
    [self.nameTextField resignFirstResponder];
}

@end
```

HelloWorld

Carrier 9:20 PM

Hello, Radu

Radu

Say Hello

Q W E R T Y U I O P

A S D F G H J K L

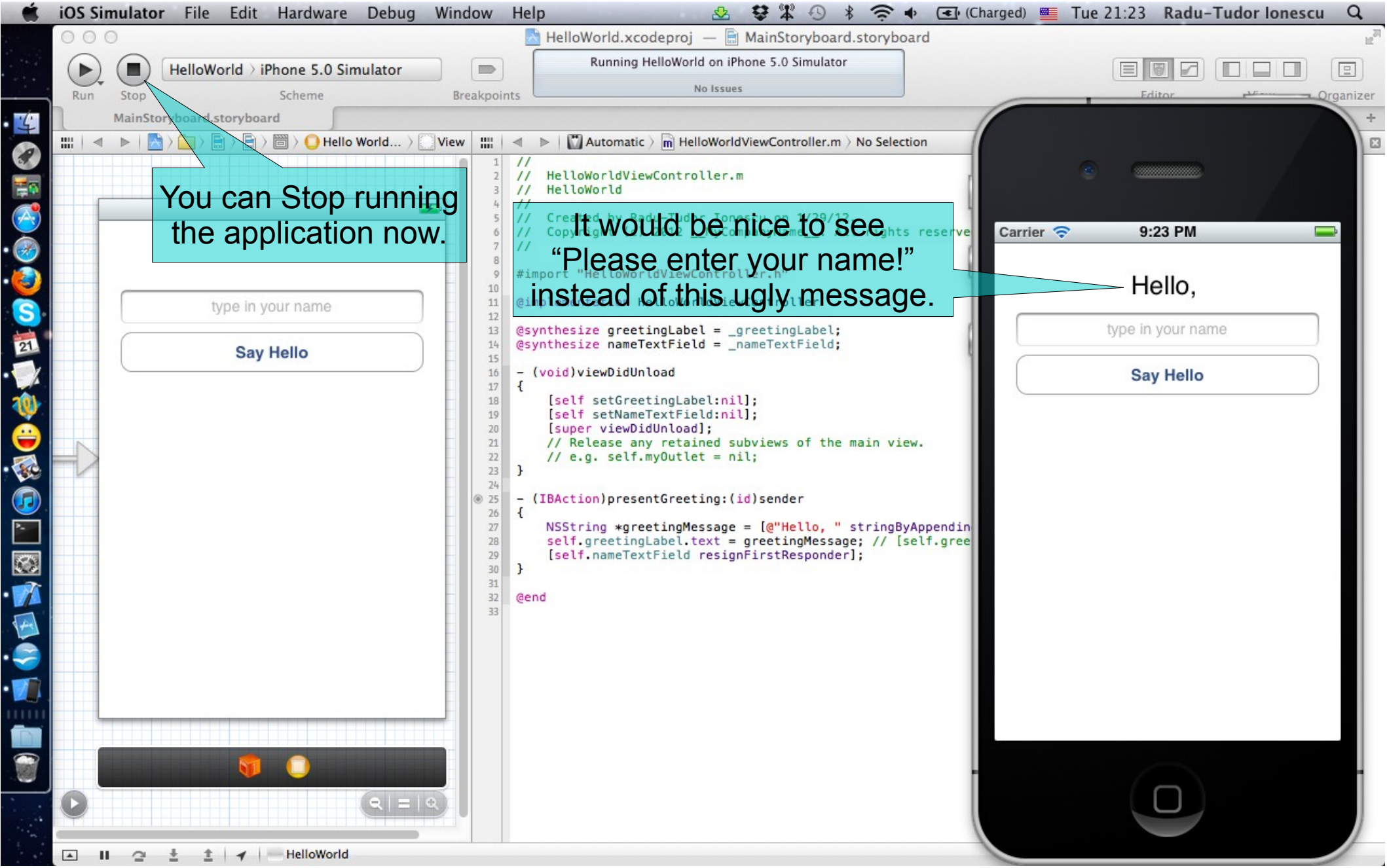
Z X C V B N M

.?123 space Done

Task 7

Task: Find out if the application has any problems that should be fixed.

3. If the user chooses not to type in his name, the application will display an ugly message “Hello, “. In this case we should display a friendly message (“Please enter your name!”) to ask the user for his name.


iOS Simulator File Edit Hardware Debug Window Help
Tue 21:23 Radu-Tudor Ionescu
HelloWorld.xcodeproj — MainStoryboard.storyboard
Run Stop
HelloWorld 〉 iPhone 5.0 Simulator
Scheme
Breakpoints
Running HelloWorld on iPhone 5.0 Simulator
No Issues
Editor
Organizer
MainStoryboard.storyboard
Hello World... 〉 View
Automatic 〉 HelloWorldViewController.m 〉 No Selection
You can Stop running the application now.
It would be nice to see "Please enter your name!" instead of this ugly message.
type in your name
Say Hello
//
// HelloWorldViewController.m
// HelloWorld
//
#import "HelloWorldViewController.h"
@synthesize greetingLabel = _greetingLabel;
@synthesize nameTextField = _nameTextField;
- (void)viewDidUnload
{
[self setGreetingLabel:nil];
[self setNameTextField:nil];
[super viewDidUnload];
// Release any retained subviews of the main view.
// e.g. self.myOutlet = nil;
}
- (IBAction)presentGreeting:(id)sender
{
NSString *greetingMessage = [@"Hello, " stringByAppendin
self.greetingLabel.text = greetingMessage; // [self.gree
[self.nameTextField resignFirstResponder];
}
@end
Carrier
9:23 PM
Hello,
type in your name
Say Hello
HelloWorld

Assignment 1

Assignment: Change the application behavior to hide the keyboard when the return button is pressed.

Hint: Search for the “Did End On Exit” event of the `nameTextField` (right click on the text field in Interface Builder). Set the `presentGreeting:` action to this event. Note that you can set the same action for two or more events (generated by different UI elements).

Assignment 2

Assignment: Add the exclamation mark at the end of the greeting message.

Hint: There are at least two possible solutions. You can either use the `stringByAppendingString:` method twice or you can use the `NSString's stringWithFormat:` class method. Check the `NSString` documentation and look for `stringWithFormat:`.

Here is an example of how to use it:

```
NSString *aString = [NSString stringWithFormat:
                     @"A string-%@. A float-%.2f",
                     @"abc", 3.14159265];
// aString is "A string-abc. A float: 3.14"
```

Assignment 3

Assignment: Check if the user name is blank and put the “Please enter your name!” message in this case.

Hint: Use the `isEqualToString:` method (for `NSString` objects) to test if the `nameTextField`'s text is equal to `@""` (a blank string). Look up this method in the `NSString` class documentation. Set the label's text to `@"Please enter your name!"` if the name is blank.

Congratulations!