

HIT3329 / HIT8329
Creating Data Driven Mobile Applications

Lecture 1

Welcome to iOS

Presented by Paul Chapman

Adjunct Industry Fellow F-ICT
Director, Long Weekend LLC
Swinburne University of Technology

Before We Begin

Lectures & Labs

12 x Tuesday Nights

Assignments

8 x homework (2 x optional)

Portfolio Assessment

Only for the brave

Rules

No Copy Paste Coding

Requirements

Access to Intel Mac

OO Programming Experience

Texts

<http://developer.apple.com>

<http://www.cocoadev.com>

Q&A

<http://stackoverflow.com>

paul at longweekendmobile dot com

You Really Should Know These

Object Oriented Fundamentals:

- Classes
- Instances
- Difference between class & instance methods
- Instance variables (ivars)
- Class inheritance
- Superclass
- Subclassing
- Interfaces/Protocols

Some of Our Work



Japanese Flash



Animal Phone

www.longweekendmobile.com/apps

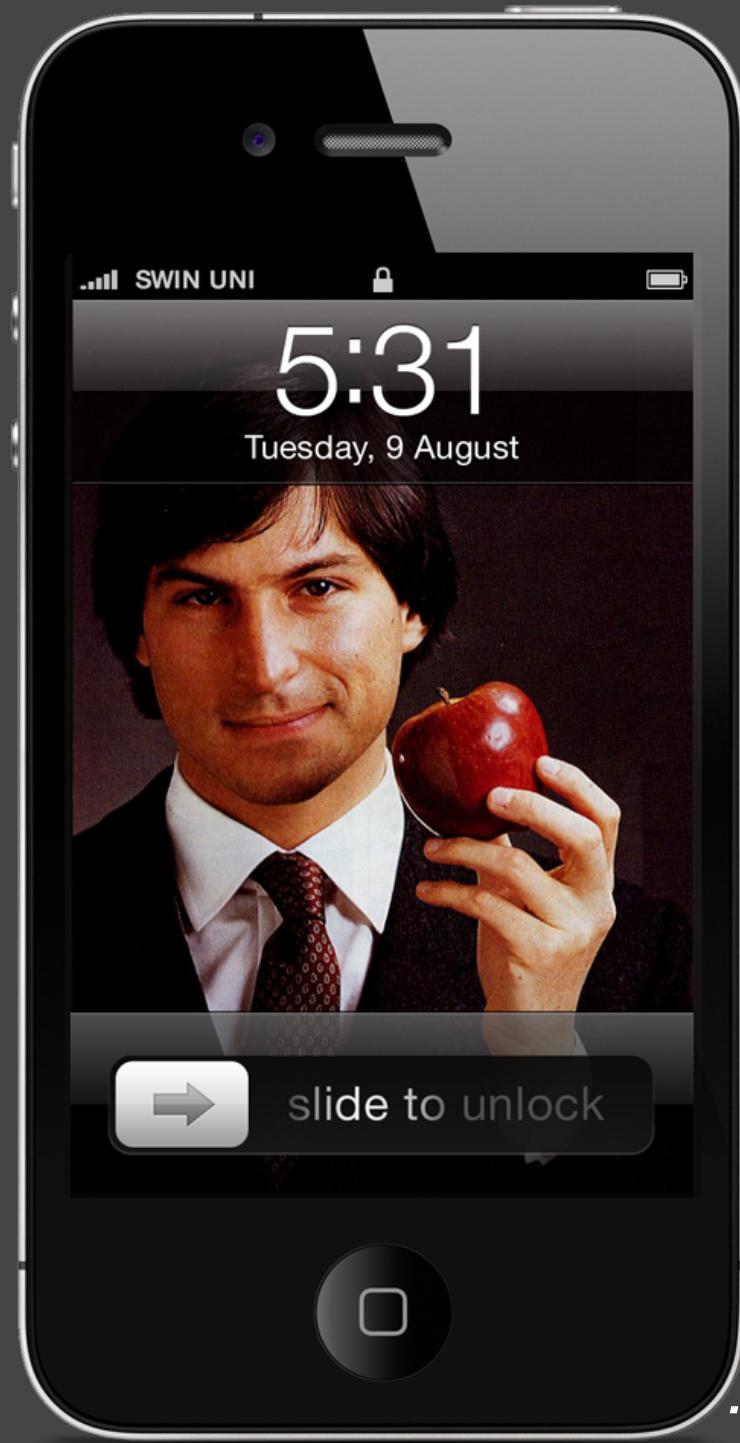
What's On For Today?

1. What This Course Is About
2. The Learning Curve
3. Objective-C
4. iOS Developer Tools

1.0 What This Course Is About

- Writing apps that are:
 - Data driven
 - Networked enabled
 - Sensor integrated
 - Touchable
 - Mobile

and most of all ...



... apps that are cool!

1.1 Learning Objectives

1. Build, test, and debug mobile applications that consume and process external data, as well as collect and publish data
2. Explain the importance of and design testable, reusable code components
3. Design mobile applications that work within device and environmental limitations
4. Use caching and concurrency to create fast applications with a better user experience

1.2 Example Data Driven App

The figure displays eight screenshots of a mobile application for Swinburne University of Technology, arranged in a 2x4 grid. The screenshots illustrate various features of the app:

- Screenshot 1:** Home screen featuring the Swinburne logo and navigation icons for ATAR, News/Events, Videos, Twitter, and More.
- Screenshot 2:** Course Search screen. It includes fields for selecting a study area (with a placeholder "Tap to select"), entering a keyword or ATAR (with a placeholder "Enter Keyword(s)"), and entering an ATAR. A large red "SEARCH" button is at the bottom.
- Screenshot 3:** Search Results screen titled "Search Results". It shows a list of courses under "Courses in Engineering and Technology": Bachelor of Arts (Digital Media), Bachelor of Arts (Digital Media and Marketing), and Bachelor of Multimedia (Media). Each course entry includes its ATAR (e.g., 71.10), location (Hawthorn), and a red circular icon with a white arrow.
- Screenshot 4:** Course Information screen for the Bachelor of Arts (Digital Media). It provides an ATAR score of 66.35, a Course Overview section, and a Contact Us section with links to "Phone us now" and "Enquire online".
- Screenshot 5:** Swinburne News screen titled "NEWS AND EVENTS". It lists three news items: "Two countries, two degrees" (about Jin You receiving two degrees), "Swinburne set to house new Design Fa..." (about a new 'Living Lab'), and "Swinburne magazine out now" (about the national magazine).
- Screenshot 6:** More screen titled "More". It contains sections for "Contact Us" (links to Phone us now and Enquire online) and "Where Is?" (listings for Hawthorn Campus, Prahran Campus, Lilydale Campus, and Wantirna Campus).
- Screenshot 7:** Enquire Now screen. It prompts the user to enter their First Name (Paul), Last Name, Email Address, Phone Number, Date of Birth (03/08/1993), and Course You Are Interested In. It also includes a "Where Is?" section for campuses.
- Screenshot 8:** More screen titled "More". It contains sections for "Contact Us" (links to Phone us now and Enquire online) and "Call Swinburne?". This section includes a blue callout asking if the user wants to call the Swinburne Call Center, with options "Don't Call" and "Call Now". It also lists campuses: Hawthorn Campus, Lilydale Campus, and Wantirna Campus.

What's On For Today?

1. What This Course Is About
2. **The Learning Curve**
3. Objective-C
4. iOS Developer Tools

2.0 The Learning Curve

- iPhone 2G released 2007
- iPhone 3G released 2008
- Long Weekend started 2009
- New, New, New
 - Language
 - Hardware
 - Tools
 - Distribution
 - Design Rules

2.1 Unfamiliar Platform

- Objective C - A Unique Language
 - object.someMethod(); // Java
 - [object someMethod]; // Objective C
- Manual Memory Management
- Centralised Distribution (App Store)
- Digital Signing of Apps
- Building to Simulator & Devices

2.2 Different Hardware

It's not a PC!

- *Reduced resources (RAM, CPU)*
- *Small screen*
- *Touch input*
- *No WIMP interfaces*

2.2 Different Hardware II

- Slow unreliable networks
- Battery can run out
- Phone can ring

2.3 The Apple App Store

Playing by the rules

- *Human Interface Guidelines*
- *Review rules unpublished (until 2010)*
- *Puzzling review process*

2.3 The Apple App Store II

- Submission lead time (up to 2 weeks)
- Bug fixes delayed (up to 2 weeks)
- Developer backend is lacking
- Even great apps get 1 star reviews

What's On For Today?

1. What This Course Is About
2. The Learning Curve
- 3. Objective-C**
4. iOS Developer Tools

3.0 Objective-C

- Dates back to 1981 *NB: C++ dates back to 1979*
- Similar to Smalltalk
- Popularised by NextStep
- Objective C Runtime
 - Provides dynamism in C
 - Open Source
 - Messaging Oriented

3.1 What is the Objective-C Runtime?

"The Objective-C Runtime is a Runtime Library ... written mainly in C & Assembler ... adds the Object Oriented capabilities to C to create Objective-C."

"... it loads in class information, does all method dispatching, method forwarding, etc ... [it] creates all the support structures that make Object Oriented Programming with Objective-C possible."

Source: <http://cocoasamurai.blogspot.com/2010/01/understanding-objective-c-runtime.html>

3.2 C Inside

- Strict superset of C
- Compatible with existing C libraries
- Optimise performance using C code
- Compatible with existing C++ libraries
- Mix C, C++ and Objective C code

3.3 Objective-C is Dynamic

- Object Oriented
- Dynamic Typing
- Introspection / Reflection

```
// get object's class name
```

```
[myObject class];
```

```
// does object responder to this 'selector'
```

```
[myObject respondsToSelector:@selector(foo)];
```

```
// get method signature for 'bar'
```

```
[myObject methodSignatureForSelector:@selector(bar)];
```

- Classes Are Objects

```
// is MyClass1 is subclassing MyClass1?
```

```
[MyClass2 isSubclassOfClass:MyClass1]
```

3.4 Objective-C is Dynamic II

- Expressive Message Syntax

[myArray insertObject:myObject atIndex:10];

- Late Binding

- Respond to unhandled messages
- Add methods to existing classes
- Extend any class at runtime with categories
- Dynamically create classes

- Language Favours Simplicity

- No templates
- No overloading
- No multiple inheritance

3.5 Message Passing

- Pass 'messages' (we don't call methods)

[object method:someObject];

- Messages matched by *method signature*

This C method declaration:

(void)doItNow(char myString[], int[] myArray);

In ObjC:

-(void)doItNow:(NSString*)stringArg withArray:(NSArray*)arrayArg;

Matches this signature:

'doItNow:withArray:'

- Unlike method calls, sending an object a message doesn't mean it will perform it!

3.6 Message Passing + Dot Notation

Setter syntax

```
object.childObject.value = 1;    // Java  
[[object childObject] setValue:1]; // ObjC
```

Setter in dot notation:

```
object.childObject.value = 1;
```

Getter syntax

```
[object childObject];      // get property  
[[object childObject] value]; // nested syntax
```

Getter in dot notation:

```
object.childObject  
object.childObject.value
```

3.7 Compiler Directives

- Compiler directives supplement standard C
- Non-standard syntax can look like this:

```
@interface ... @end  
@implementation ... @end  
@property  
@synthesize
```

- Existing C syntax has also been "augmented"

```
for(id obj in myCollection) // ObjC for loop  
for(int i=0; i<=5; i++) // C for loop
```

3.8 Header File (.h)

```
//  
// example.h - a class header file  
  
{@interface Classname : SuperClassName  
{  
    BOOL somethingHappened; // Instance variables (iVars)  
    NSInteger anInt;      // ObjC data type  
    int someOtherInt;     // C data type  
}  
  
+ (ret_type)aClassMethod;  
- (ret_type)anInstanceMethod:(param_type)param1;  
- (ret_type)myMethod:(param_type)p1 with:(param_type)p2;  
{@end}
```

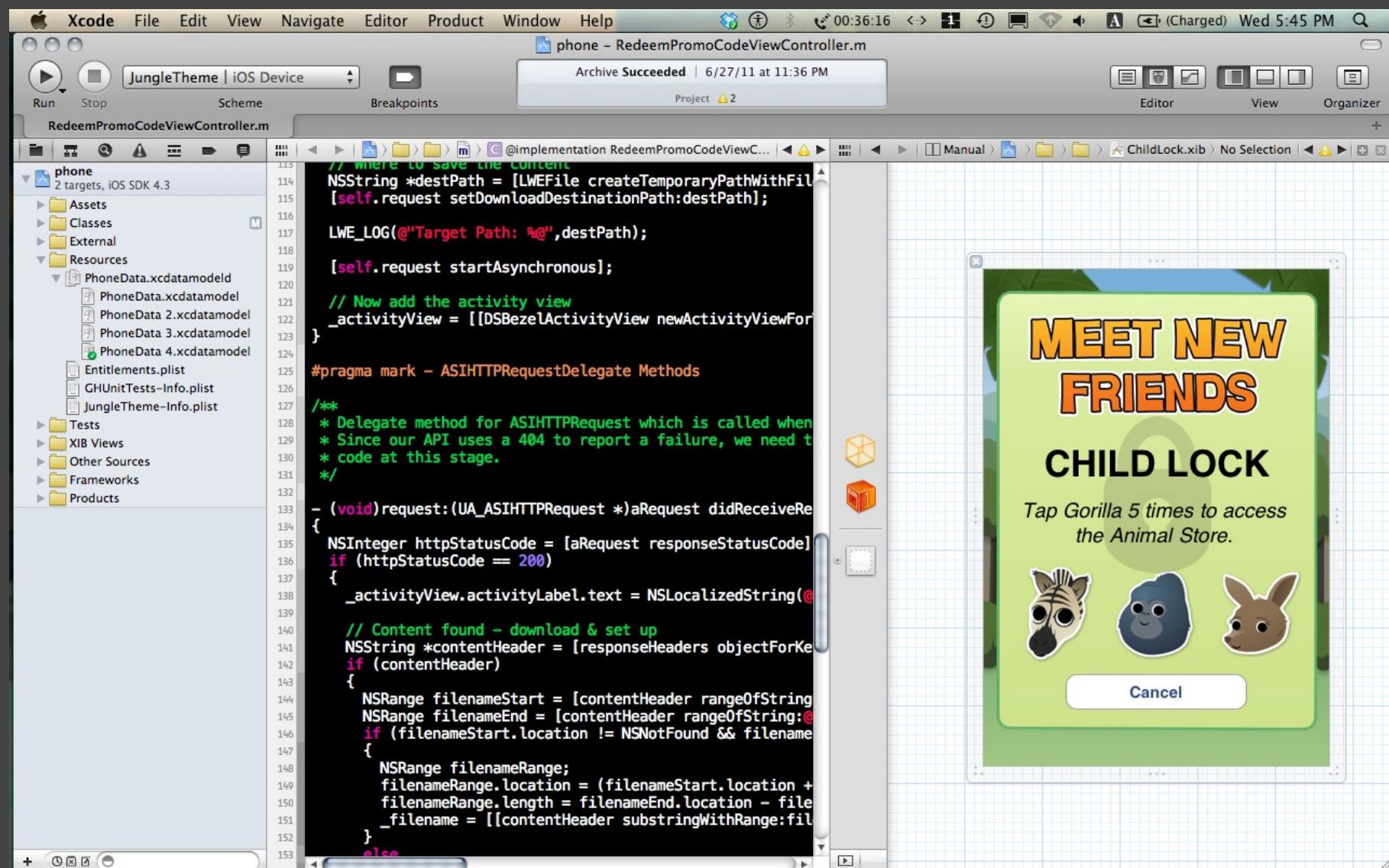
3.9 Implementation File (.m)

```
//  
// example.m - a class implementation file  
  
//  
@implementation Classname  
  
+ (return_type)aClassMethod {  
    // ...  
}  
  
- (return_type)anInstanceMethod:(param_type)param1 {  
    // ...  
}  
  
- (ret_type)myMethod:(param_type)p1 with:(param_type)p2 {  
    //...  
}  
  
@end
```

What's On For Today?

1. What This Course Is About
2. The Learning Curve
3. Objective-C
- 4. iOS Developer Tools**

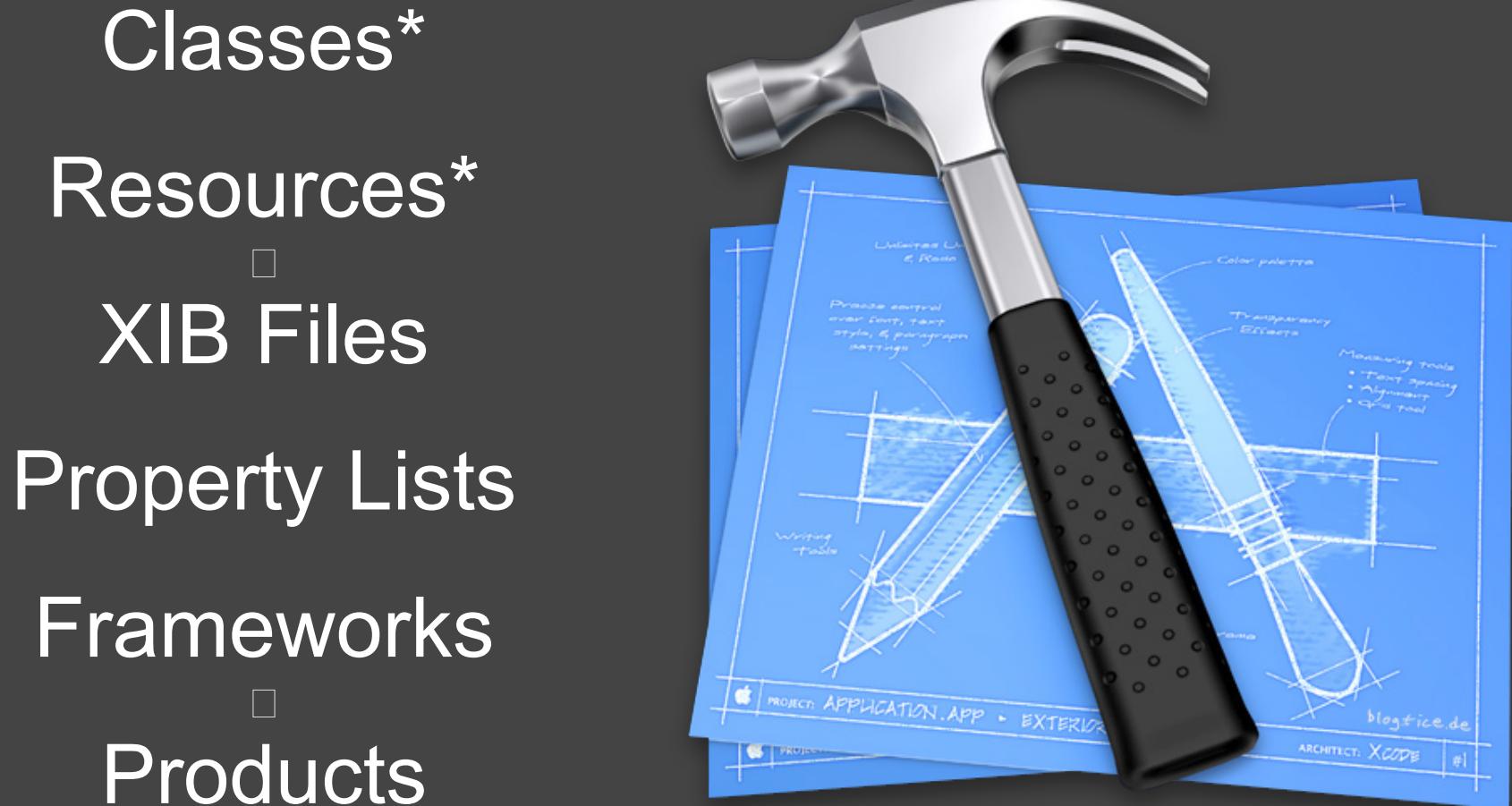
4.0 Xcode



4.1 More About Xcode

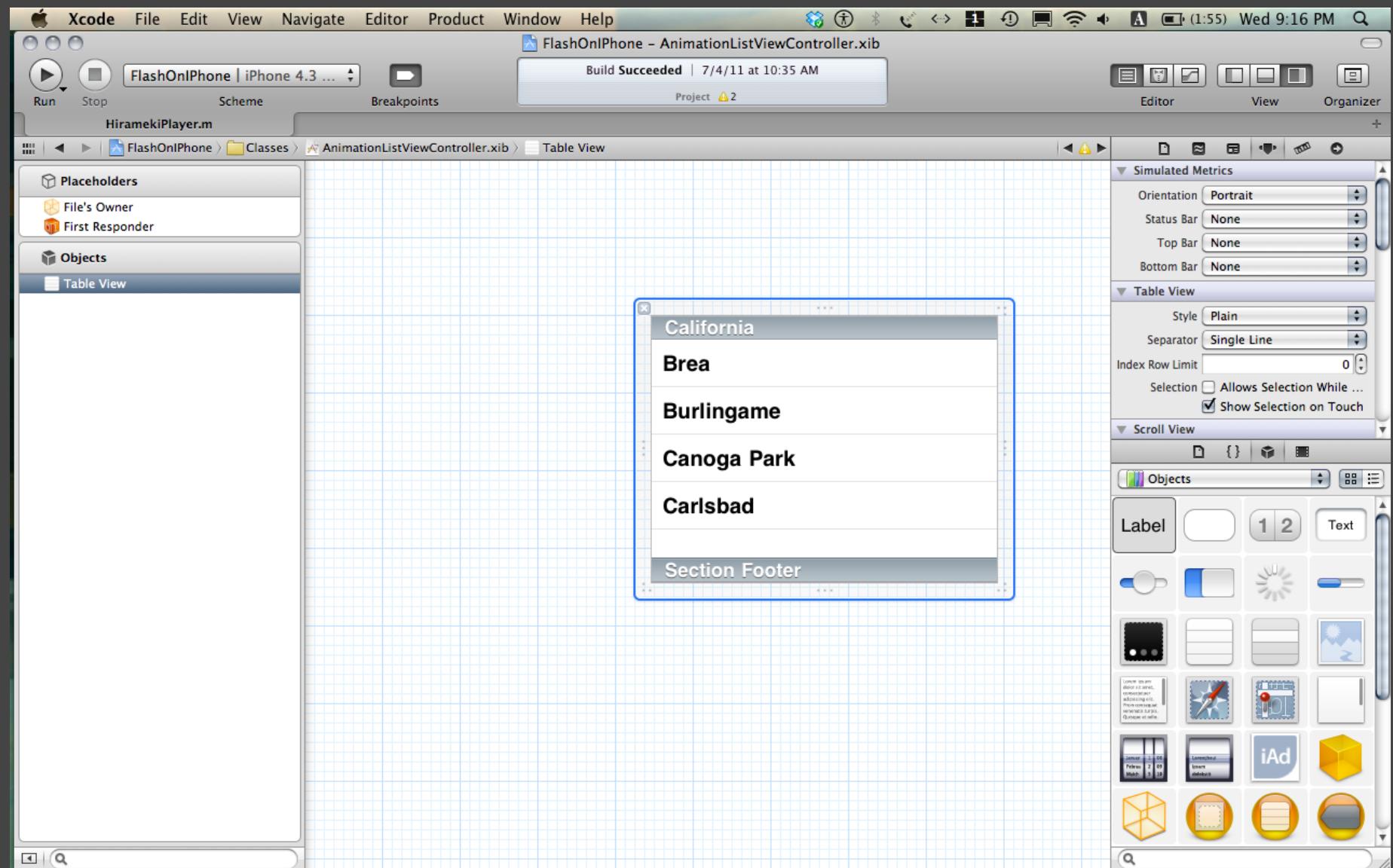
- Everyone uses it (almost)
- Free to download for ADC registrants
- Includes *Interface Builder*
- More evolved than Eclipse+Android
- Less evolved than Visual Studio

4.2 Xcode Project Tree



* Default group names, user changeable

4.3 Interface Builder



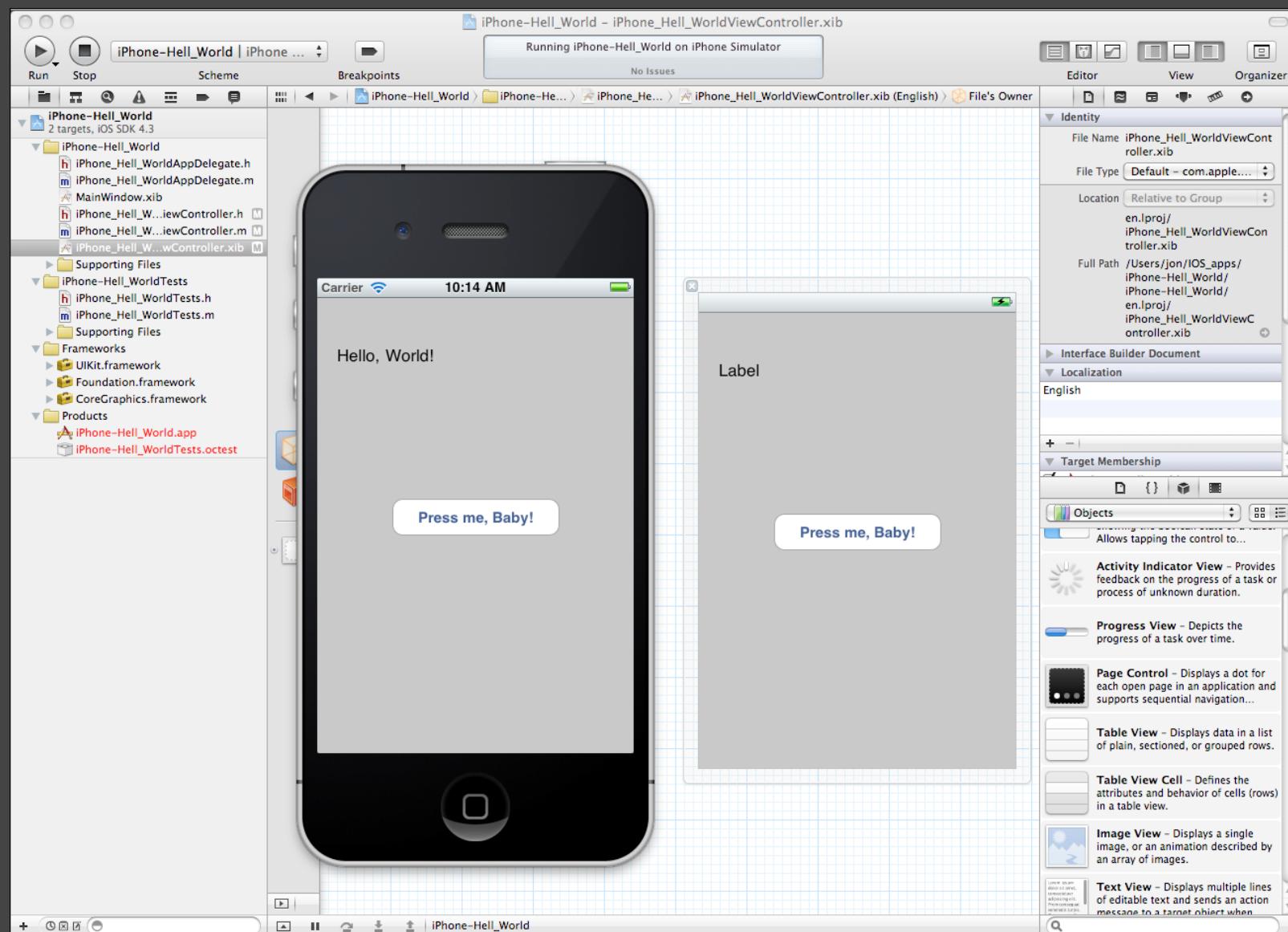
4.4 More About Interface Builder

- IB creates user interface files
- These are called XIB/NIB files
 - "*Class Archives*"
 - "*Frozen Classes*"
 - "*Serialised Classes*"
- Interfaces can be written in code too
- We "connect" XIB objects to a controller class
- Provides a library of standard interface objects

Acronym Alert!

- ★ NIB = NeXT Interface Builder
- ★ XIB = Mac OS X Interface Builder

4.5 iOS Simulator



Source: <http://xcode4iphoneheavenorhell.wordpress.com/>

4.6 Debugging Tools

The screenshot shows the Xcode interface during a debugging session. The top menu bar includes 'File', 'Edit', 'Project', 'Run', 'Stop', 'Scheme', 'Breakpoints', 'Editor', 'View', and 'Organizer'. The main window displays the code for `HPShape.m` with a breakpoint at line 15. The code is as follows:

```
//transformation is the original where this shape should be, but this shape has its own movement
//(transformation). hence the concatenation.
//also move the pen (context) afterwards.
CGAffineTransform movement = CGAffineTransformConcat(self.currentTransform, transformation);
CGContextConcatCTM(ctx, movement);

NSInteger counter = 0;
for (id fillStyle in self._fillStyles)
{
    NSArray *drawingCommands = [self._fillStyleBucket objectForKey:[NSNumber numberWithInt:(counter+1)]];
    if ((fillStyle != nil) && ([drawingCommands count] > 0))
    {
        CGPathRef path = (CGPathRef)[_fillPaths objectAtIndex:counter];
        [self _drawPath:path withStyle:fillStyle context:ctx mode:HPPFillStyling];
        counter++;
    }
}

//for every linestyle, draw the path and stroke it.
```

The status bar indicates "Running FlashOniPhone on iPhone Simulator". The bottom pane shows the "All Output" terminal window with the following content:

```
print transformation
$1 = {
  a = 1,
  b = 0,
  c = 0,
  d = 1,
  tx = 0,
  ty = 0
}
(gdb) po drawingCommands
<__NSArrayM 0x59f4200>(
[LineToEdge]From (62.000000,17.450001) to (68.150002,12.900001), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (68.150002,12.900001) to (71.700005,15.600000), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (71.700005,15.600000) to (75.500008,11.350000), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (75.500008,11.350000) to (85.000008,16.700001), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (85.000008,16.700001) to (88.000008,10.200001), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (88.000008,10.200001) to (92.850006,11.450001), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (92.850006,11.450001) to (92.900009,7.600001), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (92.900017,7.599999) to (93.000015,7.599999), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (93.000015,7.599999) to (95.550018,13.900000), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (95.550018,13.900000) to (89.000015,11.400000), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (89.000015,11.400000) to (86.000015,18.799999), fillStyle: 1, lineWidth: 0,
[LineToEdge]From (86.000015,18.799999) to (76.000015,13.349998), fillStyle: 1, lineWidth: 0,
```

A message "Thread 1: Stopped at breakpoint 15" is visible in the code editor area.

4.7 More About Debugging

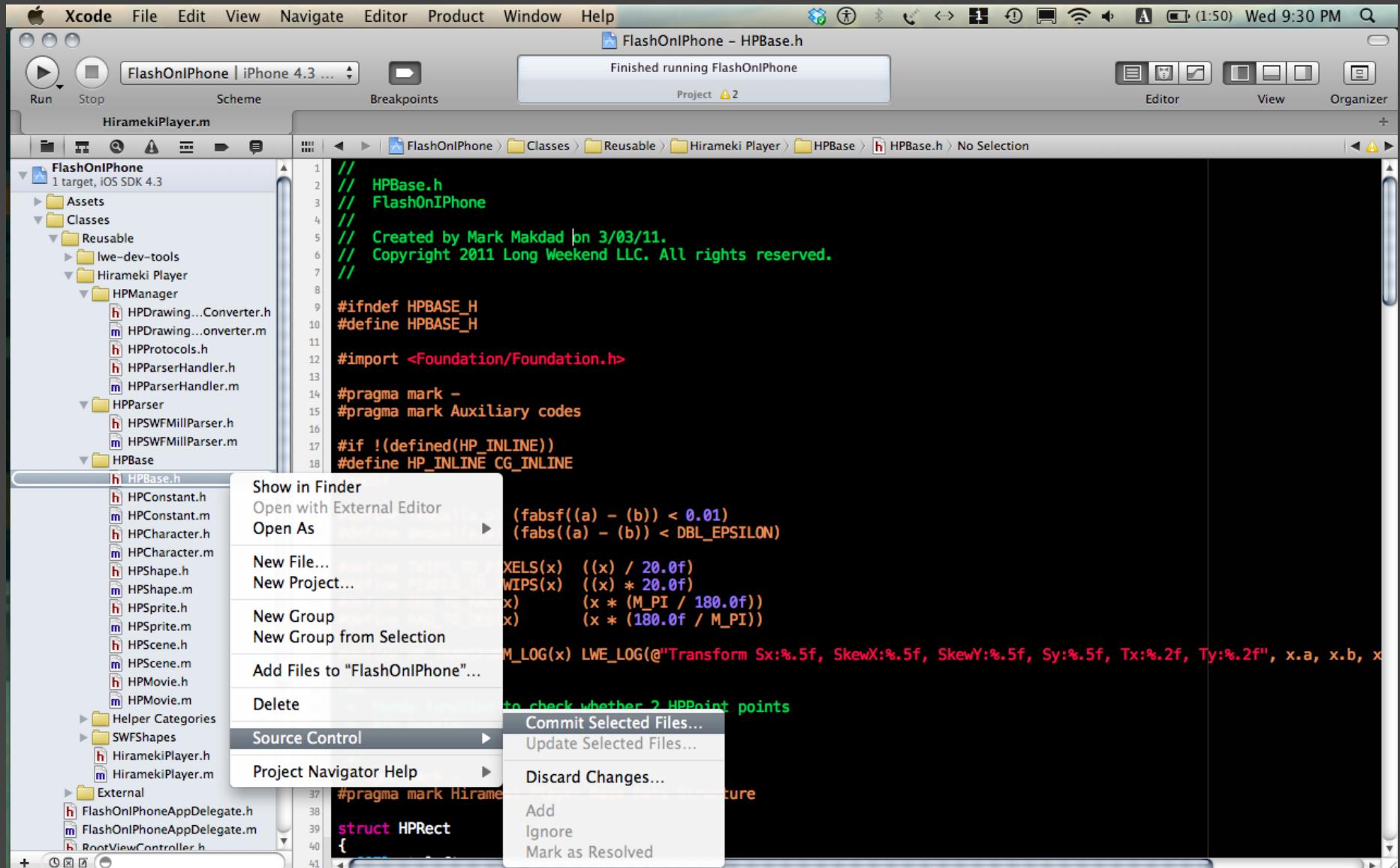
- Adding & removing breakpoints
- Stepping through code
- Debugging console commands
 - 1. print - prints a value
 - 2. po - prints objects (NSObjects)
 - 3. po [object class] - prints class name
- Log to console
 - 1. NSLog(@"%@", "Something happened!");
 - 2. NSLog(@"%@", myString);
 - 3. Takes format specifiers: %d %f %U

4.8 Tethered Debugging



Source: Apple Inc

4.9 Integrated Version Control



Integrates with SVN + Git client binaries

4.10 Cocoa Touch

Cocoa Touch Framework*

* Not indicative of size or importance

Foundation

Inherited from OS X

UIKit

UI Framework for iOS

Others: Game Kit, Map Kit, Address Book, Messaging, Core Location, Core Audio, Core Graphics, Core Animation, Core Data, etc...

"Cocoa Touch provides an abstraction layer of iOS ... Cocoa Touch is based on the Mac OS X Cocoa API ... primarily written in the Objective-C language ... allows the use of hardware and features that are not found in Mac OS X computers"

Quote Source: http://en.wikipedia.org/wiki/Cocoa_Touch

4.11 Other Included Frameworks

Audio and Video

☆ Core Audio ☆
OpenAL
Media Library
AV Foundation

Graphics & Animation

☆ Core Animation ☆
OpenGL ES
Quartz 2D

User Applications

Address Book
☆ Core Location ☆
Map Kit
Store Kit

Data Management

☆ Core Data ☆
SQLite

Networking & Internet

Bonjour
WebKit
BSD Sockets

End of Lecture 1

1. Lab Work: Starts next week
2. Familiarise yourself with ObjC syntax
3. Read the Assignments
4. *Dream of ObjC*