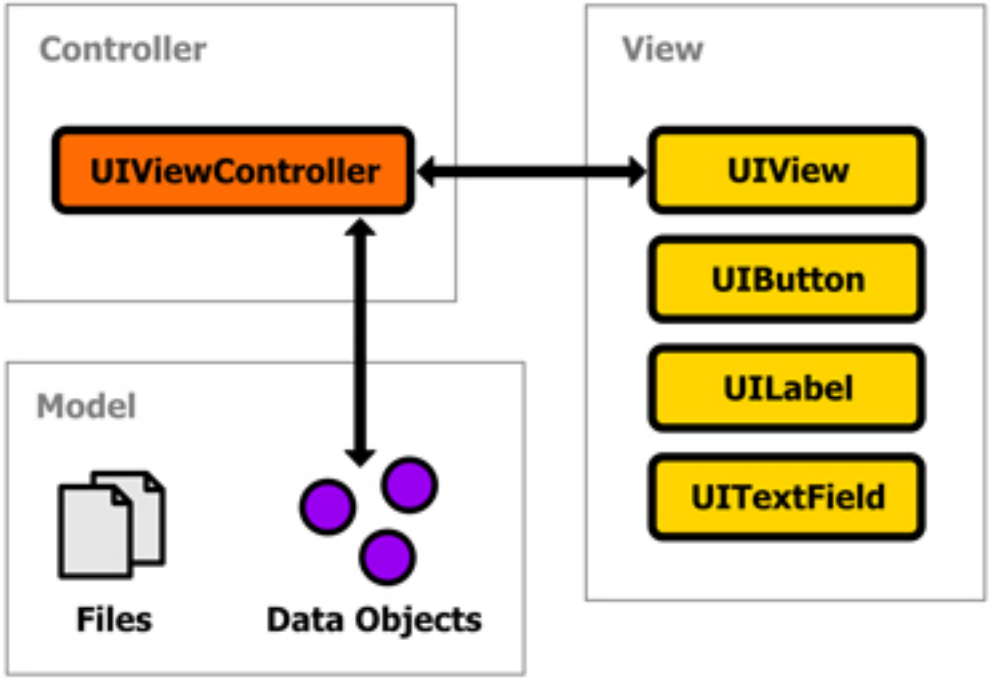
ITP 342
Mobile App Dev



Model

MVC

- Model – Hold data
- View – User Interface
- Controller – Views & View Controllers
  - Every app has a single *window*
  - Every screenful of content ("scene") sits in a *view*
  - Navigation changes the window's current *view*
  - Each screen's *view* is managed by a *view controller*

Model Class

- Create a new model class
- Use one of 2 ways
  1. Use the Library – File Template Library
     - Drag **Cocoa Touch Class** to the Project Navigator
  2. From the main menu, select **File → New → File…**
     - Under the iOS Source template, select **Cocoa Touch Class**
- Either way the .h & .m files will be created.
  - The second way allows you to select the parent class.

Interface

- In the interface file (.h) add any public properties and methods
  - These are the ones that we want other classes to access

```
// Student.h

#import <Foundation/Foundation.h>

@interface Student : NSObject

// public property
@property (strong, nonatomic) NSString *firstName;
@property (strong, nonatomic) NSString *lastName;

// public methods
- (NSString *) fullName;

@end
```

Implementation

- In the implementation file (.m) add any private properties and implement all methods

```
// Student.m

#import "Student.h"

@interface Student ()
// private property
@property (strong, nonatomic) NSString *userName;
@end

@implementation Student
- (NSString *) fullName {
  NSString *fullStr = [ [NSString alloc]
    initWithFormat: @"%@ %@",
    self.firstName, self.lastName ];
}
@end
```

Properties

- For any private and public properties that are objects, we need to alloc/init them.
- Where?
  - We create our own **init** method.
  - When a different class (such as a ViewController or AppDelegate) creates a property of our model class, then it can call the **init** method.

init

- Use the Library – Code Snippet library
  - Search for **init**
  - Find **Objective–C –init Method**
  - Drag into .m file
- Use the ivar (*_propertyName*), not self.*propertyName*

Implementation

```
// Student.m
@interface Student ()
// private array
@property (strong, nonatomic) NSMutableArray *orgs;
@end

@implementation Student

- (id) init {
  self = [super init];
  if (self) {
    _orgs = [ [NSMutableArray alloc] initWithObjects:
      @"ACM", @"Girls in Tech USC",
      @"Women in Computing", nil];
  }
  return self;
}

@end
```

Interface

```
// Student.h

#import <Foundation/Foundation.h>

@interface Student : NSObject

// public methods
- (NSUInteger) numberOfOrgs;
- (NSString *) randomOrg;
- (NSString *) orgAtIndex: (NSUInteger) index;

@end
```

Implementation

```
// Student.m

- (NSUInteger) numberOfOrgs {
  return [self.orgs count];
}

- (NSString *) orgAtIndex: (NSUInteger) index {
  return [self.orgs objectAtIndex: index];
  // Alternative way:
  // return self.orgs[index];
}
```

Implementation

- Get a random index for the array
  - Use the `random()` function in stdlib to get a random number
  - Use the `%` (mod) operator to get a number between 0 and `count` - 1

```
// Student.m

- (NSString *) randomOrg {
  return [self.orgs objectAtIndex:
    random()%[self.orgs count]];
}
```