Today

- **MVC**
  - Calculator

- **Objective-C**
  - Declaring and implementing objects
  - Sending messages between objects

- **Interface Builder**
  - Graphically creating your View
  - “Wiring up” objects to send messages to each other
  - Setting the properties of objects

- **Xcode**
  - Managing and editing your code
  - Running your application in the simulator
Calculator MVC

Controller

Model

View

should

will

did

data

count

data source

data source

outlet

action

target

delegate

Stanford

CS193p

Fall 2010
Controller MVC

Controller

Model

View

CalculatorBrain

target
outlet
action
Calculator MVC

Controller

Model
CalculatorBrain

View

CalculatorViewController

target

outlet

action
Calculator MVC

Controller

CalculatorViewController

Model

CalculatorBrain

View

IBOutlet

UILabel

UIButtons

3.21
1
4
7
2
5
8
0
+ 
- 

Controller

target

outlet

action
Calculator MVC

Model
CalculatorBrain

Controller

View
UIButtons

display

target

action
Calculator MVC

Controller

CalculatorViewController

target

UILabel

+ -

3.21

CalculatorBrain

Model

digitPressed:

View

UIButton

operationPressed:

display
This is the header file for this class. It documents its public API.
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject

@end

@interface CalculatorBrain : NSObject

@end
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject

@end

This class's superclass.
@import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
@end

Model

We must import the header for our superclass.
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
  // Instance variables go here.
}
@end
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}
@end
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
  double operand;
}
@end
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;
- (double)performOperation:(NSString *)operation;

@end
Specifying void as the return type means that this method returns no value.

```objective-c
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;
- (double)performOperation:(NSString *)operation;
@end
```
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

The name of this method is “setOperand:”

- (void)setOperand:(double)anOperand;
- (double)performOperation:(NSString *)operation;

@end
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;
- (double)performOperation:(NSString *)operation;
@end

It takes one argument, a double called "anOperand"
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

-(void)setOperand:(double)anOperand;
-(double)performOperation:(NSString *)operation;
@end
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;

- (double)performOperation:(NSString *)operation;

@end

This method returns a double.
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;
- (double)performOperation:(NSString *)operation;

@end

It takes as its argument a pointer to an NSString object. That's right, we're passing an object to this method.
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

-(void)setOperand:(double)anOperand;
-(double)performOperation:(NSString *)operation;
-(NSArray *)foo:(int)zap bar:(id)pow;
@end
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

-(void)setOperand:(double)anOperand;
-(double)performOperation:(NSString *)operation;
-(NSArray *)foo:(int)zap bar:(id)pow;
@end

This method takes two arguments and is called “foo:bar:” (pronounced “foo colon bar colon”)
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;
- (double)performOperation:(NSString *)operation;
- (NSArray *)foo:(int)zap bar:(id)pow;
@end

It returns a pointer to an NSArray (a collection class in Foundation).
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

- (void)setOperand:(double)anOperand;
- (double)performOperation:(NSString *)operation;
- (NSArray *)foo:(int)zap bar:(id)pow;
@end

The second argument is of type “id”. This means “a pointer to any kind of object!”
#import <Foundation/Foundation.h>

@interface CalculatorBrain : NSObject
{
    double operand;
}

-(void)setOperand:(double)anOperand;
-(double)performOperation:(NSString *)operation;
@end
This is the implementation file. Both public and private implementation goes here.

```objc
#import "CalculatorBrain.h"

@implementation CalculatorBrain

@end
```
We must import our own header file.

```objective-c
#import "CalculatorBrain.h"

@implementation CalculatorBrain

@end
```
#import "CalculatorBrain.h"

@implementation CalculatorBrain

Note that we don’t specify our superclass in the implementation

@end
#import "CalculatorBrain.h"

@implementation CalculatorBrain

- (void)setOperand:(double)anOperand
{
    <code goes here>
}

@end
#import "CalculatorBrain.h"

@implementation CalculatorBrain

- (void)setOperand:(double)anOperand
{
    operand = anOperand;
}

- (double)performOperation:(NSString *)operation
{
    [operation sendMessage:argument];
    return aDouble;
}
@end
```c
#import "CalculatorBrain.h"

@implementation CalculatorBrain

- (double)performOperation:(NSString *)operation {

    return aDouble;
}

- (void)setOperand:(double)anOperand {
    operand = anOperand;
}

@end
```

Square brackets mean “send a message.”
#import "CalculatorBrain.h"

@implementation CalculatorBrain

-(double)performOperation:(NSString*)operation { [operation sendMessage:argument]; return aDouble; }

-(void)setOperand:(double)anOperand { operand = anOperand; }

@end

This is the object to send the message to (in this case, the NSString called “operation” that was passed as an argument to performOperation:).
#import "CalculatorBrain.h"

@implementation CalculatorBrain

-(double) performOperation:(NSString *)operation {
    @return aDouble;
}

-(void) setOperand:(double)anOperand {
    operand = anOperand;
}

@end
```c
#import "CalculatorBrain.h"

@implementation CalculatorBrain

-(double)performOperation:(NSString *)operation{
    [operation sendMessage:argument];
    return aDouble;
}

-(void)setOperand:(double)anOperand{
    operand = anOperand;
}
@end
```

And this is its one (in this case) argument.
#import "CalculatorBrain.h"

@implementation CalculatorBrain

-(double)performOperation:(NSString *)operation {
    [operation sendMessage:argument];
    return aDouble;
}

-(void)setOperand:(double)anOperand {
    operand = anOperand;
}

@end
#import <UIKit/UIKit.h>

@interface CalculatorViewController : UIViewController
{
    CalculatorBrain * brain;
    IBOulet UILabel * display;
}

- (IBAction)digitPressed:(UIButton *)sender;
- (IBAction)operationPressed:(UIButton *)sender;
@end
#import <UIKit/UIKit.h>

@interface CalculatorViewController : UIViewController

IBOutlet UILabel *display;

@end

Our Controller inherits from UIViewController. UIKit supports MVC primarily through this class.

Our Controller inherits from UIViewController. UIKit supports MVC primarily through this class.
#import <UIKit/UIKit.h>

@interface CalculatorViewController : UIViewController

IBOutlet UILabel *display;

- (IBAction)digitPressed:(UIButton *)sender;

- (IBAction)operationPressed:(UIButton *)sender;

@end
#import <UIKit/UIKit.h>

@interface CalculatorViewController : UIViewController

IBOutlet UILabel *display;

CalculatorBrain *brain;

- (IBAction)digitPressed:(UIButton *)sender;

- (IBAction)operationPressed:(UIButton *)sender;

@end
#import <UIKit/UIKit.h>

@interface CalculatorViewController : UIViewController
{
    CalculatorBrain * brain;
    IBOutlet UILabel * display;
}

- (IBAction)digitPressed:(UIButton *)sender;
- (IBAction)operationPressed:(UIButton *)sender;
@end
CalculatorViewController.xib
“File’s Owner” is our Controller

CalculatorViewController.xib
My First Project

A picture (or demo) is worth 1,000 words.