iPhone App Creation

LESSON 02

INSTRUCTOR : JESSE SCOTT

OBJECT-ORIENTED PROGRAMMING

#5 CLASSES / OBJECTS

Cookie vs Cookie Cutter

* The Class is the cookie cutter - the template that you create from
* The Object is the cookie - what you create from the template

OOP in OBJ-C

* Object-Oriented Programming is used extensively throughout Objective-C
* This is in part because this is how the iOS SDK is built...

#5 INTERFACE

Interface

* Blueprint For What Happens In Your Class
* Name and define your methods, but don't elaborate on the code just yet...

```
@interface ClassName : ParentClass
   -(void) someMethod;
   -(void) anotherMethod : (int) someParametre ;
@end
```

#5 IMPLEMENTATION

Implementation

* Actual Instructions & Variables For What Happens In Your Class
* Write your actual code here...

```
@implementation ClassName {
  int localVariable;
}
  -(void) someMethod {
    // code
  }
  -(void) anotherMethod : (int) someParametre {
    // code
  }
@end
```

#5 ALLOC INIT

Declare

* Declare Your Object To Be An Instance Of Your Class

```
MyClass *myObject;
```

Allocate & Initialize

* Allocate Memory Space & Set For Use

```
myObject = [ [ MyClass alloc ] init ];
```

Message

* Tell Your Class To Perform A Method

```
[ myObject someMethod ];
```

#6 SIMPLE CALCULATOR

add()

* Change the name of our Class, Object, and Method...

setVal()

* Call the method ('message the object') directly from the console input

scanf

* simple C method to get user input

#7 ADVANCED CALCULATOR

Expand Methods

* copy/modify subtract() / multiply() / divide() methods...

Determine Operator

* if/else if statement to see which operator the user has chosen...

IOS + XCODE

IOS + XCODE

* Application Templates
* Build Settings
* File Structure
* UI Library & Storyboard
* Emulator

TEMPLATES

* Use XCode to explore file structure, settings, and system architecture of iOS apps
  - create a new project with each template, and explore

* Try creating a project with/without Storyboards
  - notice the code that replaces the use of Storyboards in AppDelegate.m

* Templates are useful if you have an idea about how you want your app structured (and you're never really going to NOT use one, unless you're crazy!)

BUILD SETTINGS

* Use XCode to set your highest and lowest target for your app
  - select your project in the Navigator workspace

* Set Orientation, Launch Images, App Icons, and more
  - simply drag & drop, or ctrl-click
  - consult recommended file properties first!

FILE STRUCTURE

* .h is a 'Header File' used to hold the @interface
* .m is a source file used to hold the @implementation
* they don't have to have the same name, but it's recommended...
* File > New > File > Cocoa > Objective-C Class ...

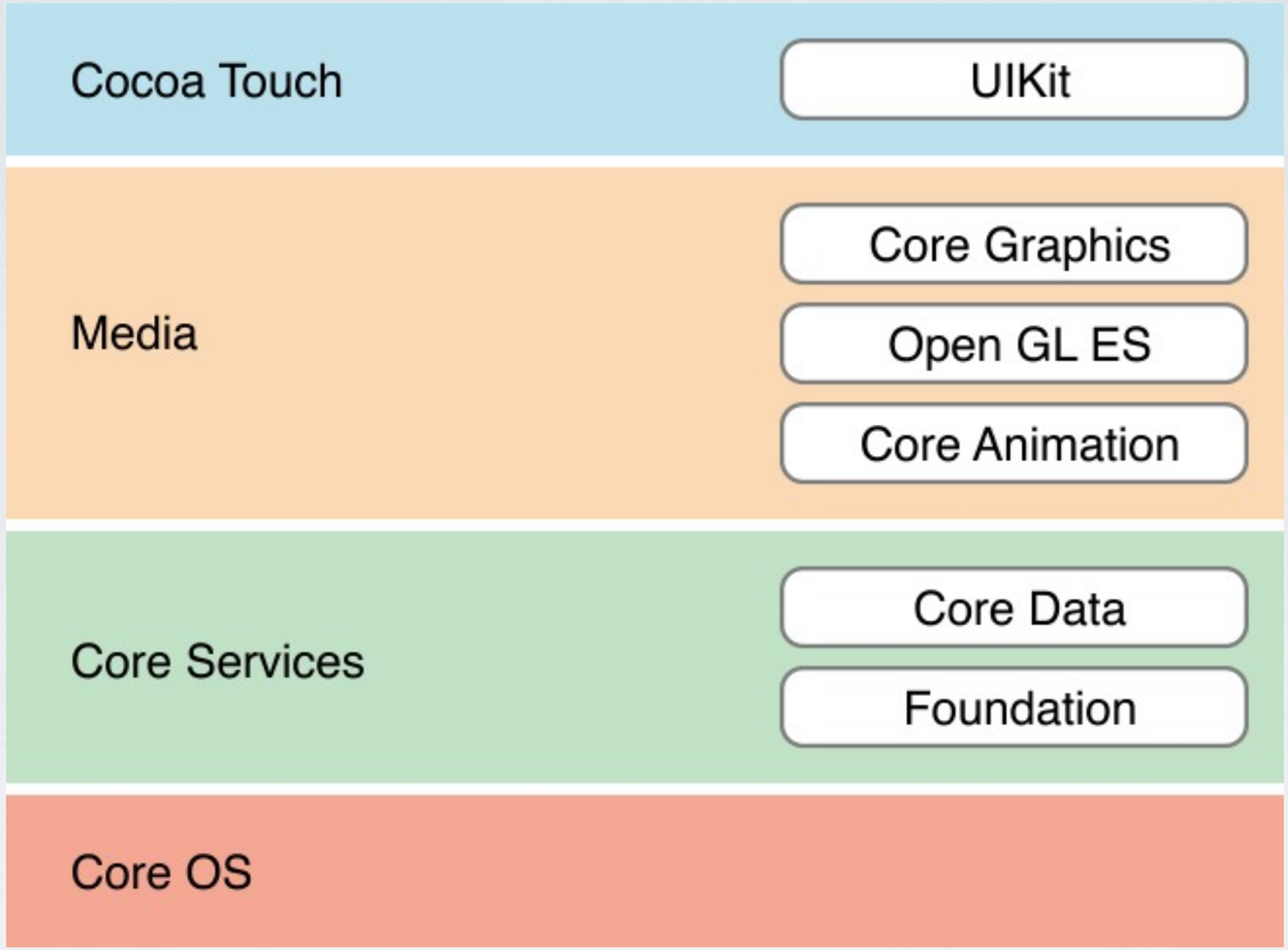
UI LIBRARY & STORYBOARD

* Ridiculously easy drag n' drop style
* Override UI look, add accessibility, etc right from the Utilities Workspace
* Tons of presets to choose from

EMULATOR

* This is your 'Target'
* Make sure you choose the right one! (and one you have designed for ...)
* It takes a while to boot up - be patient!

FRAMEWORKS



* https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/SurveytheMajorFrameworks/SurveytheMajorFrameworks/SurveytheMajorFrameworks.html

CORE SERVICES

* Foundation
  - access system resources
  - threading & memory management
  - manage datatypes such as strings & arrays

* Core Data
  - save / load to disk
  - undo / redo
  - filter, group, organize data

MEDIA

* Core Animation
  - key framing, timing, animations

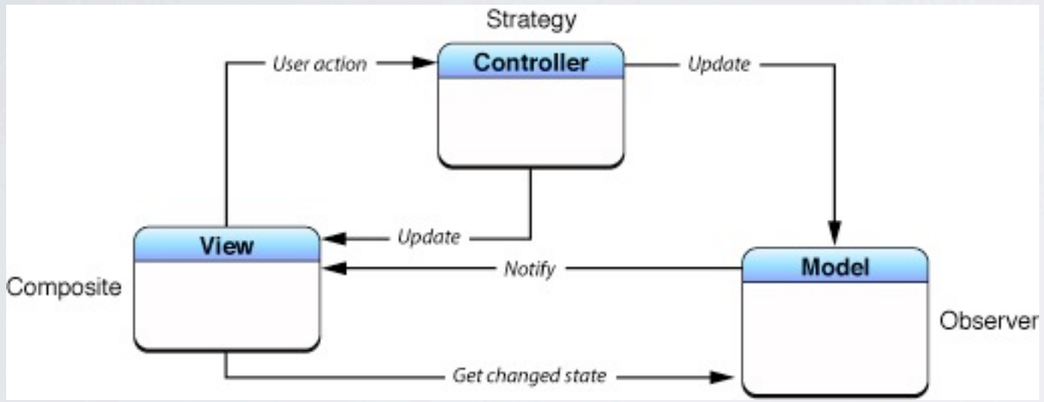
* OpenGL ES
  - access underlying graphics hardware
  - 3D graphics & games

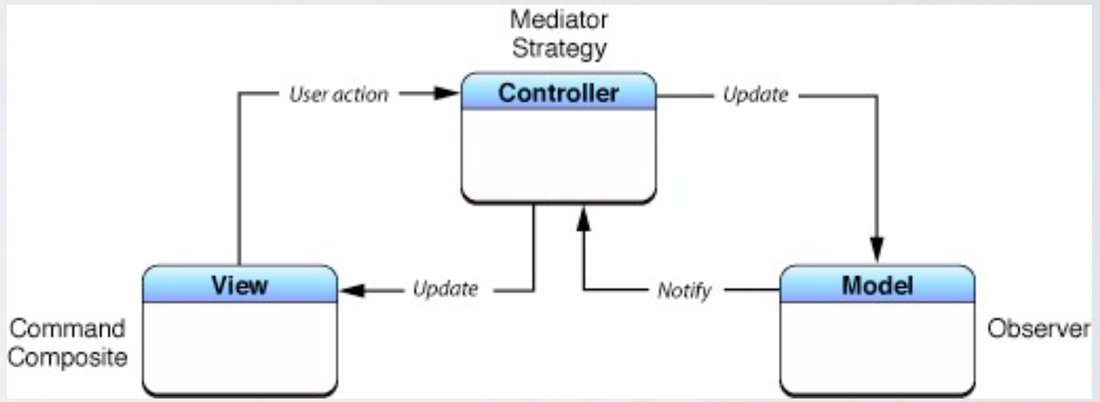
* Core Graphics
  - coordinate space drawing
  - gradients, images, colours, etc

COCOA TOUCH

* UIKit
  - handle touch events
  - display & manage User Interface
  - present text

MVC


Strategy
User action
Controller
Update
View
Update
Notify
Model
Composite
Observer
Get changed state


Mediator
Strategy
User action
Controller
Update
View
Update
Notify
Model
Command
Composite
Observer

MODEL-VIEW-CONTROLLER

* This is a 'Design Pattern' used by Apple for iOS apps
  - Helps to modularize code, making it more reusable and testable
  - Allows you to reuse interfaces for different projects, for example

* View is what the user sees...
* Model is what updates data/values in your code...
* Controller is the bridge between them

* http://developer.apple.com/library/ios/#documentation/general/conceptual/DevPedia-CocoaCore/MVC.html (or just Google it)

SOURCE CODE

#8 MVC

1) Create a SingleView iOS Application. Name it 'MVC'
2) Navigate to Object Library and add a Label, Slider, and Button to your Storyboard
3) Select the Slider > Attributes Inspector : set min/max (0-100) and initial (50) values
4) Select Label > Attributes Inspector : set initial text, colour, etc.
5) Open Assistant Editor, select Storyboard in one, ViewController.h in the other
6) Ctrl-drag from Label to Header file. Select new Outlet. Name it 'sliderLabel'
7) Ctrl-drag from Slider to Header file. Select new Outlet. Name it 'slider'
8) Ctrl-drag from Slider to Header file. Select new Action. Name is 'sliderChanged'
9) Ctrl-drag from Button to Header file. Select new Action. Name is 'resetSlider'

#8 MVC CONT...

10) Go back to Standard Editor, navigate to ViewController.m

12) Add @synthesize slider & sliderLabel

13) in sliderChanged {} add

```
self.sliderLabel.text = [NSString  stringWithFormat:@"%.02f",  self.slider.value];
```

14) in resetSlider {} add

```
[self.slider  setValue:50  animated:YES];
```

15) Save, Build, and Run > iPhone 6.0 Emulator

#9 SIMPLE CALCULATOR

1) Create a SingleView iOS Application. Name it 'Simple Calculator'
2) Add two Textfields, two Buttons, and three Labels to your Storyboard
3) Change the Button text to 'add' and 'clear'
4) Change the Label text to '+' , '=' , and ' '
5) Open Assistant Editor, select Storyboard in one, ViewController.h in the other
6) Ctrl-drag from 1st TextField to Header. Select new Outlet. Name it 'textField1'
7) Ctrl-drag from 2nd TextField to Header. Select new Outlet. Name it 'textField2'
8) Ctrl-drag from ' ' Label to Header. Select new Outlet. Name it 'label'
9) Ctrl-drag from 'add' Button to Header. Select new Action. Name is 'add'
10) Ctrl-drag from 'clear' Button to Header. Select new Action. Name is 'clear'
11) Go back to Standard Editor, navigate to ViewController.m

#9 SIMPLE CALCULATOR CONT...

11) Add @synthesize textField1, textField2, label

12) in add {} put

```
float a = ([textField1.text  floatValue]);
float b = ([textField2.text  floatValue]);
float sum = a + b;
label.text = [[NSString alloc]  initWithFormat:@"%.02f",  sum];
```

13) in clear {} put

```
textField1.text = @"";
textField2.text = @"";
label.text = @"";
```

14) Save, Build, and Run > iPhone 5.0 Emulator

CHALLENGE #1

CHALLENGE #1

* Copy the 'Simple Calculator' project
* Make three new operations - 'subtract' , 'multiply' , and 'divide'
* Pretty it up ! Make it unique.