Introduction to
iPhone/iPod Touch SDK
Programming Concepts

Session 1 Of 10
This Session

- Programming Paradigm’s
  - Target-Action, MVC & Delegation
- Introduction To Obj-C
- Introduction To Cocoa/Cocoa Touch
Programming Paradigms
Programming Paradigms

- Target-Action
- Model, View, Controller
- Delegation
Target-Action

- Target instigates an action.
- We set the target & the action to be performed.
- Eg: When a Button is pressed we call a method to display an alert
MVC

- Splits the Application into manageable parts
- Model - The data object
- View - Displays the content
- Controller - Creates the view and the Model
**MVC Example**

Model

- **Circle**
  - Path
  - Colour
  - Radius

Controller

- **CircleController**
  - Creates instances of Model and View
  - Manages an array of Circles (Model)

View

- Draws the circle & receives events
MVC Example
Real World?

Note
Holds the note Title & Content

NoteViewController
Holds the Array of CONotes & Handles the events to display them

NoteListView
Physically displays the notes on the screen
Why Should I Use MVC?

- Keeps Things clean
- Objects become resuable
- Apple uses it everywhere
Delegation

- A Delegate allows notifications to be sent back to another class
- Much simpler the subclassing, Encapsulate instead.
- A way to tap into Apple’s Event Loop
Delegation Explained

- An Object has a delegate
- This Object can ask its delegate for information
- This Object can also inform its delegate when its finished doing what it needs to do
Example

Object

Delegate

How Many Hours Should I Teach For?

7
Example...

Object

I Finished Teaching

Delegate
Introduction To Objective-C

- Strict superset of C
- Object Oriented
- It’s all Pointers!!!!! (Except for C Types)
Types

- id - Generic Type Definition
- C types: int, float, double, long, etc
- Obj-C: NSNumber, NSString, etc
NSObject

• ALL Apple classes inherit from NSObject
Self & Super?

- **Self** refers to the instance class the word is mentioned in.
- **Self** is the same as **this** in say C++ or Java
- **Super** refers to the class’ super class
Objective-C
Source Files

- `.h` for headers
- `.m` for implementation
Declaring Methods

C

```c
void killBrock(void *itemToKillWith);
```

C++

```c++
void killBrock(MyClass item);
```

Objective-C

```objective-c
-(void)killBrock:(id)ItemToKillWith;
```
Method Types

- Instance Methods
- Class Methods

- + (void) myClassMethod;
- - (void) myInstanceMethod;
Sending a Message / Method Calls

C

myStruct.object

C++

object.killBrock(Gun);

Objective-C

[object killBrock:Gun];
SHOW ME CODE!!!!!
#import <UIKit/UIKit.h>
@interface AUCPerson : NSObject
{
    NSString *FirstName;
    int age;
}
-(void)IncrementAge;
@end
#import "AUCPerson.h"

@implementation AUCPerson
- (id) init
{
    self = [super init];
    if (self != nil) {
        FirstName = @"AUC Generic";
    }
    return self;
}
@end
-(void)IncrementAge
{
    age++;
}

-(void)dealloc
{
    [super dealloc];
}
@end
Properties

- Dot Syntax for easy access to an instance variable of an object
#import <UIKit/UIKit.h>
@interface AUCPerson : NSObject
{
    NSString *FirstName;
    int age;
}
-(void)IncrimentAge;
@property (retain) NSString *FirstName;
@end
#import "AUCPerson.h"

@implementation AUCPerson

@synthesize FirstName;    // property

- (id) init{
self = [super init];
if (self != nil) {
    FirstName = @"AUC Generic";
}
return self;
}
Creating An Instance

- Allocating a new instance

```objective-c
AUCPerson *nick = [[AUCPerson alloc] init];
nick.firstName = @"Nicholas";
```
Logging Out

- `NSLog();`
- It’s the Cocoa version of `printf()`
- Warning: Pass the right number & types of iVar’s (or it will crash at runtime!)

```swift
NSLog(@"The Student Firstname Is %@",nick.firstName);
```
Memory Management

- Everything you allocate you MUST release
- Same as new and delete in C++
- alloc, release, retain, autorelease
- No Garbage Collection (by default)
Cocoa Classes

• A lot of them
• Very powerful
• Often just add NS or UI before a variable
• NSString, UIView, NSInteger, NSUInteger, NSUrl
Immutable

• Classes such as NSString are not mutable
• That is you can’t edit them once created

NSString *sentence = @"His Name is Robert Paulson";
Mutable

• However `NSMutableString` is mutable:

```c
NSMutableString *sentence = @"His Name is";
[sentence appendString:@"Robert Paulson"];
```
Questions?