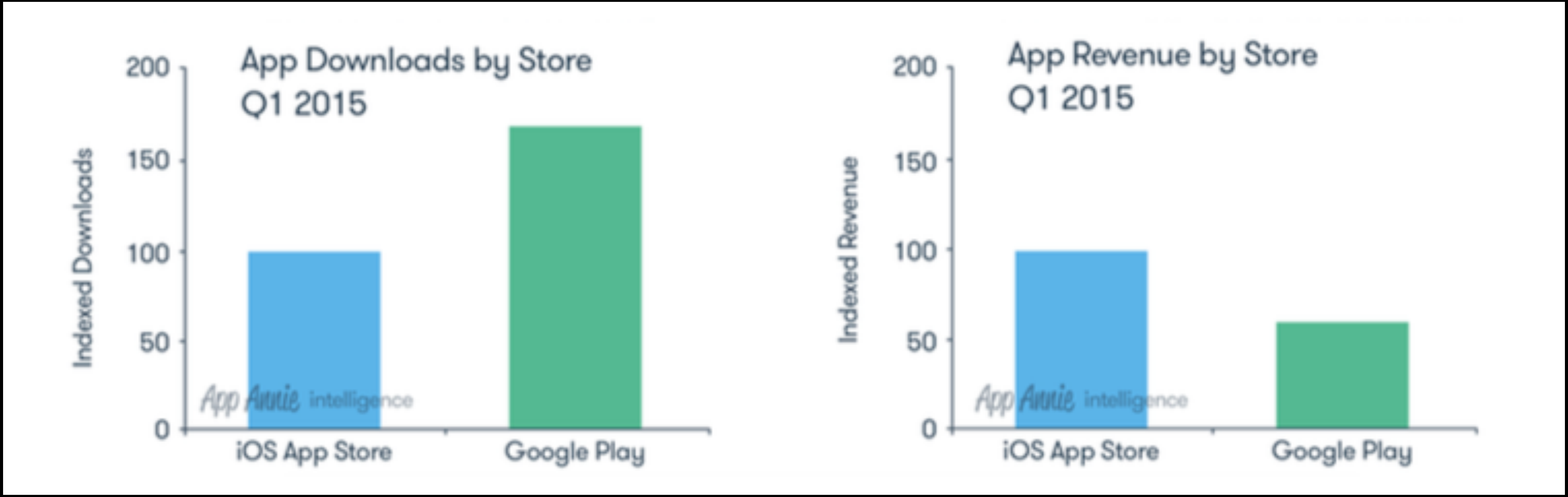
Day 2

1. iOS
2. Cocoa Touch
3. Xcode and Interface Builder
4. The app lifecycle
5. Object Lifecycle (objects, views, view controllers)
6. Storyboards and Nibs
7. Gesture Recognizers
8. Segues
9. Views
10. Common patterns (mvc, target-action, KVO, notifications)
11. Creating a Custom View
12. Coding Conventions

Homework

Something Interesting



Design is **Important**

Quality !!!!!!

Understood patterns!

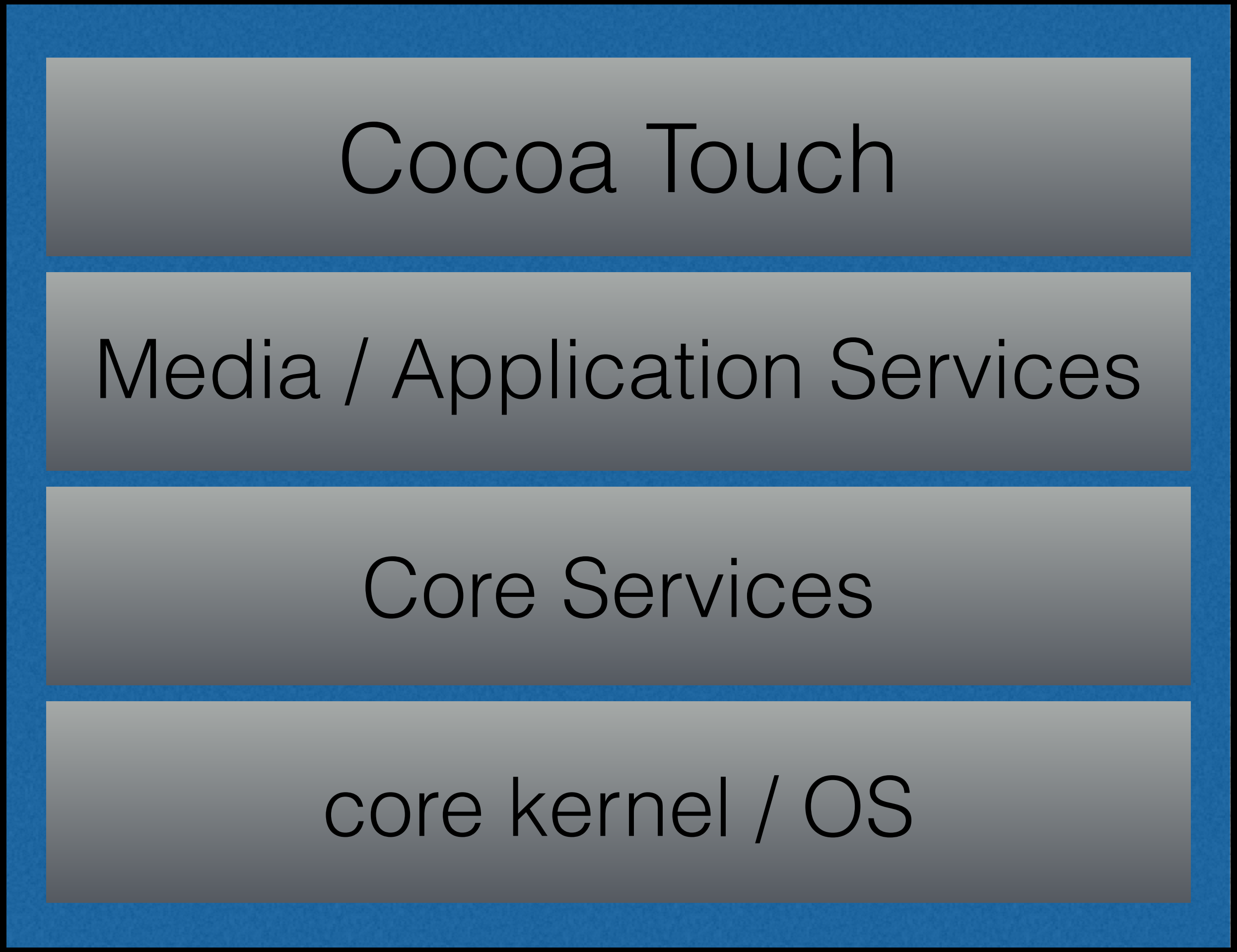
Use apple apps as a guide

https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/

iOS

- Unveiled in 2007 with the first iPhone
- Apples mobile operating system
- Based on direct manipulation and multi touch gestures
- Uses Cocoa Touch instead of Cocoa
- UIKIt instead of AppleKit (lightweight version)
- iOS is a darwin OS but not quite Unix compatible
- It is a highly sandboxed environment

Cocoa Touch


Cocoa Touch
Media / Application Services
Core Services
core kernel / OS

Xcode

- Apples IDE for all development for Apple devices
- Code editing, user interface design, asset management, testing, and debugging
- Built from many different tools. App loader, interface builder, file merge
- Provide all the command line tools needed for developing apps.
- Attempts to remove the hassle in provisioning

App LifeCycle

Figure 2-2 Processing events in the main run loop

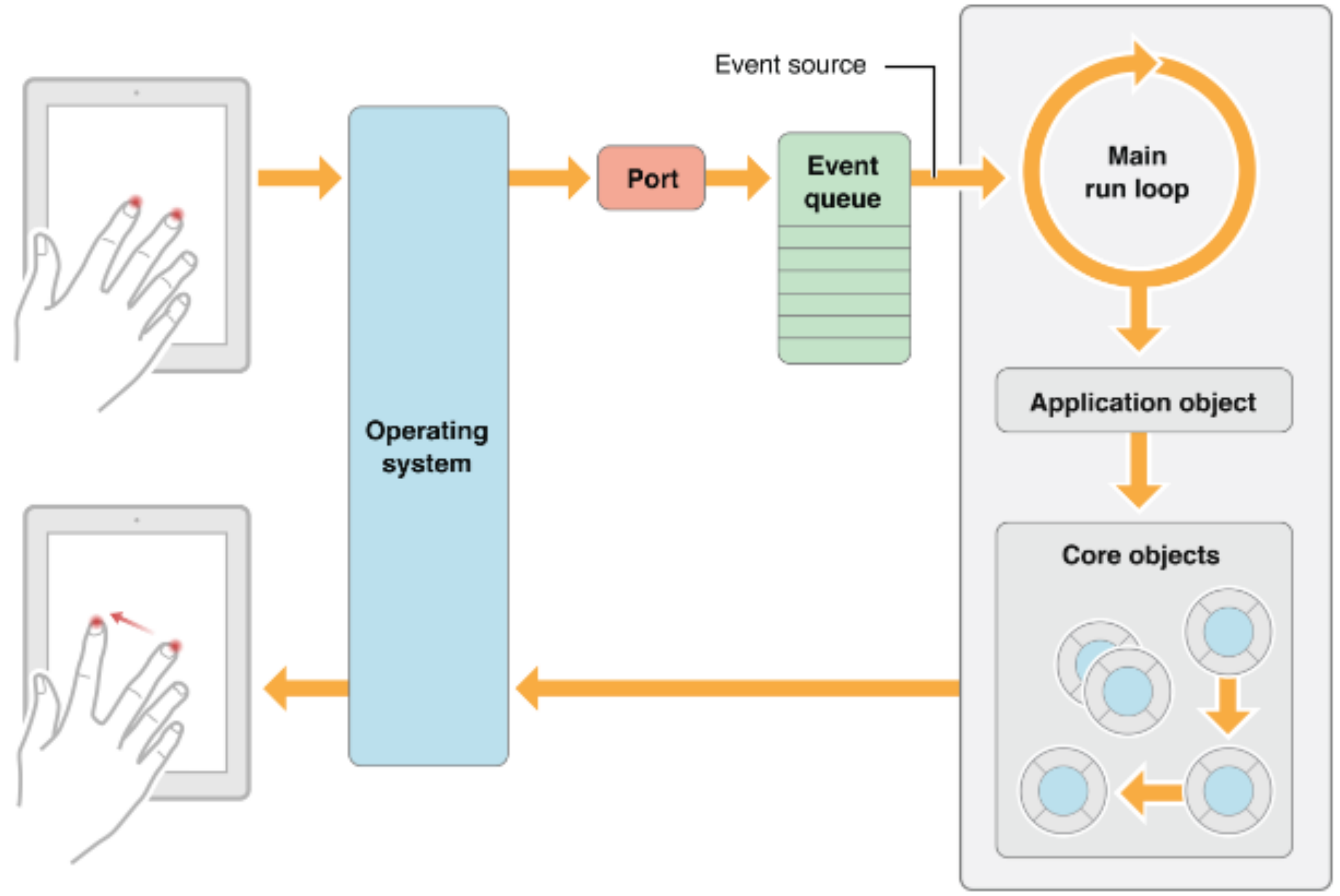


Figure 2-3 State changes in an iOS app

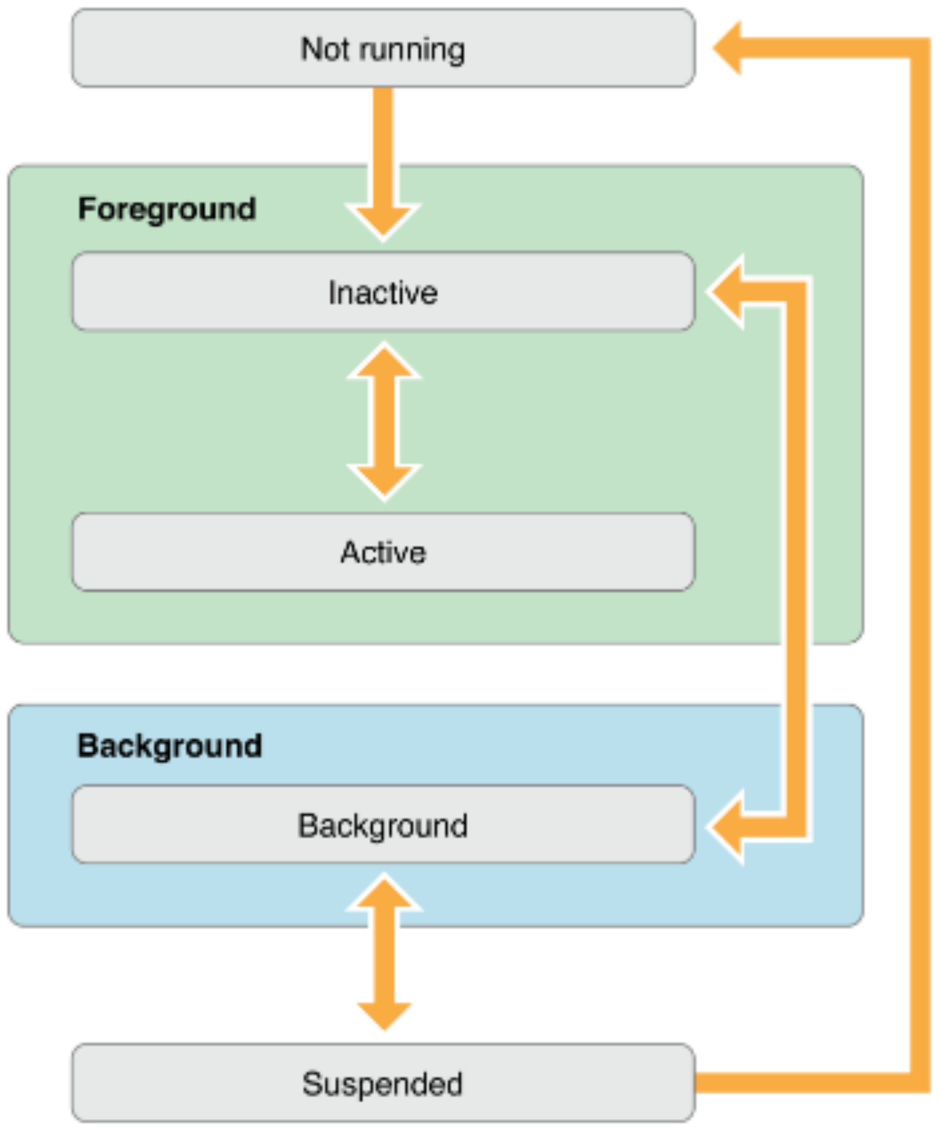


Figure 4-1 Launching an app into the foreground

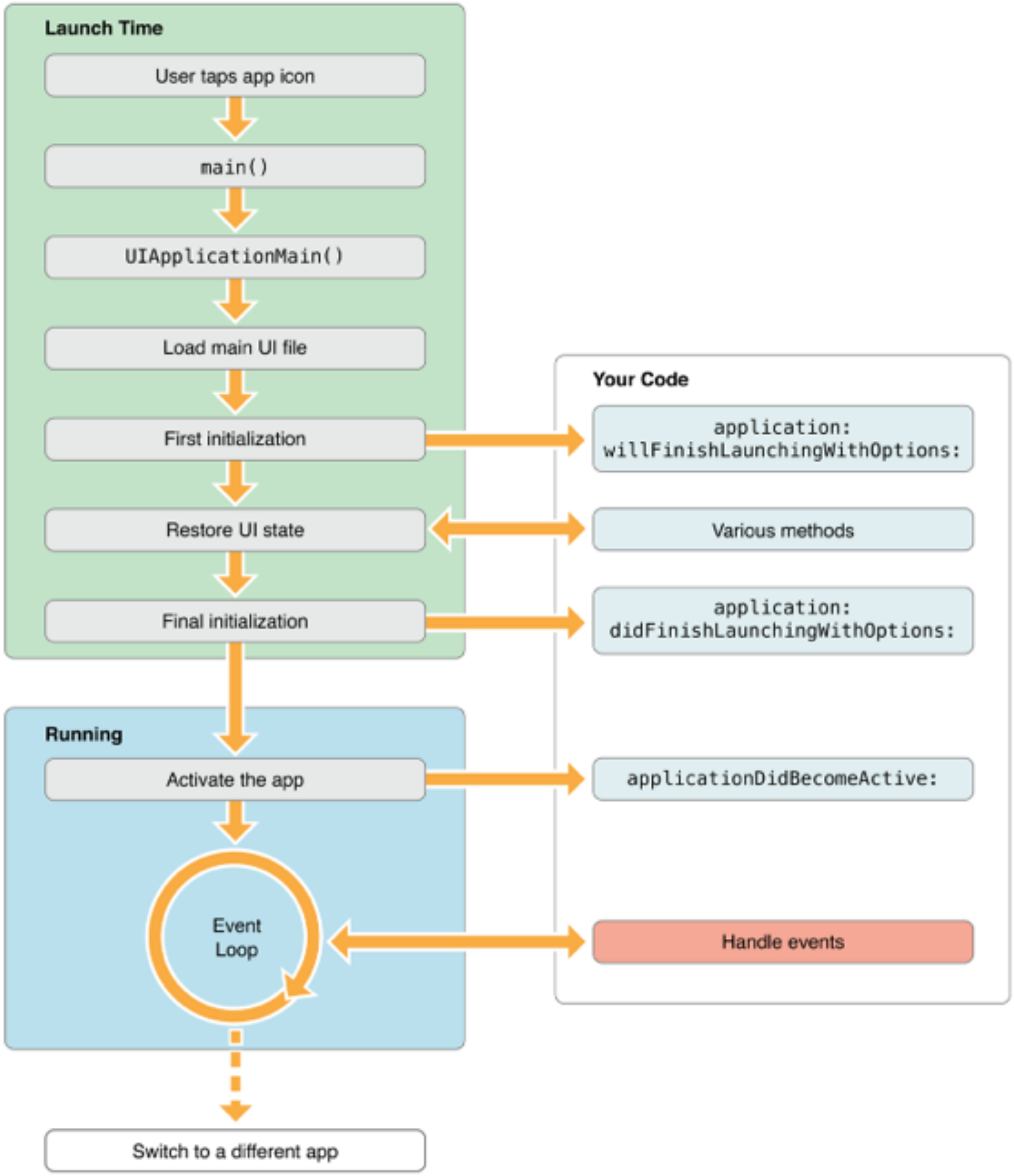
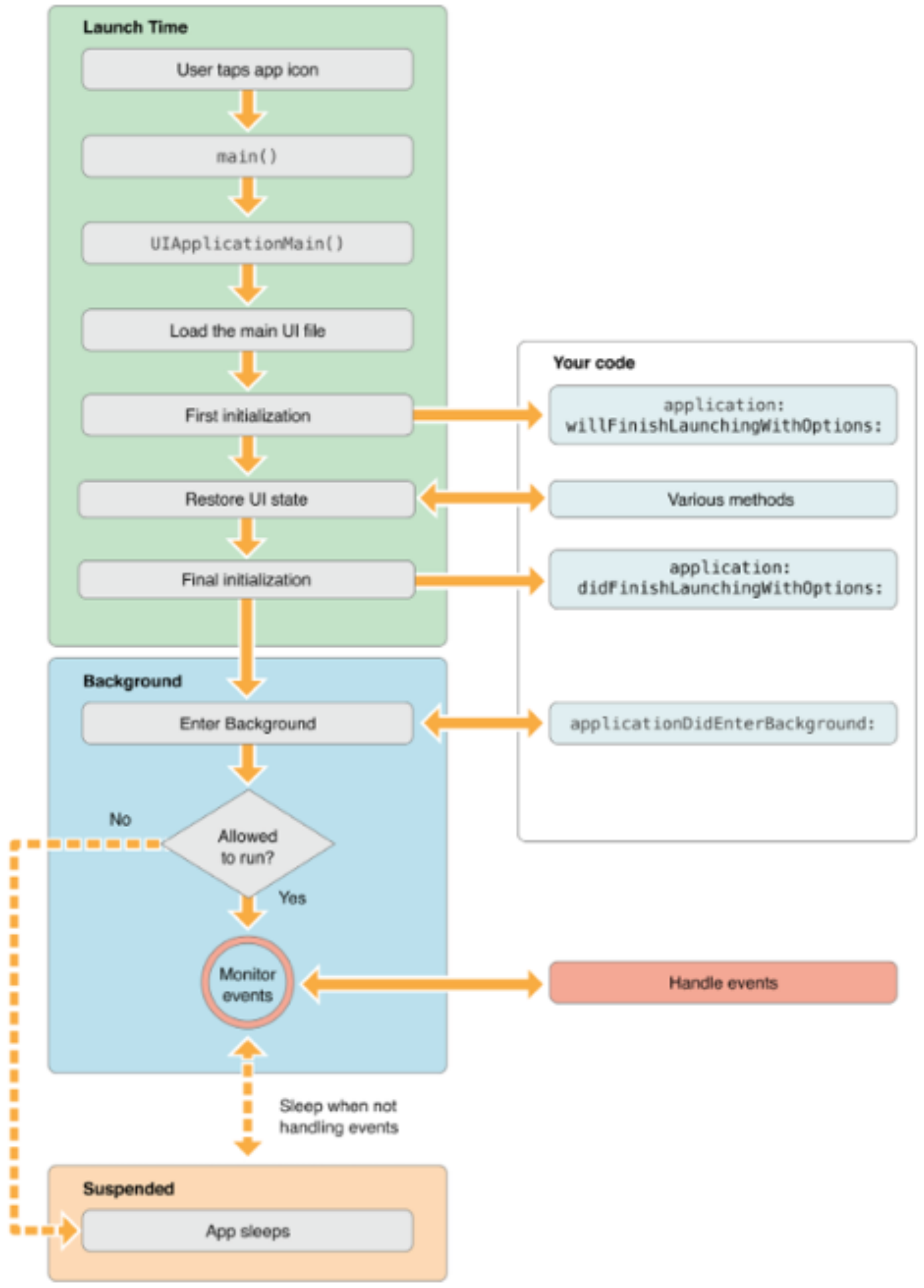


Figure 4-2 Launching an app into the background



IBOutlets and IBActions

IBOutlets form the link between the xib file and the code property

```
@interface ELDViewController : UIViewController

@property  nonatomic  weak  IBOutlet UITextField  nameEntryField;
@property  nonatomic  weak  IBOutlet UILabel  nameLabel;

@end
```

```
@implementation ELDViewController
    {
    IBOutlet __weak UITextField  _nameEntryField;
    IBOutlet __weak UILabel  _nameLabel;
    }
```

IBActions form the link between the xib file and the code action.
They encapsulate a target and an action

```
- (IBAction)onNameTextChange:(id)sender
   {
   _nameLabel text = _nameEntryField text;
   }


- (IBAction)onSaveButtonTap:(id)sender
   {
   _name = _nameEntryField text;
   }
```

Segues

- Visual transition between two controllers
- Several built in Segues
- Can create custom segues (UIStoryBoardSegue)

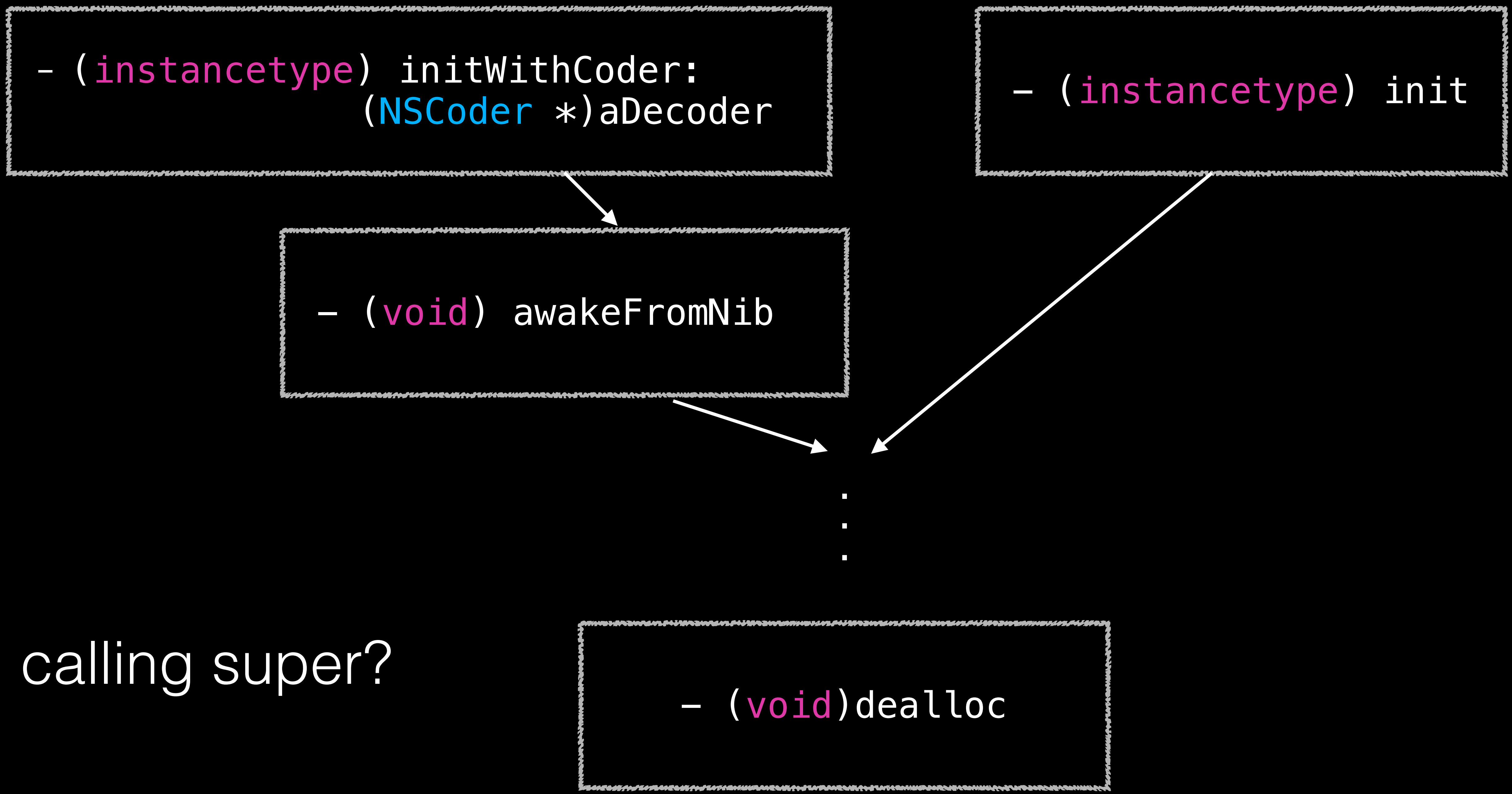
```
[self performSegueWithIdentifier:@"FromUserSetupToGreetingPage" sender self];

  - (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender
      {
      UIViewController* destinationController;

      if ([segue.identifier isEqual:@"FromUserSetupToGreetingPage"])
          {
          destinationController = segue.destinationViewController;
          }
      }
```

Lifecycle

NSObject



```
- (instancetype) initWithCoder:
                 (NSCoder *)aDecoder
```

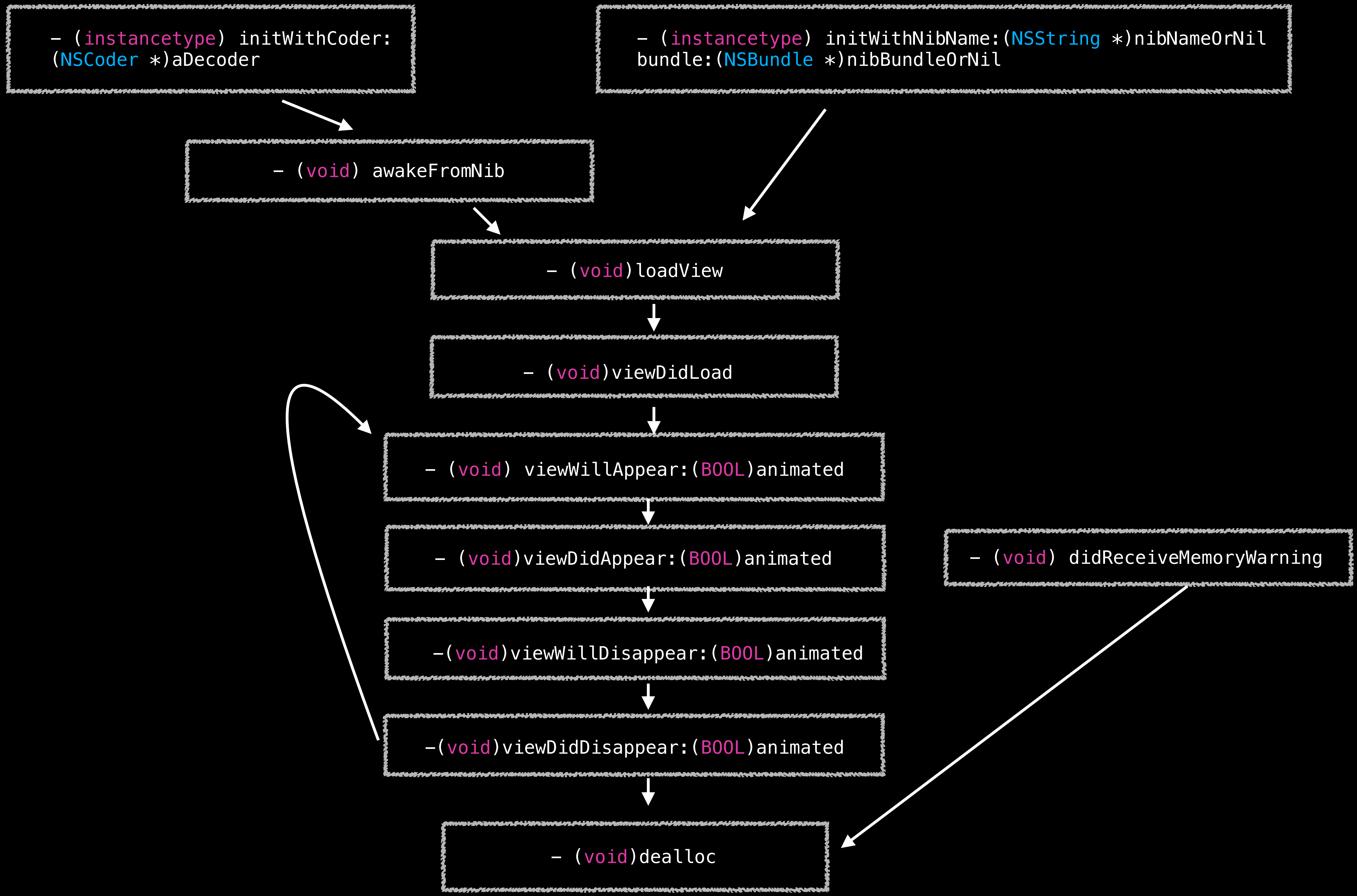
```
- (instancetype) init
```

```
- (void) awakeFromNib
```

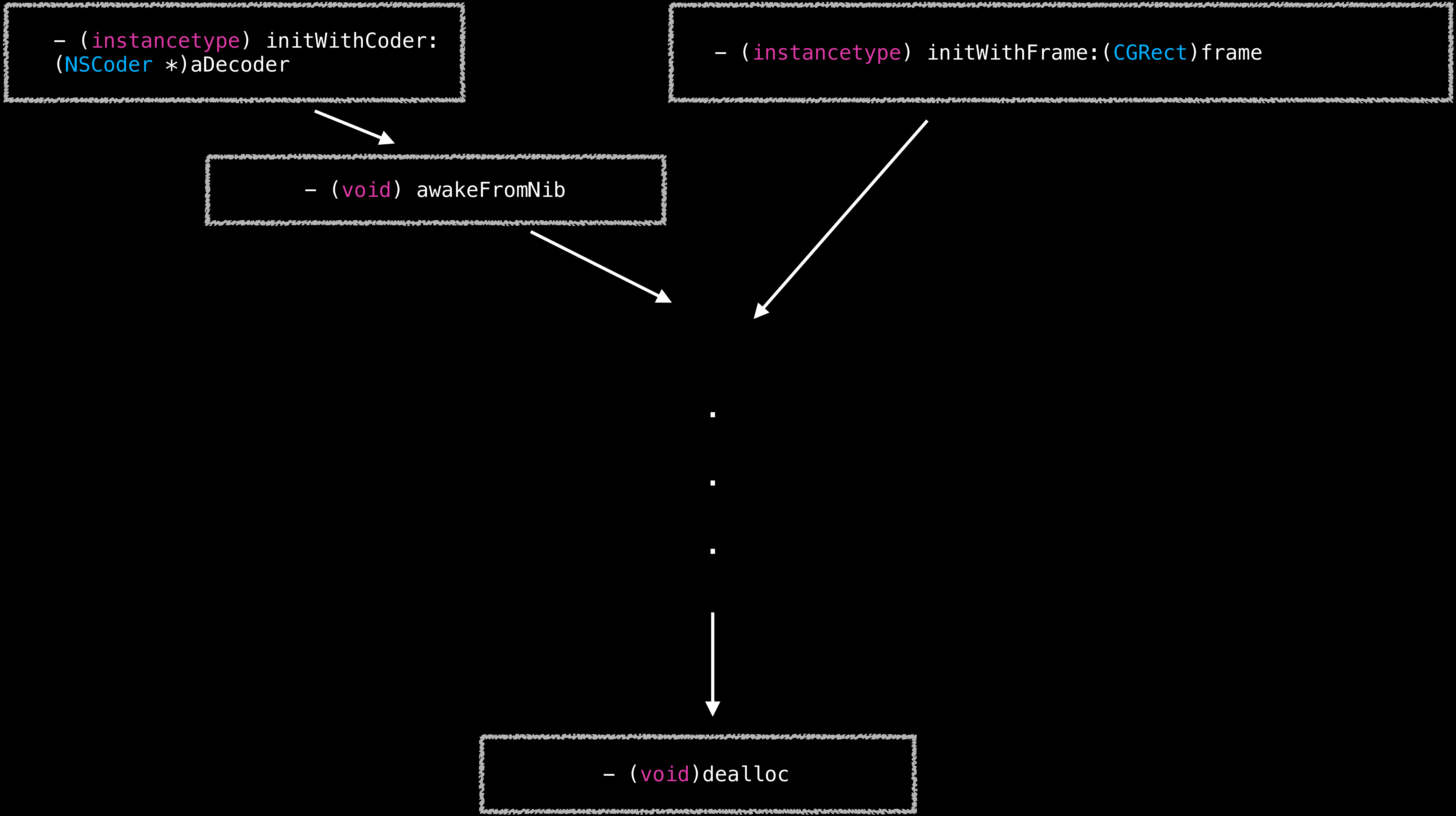
```
- (void)dealloc
```

calling super?

UIViewController


- (instancetype) initWithCoder:(NSCoder *)aDecoder
- (instancetype) initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
- (void) awakeFromNib
- (void)loadView
- (void)viewDidLoad
- (void) viewWillAppear:(BOOL)animated
- (void)viewDidAppear:(BOOL)animated
- (void) didReceiveMemoryWarning
-(void)viewWillDisappear:(BOOL)animated
-(void)viewDidDisappear:(BOOL)animated
- (void)dealloc

UIView



```
- (instancetype) initWithCoder:
(NSCoder *)aDecoder
```

```
- (instancetype) initWithFrame:(CGRect)frame
```

```
- (void) awakeFromNib
```

.

.

.

```
- (void)dealloc
```

Exercise

Add another label that displays the persons age

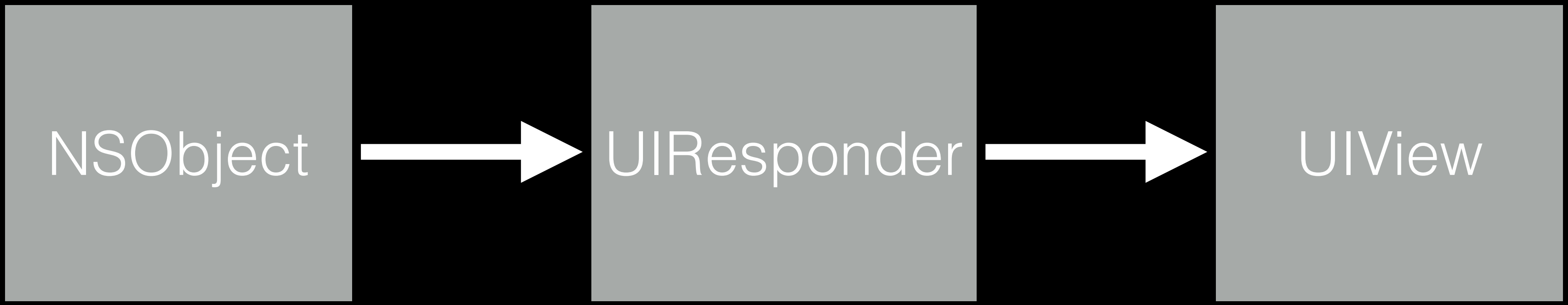
Add a switch that makes a label either show or hide

Views

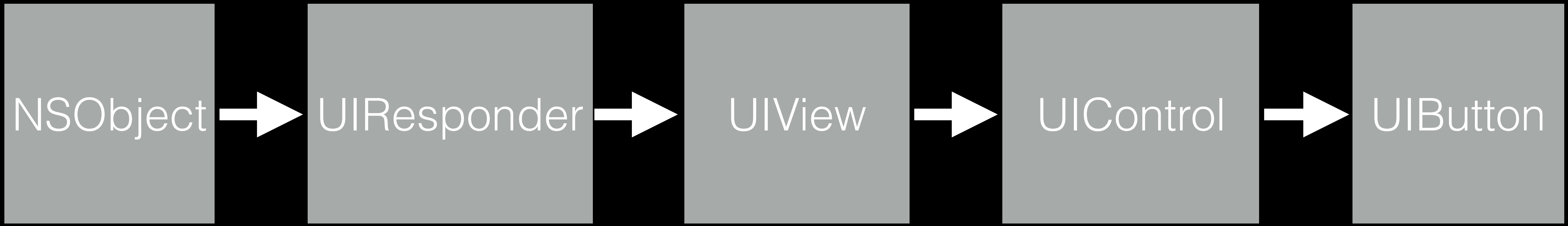
- Views allow you to represent visual content on a screen
- On iOS there is one window (subclass of UIView) that acts as a container
- Views are responsible for drawing content, handling multitouch events, and managing the layout of any subviews
- A view responds to touch events in its rectangular area either by using gesture recognizers or by handling touch events directly
- Views can have a parent child relationship (subview, superview)

https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIView_Class/index.html

UIView



The base class of all UI Objects



Gesture Recognizers

- Convert low level enter into high level action
- Attached to a view
- Convert the view into a control

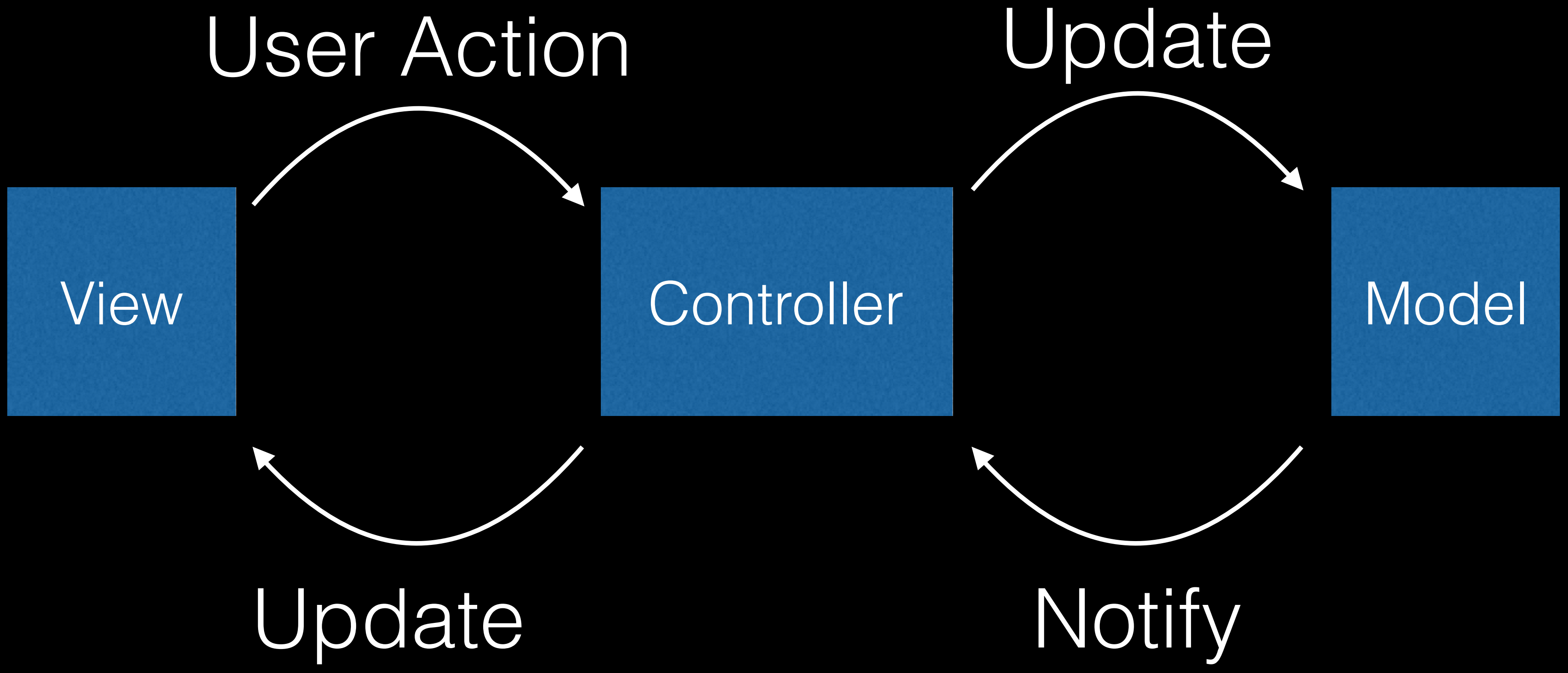
```
UITapGestureRecognizer* tapRecognizer;

  tapRecognizer = [[UITapGestureRecognizer alloc] initWithTarget self action @selector(onViewTap:)];
[self addGestureRecognizer:tapRecognizer];
```

iOS Design Patterns

- Model View Controller (MVC)
- Target - Action
- Delegation and Data sources
- Key Value Observing
- Broadcasting

Model-View-Controller (MVC)



Target - Action

UIControl

```
@property (assign) SEL action
@property (nonatomic, weak) id target
```

```
- (void)addTarget:(id)target action:(SEL)action
   forControlEvents(UIControlEvents)controlEvents

-  (void)removeTarget:(id)target action:(SEL)action
    forControlEvents:(UIControlEvents)controlEvents
```

in use

```
{
UIButton* button;
button.target = self;
button.action =@selector(onAddTap:);
}
```

```
-   (IBAction) onAddTap: (id)sender
  {
  //do stuff
  }
```

Delegation and Data Sources

Both are implemented through protocols. In either case either the delegate or the data source adheres to a specific protocol. Often the delegate and data source of an UI object are the same object.

Delegate:

Is for event driven behavior

Data Source:

Is for control of data

Both are just objects that adhere to a protocol and have been typedef'd

```
@protocol UITableViewDelegate

- (void)tableView:(UITableView *)tableView
didSelectRowAtIndexPath:(NSIndexPath *)indexPath;
..
..

@end

@protocol UITableViewDataSource

- (NSInteger)tableView:(UITableView *)tableView
 numberOfRowsInSection:(NSInteger)section..
..

@end
```

Key Value Observing

```
-(void)addObserver:(NSObject *)anObserver
          forKeyPath:(NSString *)keyPath
             options:(NSKeyValueObservingOptions)options
             context:(void *)context


 – (void)removeObserver:(NSObject *)anObserver
             forKeyPath:(NSString *)keyPath


– (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary *)change
                       context:(void *)context
```

Broadcasting - NSNotificationCenter

```
- (void)addObserver:(id)notificationObserver
           selector:(SEL)notificationSelector
               name:(NSString *)notificationName
             object:(id)notificationSender
```

```
- (void)removeObserver:(id)notificationObserver
                  name:(NSString *)notificationName
                object:(id)notificationSender
```

```
- (void)postNotificationName:(NSString *)notificationName
                      object:(id)notificationSender
                    userInfo:(NSDictionary *)userInfo
```

[[NSNotificationCenter defaultCenter] postNotification: ELNotificationPersonNameChange object:self userInfo: @{@“initialName”: @“Bob”, @“changedName”: @“Bob the great”}];

Creating a custom View

- to drawRect or not to drawRect that is the question?
- Compose your view where possible
- layoutSubviews & setLayoutIfNeeded
- Layers also an option

```
- (void) layoutSubviews
    {
     ...
    }

VS

- (void)drawRect: CGRect rect
    {
     ...
    }
```

Coding Guidelines

Homework

Create a custom view that can be dragged around the screen. When touched it changes color.

Write a calculator app that adds subtracts and multiplies. Use a protocol to make the use of mathematical functions weakly bound to the calculator and the addition of more mathematical function very easy. It must have one page that is the calculator and one page that displays saved calculations. Use the notification center to broadcast when a calculation has been done. Create a count property that stores the number of calculations performed and use key value observing to display this property on the screen.

References

- App Annie Index Market Q1 2015, http://www.appannie.com/intelligence/
- iOS, Wikipedia, http://en.wikipedia.org/wiki/IOS, last edited 4 June 2015.
- iOS Human Interface Guidelines, https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/
- App Programming Guide for iOS, https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/ExpectedAppBehaviors/ExpectedAppBehaviors.html
- View Programming Guide, https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewPG_iPhoneOS/WindowsandViews/WindowsandViews.html
- The App Lifecycle, https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html
- View Controller Programming Guide, https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/AboutViewControllers/AboutViewControllers.html
- https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIKitFunctionReference/index.html#//apple_ref/c/func/UIApplicationMain
- Event handling guide for iOS, https://developer.apple.com/library/ios/documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/Introduction/Introduction.html#//apple_ref/doc/uid/TP40009541

- App Programming Guide for iOS, https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/TheAppLifeCycle/TheAppLifeCycle.html
- Introduction to Xcode, https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/index.html#//apple_ref/doc/uid/TP40010215
- UIControl, https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIControl_Class/index.html#//apple_ref/doc/uid/TP40006779
- UITableViewDelegate, https://developer.apple.com/library/ios/documentation/UIKit/Reference/UITableViewDelegate_Protocol/
- UITableViewDataSource, https://developer.apple.com/library/ios/documentation/UIKit/Reference/UITableViewDataSource_Protocol/
- Key Value Observing programming guide, https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/KeyValueObserving/KeyValueObserving.html
- Notification Programming Topics, https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Notifications/Introduction/introNotifications.html#//apple_ref/doc/uid/10000043i