iPhone/iPod Application Development

Class # 3, Fall 2016

Properties

- Object encapsulates it's data through it's properties.
- For e.g. @property NSString *myName in @interface section of the class declares string property called myName. In this example @property in @interace section of the class is a public property.
- You can have class properties visible only within the class.

Accessor Methods

- Methods to access data - i.e. get data or set data are available after synthesizing properties. In order to synthesize a property called myData, you will need to add a statement in @implementation section: @synthesize myData.
- Compiler will generate setters and getters for synthesized properties.
- Synthesized method to get data for property called "myData" will be a method called myData. For e.g. myVar = [myClass myData] will provide you value of myData property by sending myData message on myClass.
- You may access properties directly with . notation. Using previous example: myVar = myClass.myData will provide you same access.
- You may set property value using dot notation as well: myClass.myData = myVar
- Generally you would set your instance variables to property. However you public class properties don't have to be instance variables.

Type of Properties

- By default properties are atomic. The get or set operation is atomic and completes without interruptions. Use atomic properties when integrity of operation is important. The gets will be protected from sets by another thread in progress and vice-e-versa. Declare property to be non-atomic if concurrency is not important. Atomic operations are CPU intensive.
- Readwrite property allows operations on the property as you would guess from the name. The opposite of readwrite is readonly.
- Assign, weak, retain or copy type of properties are to describe how properties interact with memory management. Assign is the default.
  - Weak property will zero our pointers and it's used with ARC.
  - Retain or Strong property will retain on assignment with new value and old value will be released when objects are subclassed
  - Copy is used for mutable type of property. Users can send a mutable version and change them without using objects' setters.
- You can use multiple characterization of a property. For e.g. @property (nonatomic, readonly, weak) NSInteger myProp;
- Source and good read on the topic is at:

  https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/EncapsulatingData/EncapsulatingData.html

Synthesized Accessors Example

- Review exercise 7.1 where access to the properties is available through methods.
- Check out exercise 7.2 and note that getter and setters are no longer needed.
- Preferred way to access getter and setter, plus it's free.
- Access properties with "." notation.

Instance.property = value

Category

- Categories can be used to define additional methods of an existing class without subclassing even when source code is unavailable.
- Useful to adopt and extend behavior of framework class where source code is not available to you.
- Added methods are inherited by subclasses. There will be no difference between category or class methods whose categories were adapted. In other words sub class won't know class method from category.
- Syntax to for category definition in .h file (from http://developer.apple.com)

```
#import "SystemClass.h"

@interface SystemClass (CategoryName)
// method declarations
@end
```

Category

- Implementation of category in .m file (source http://developer.apple.com)

```
#import "MyClass.h"

@interface MyClass (PrivateMethods)
// method declarations
@end

@implementation MyClass
// method definitions
@end
```

- Class Example

Extending Class with Categories

- Extend class behavior to an existing class by using category.
- Category can be declared for any class, even framework classes for which you don't have implementation details.
- Any method in a category will be available to all instances of the Original class as well as any subclass of the original class.
- At run time there is no difference between a method added by a category and method implemented in original class.
- Demo of category implementation from chapter 11 of the Kohn book.

Extending Class with Categories Cont'd

- You can define categories between @interface and @ end in class definition but better to put category in it's own .h and .m. Kohn book puts category definition and implementation in .m file for convenience.
- Once category is declared and implemented, you can use methods from any instance of the class in effect extending class without knowing details of implementation of the base class.
- Behavior of class can be extended but not the properties of a class. You can even include property declaration in a category but not possible to declare an additional property instance variables - just methods.
- Useful to extend an existing class by adding behavior that is useful only in certain situations. For e.g. you need to display string in certain way for a specific condition in your app. Adding that behavior by adding category to NSString would be a good use of category.

Inheritance

- Various Terminology to describe class hierarchy. Most common is child classes and parent classes. Classes, Subclasses and Superclasses is another. They mean to say when a new class is defined, the class inherits properties from parent or superclass. Class that has no parent is a root class.
- Hierarchy of search will start from the subclass and then it's parent and so on. If a method is not found in the search the message may not be generated. More on this later.
- Extending a class – typical OO concept. Review program 8.2

@try, @catch, @throw and @finally

- Exception handling very similar to C++. Test conditions that can fail using try and handle exceptions in catch block. Generally catch would have logging to understand exception, possibly corrective action and possibly terminating the program after general housekeeping.
- Demo program 9.5 to show Exception Handling
- Use @throw to throw and exception from your software. Similar to the system throwing exception for unusual events.
- @finally begins a block that's executed regardless of the exception.

```
@try {
...
}
@catch (CustomException *ce) {
// 1...
}
@catch (NSException *ne) {
// 2// Perform processing necessary at this level....
}
@finally {
// 3// Perform processing necessary whether an exception occurred or not....
}
```

Overriding Methods

- Methods inherited from parent class cannot be removed but they can be modified or overloaded. Give same method name and provide implementation of the method to be overridden in child class. Code using child class will use overridden method and code using parent class with use method provided in the parent class.
- The complier will search up the hierarchy for a method to apply.
- Example 8.6 on Overloading Methods
- Follow exercise 8.8 if you are fuzzy on this concept

Protocols & Delegates

- Protocol is a set of behavior that is expected of an object in a given situation. Real world analogy is Law enforcement officials required to “follow protocol” when making enquiries or collecting evidence.
- Protocols - Methods to be used for a particular situation.
- Delegate - Object in charge of providing implementation of protocols.
- Generally an object that agrees to implement protocols is called a delegate object - i.e. implementation of protocols is delegated to delegate object.
- Formal protocols are mandatory - all of the formal protocols must be implemented by the delegate. Compiler checks at compile and run time for implementation.
- Informal or Optional protocols may be implemented by the delegate. Some or none of the informal or optional protocols maybe implemented.

Misc topics before we delve into apps

- delegate: An object that acts on behalf of another object
- Static typing: Fraction myFrac; is an example of static typing. We are telling compiler at build time nature of myFrac, what methods and variable types will be encountered.
- Dynamic typing: Discovering the class of an object and it's attributes at runtime rather than at compile time.
- polymorphism: Ability of different object to respond, each in its own way to the same message (problem 9.1 from Kohn book).

Basic introduction to UI Kit

The Basic Canvas of iPhone/iPod – UIWindow and UIView

- Think of UIWindow as TV and UIView as shows that come on to TV. Keep one window to display many different views as opposed to one view per window.
- Technically UIWinow is a container for UIView. UIView in turn contains other objects that provide next lower level of details such as text, audio, video etc.
- Objects used for Data display:
  - UITextView presents text and allows text edit.
  - UILabel presents read-only text view. Our "Hello World" program in first class used UILabel. Varity of property such as font, color, background etc. available to spice up the label. Generally pick size of the label, work rest of your design and iterate back to improve Human Interface factors.
  - UIImageViews presents images loaded with UIImage objects. UIImageView scales to fit contents. You can also load sequence of images and animate them on demand.

Data Display Continued

- UIWebView offers display capabilities for nearly all data types supported by the built-in Safari browser. Editing not allowed but other features such as zoom and scroll available. User experience very similar to Safari web browser.
- MKMapViews present Map Kit capability to view map information and interact with the map content as you would with the Maps application. You can annotate using MKAnnotationView and MKPinAnnotationView classes.
- UIScrollView to present content that is larger than normal size of window. Better to use UIScrollView than UITextView unless you know that UITextView will not exceed the size.
- UIAlertView produces pop up window. Buttons can be customized to ask users questions. Absence of buttons make them Information Alerts.
- UIActionSheet offers menu that scroll up from the bottom of the screen. Action Sheet display a message and present buttons for the users to choose from. Functionally they are same as UIAlertView except that there is more than one choice.

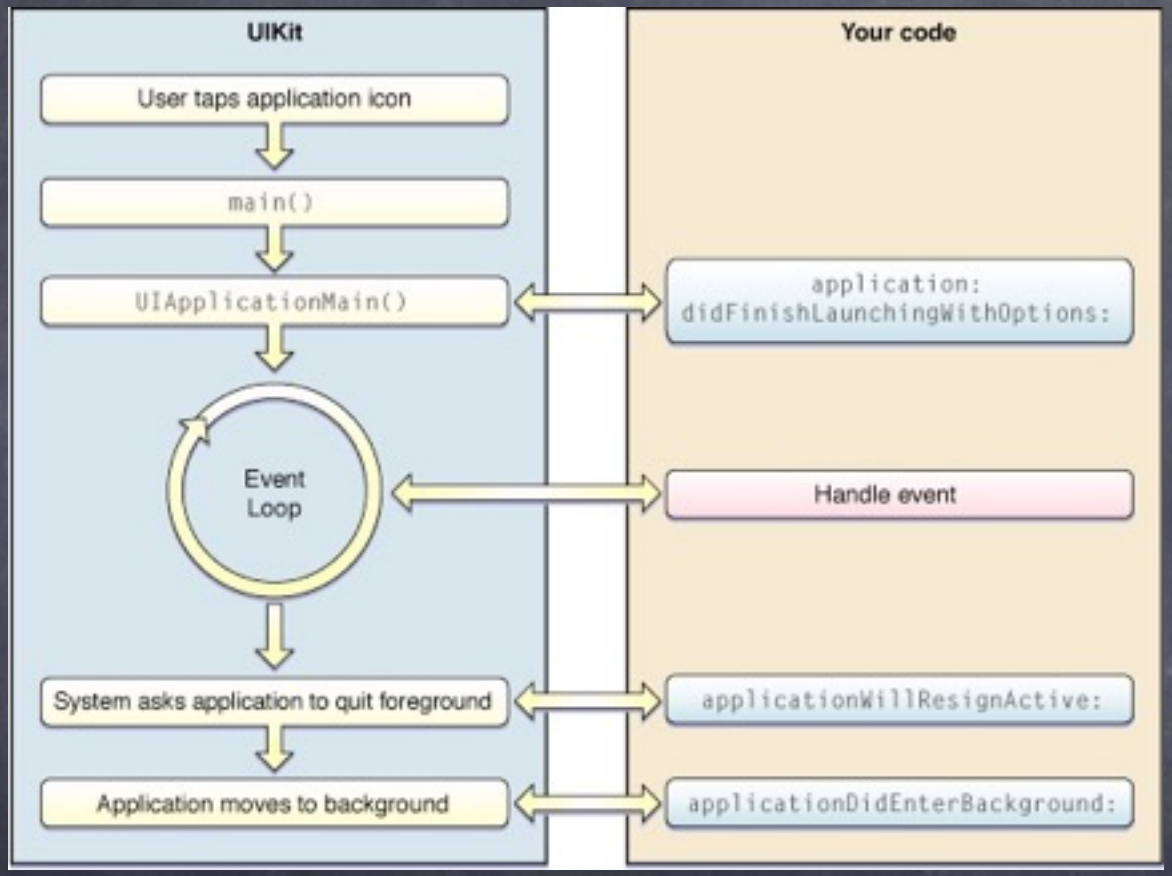
Controls

- Onscreen objects that transform the user touches into callback triggers. They may provide numeric or test values consumed by your application. Controls include buttons, switches, sliders etc.
  - UIButton provides onscreen buttons, when pushed will trigger a callback via target/action programming. Typical trigger is Touch up inside – i.e. when user releases touch on that button.
  - UIBarButtonItem store the properties of toolbar and navigation bar buttons but are not visible button themselves.
  - UISegmentedControl offers row of equally sized buttons like buttons on Radio or piano if look at elevation view. Just like buttons on Radio, you can push only one button at a time.
  - UISwitch offers binary on/off switch control.
  - UISlider allows users to choose a value from a specified range by sliding an indicator along a horizontal bar.
  - UIPageControl offers a series of small dots on screen taking you to different pages.
  - UITextFields lets you enter text. Generally small text input, like name, city, street etc.

Tables, Bars, and Progress Markers

- UITableView class provides most commonly used table style to select a row for example a line on iTunes.
- UINavigationBar usually presented on top of many interfaces to provide navigation state.
- UIActivityIndicatorView offers a spinning-style wheel for an ongoing task.
- UIProgressView offers a way to feedback % completion of a task.

Run Loop



UIApplication has 4 args: first two are traditional argc, argv. Third parameter is name of application's principle class – responsible for running the app. Defaulting it to null tells UIKit to use UIApplication. Fourth parameter is name of application delegate class – responsible for managing high level interaction between framework and your code. Value of nil tells UIKit to load application delegate from application's main nib file.

Source http://developer.apple.com

Target Action

- A pattern in which an object holds the information necessary to send a message to another object when an event occurs. An action selector identifies the method to be invoked. Target is the object to receive the message.
- Read detailed article on wikipedia under Command Pattern. Also read up documentation on developr.apple.com for detailed discussion.
- Hands on class exercise for target action using UIAlertSheet

Summary

- Typical class files, extensions of classes and discussion on how objective C, Xcode produce a product in an IDE.
- Categories useful to extend class - a different notion than true OO languages.
- Protocols and Delegates are Objective C language constructs and new to all of you. Both will be used a lot and you will get familiar with them. We will cover protocols and delegates later in the class as we use concrete examples.
- Introduction to UI kit in iOS, the first foundation block to build apps.
- Look at IDE options with Target Action app, and a general idea on what changes you will be making to realize an app.
- No HW this week. I will assign HW next week