# Objective-C Language

Tutor : Michael

# Hello Obj-C

```
#import <Foundation/Foundation.h>

int main(int argc, const char * argv []){
        NSAutoreleasePool* pool =
[[NSAutoreleasePool alloc ] init];

        NSLog(@"hello obj-c");

        [pool drain];
        return 0;
}
```

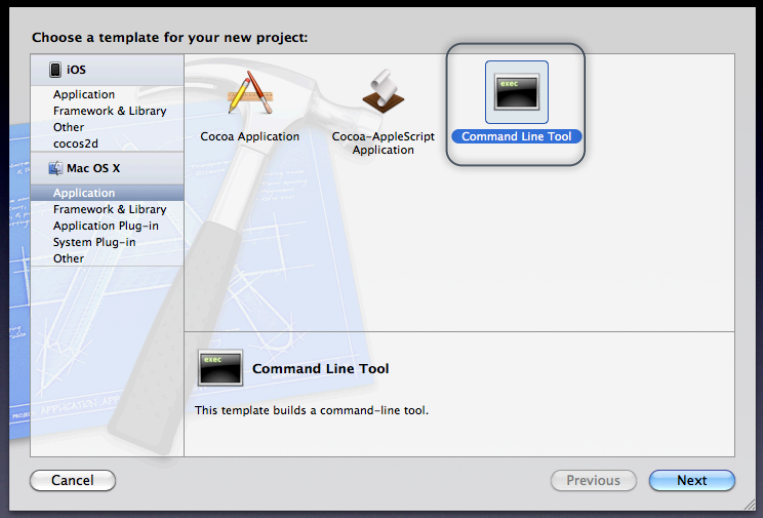
# 在Terminal執行

1. gcc -framework Foundation xxx.m -o prog
2. ./prog

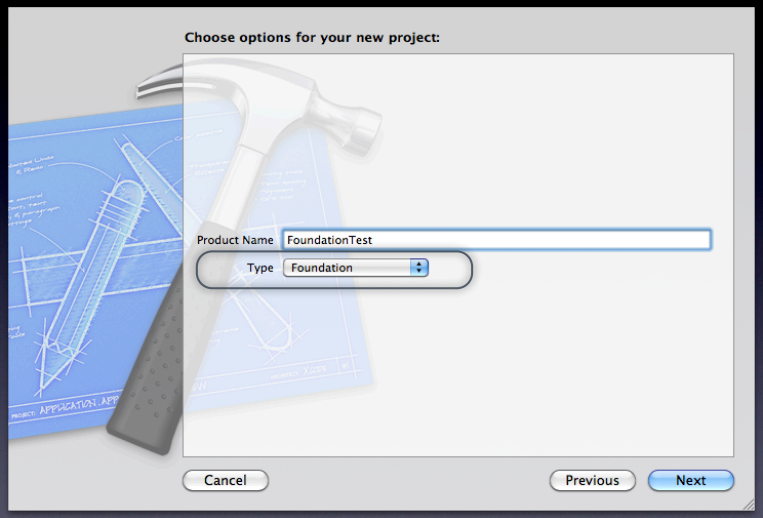
# First two things

- **NSLog()** is like printf() in C
- Use obj-c type **@"string"** instead of c type "string"

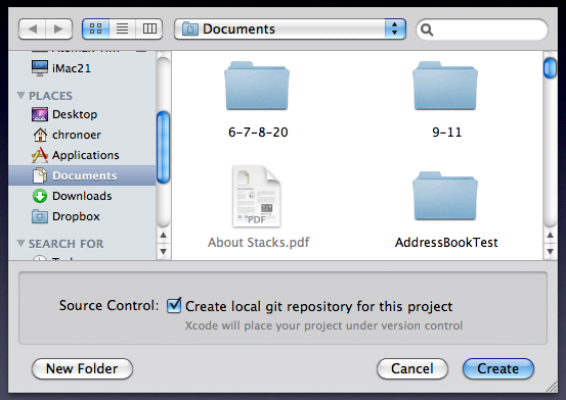
# Start from XCode



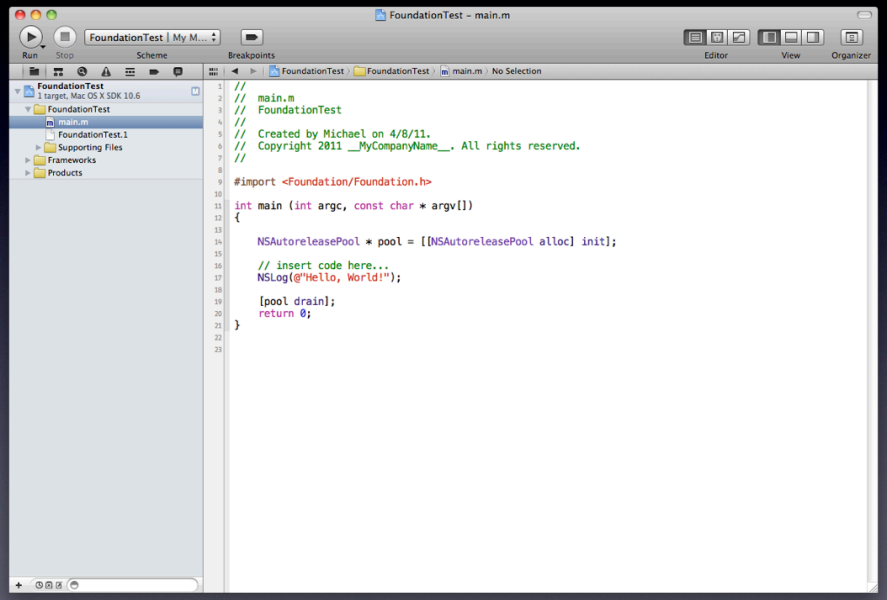
# Foundation


Choose options for your new project:
Product Name
FoundationTest
Type
Foundation
Cancel
Previous
Next

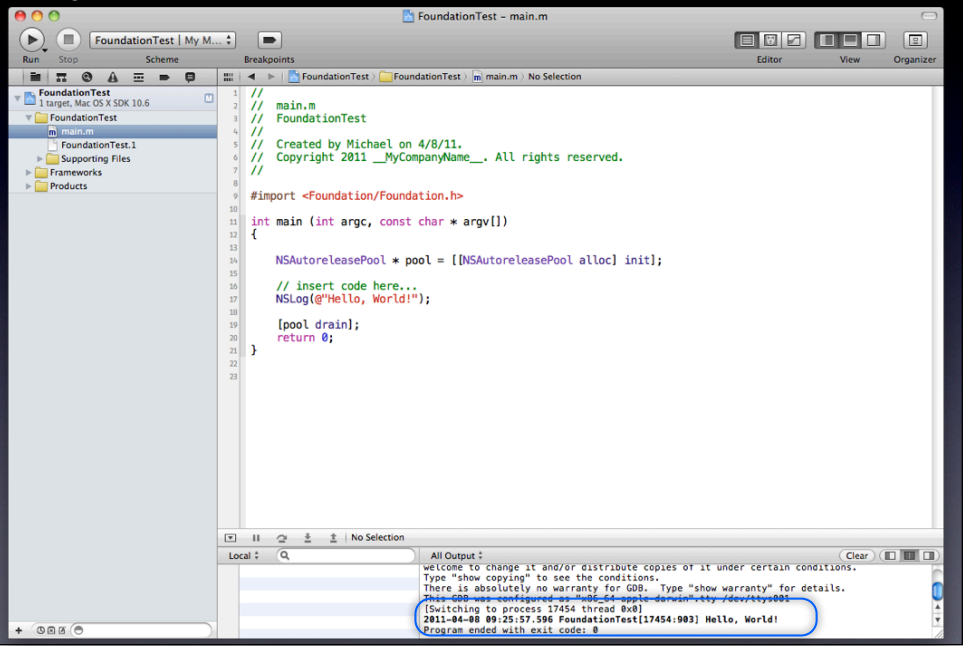
# Create Project



# Select Main file



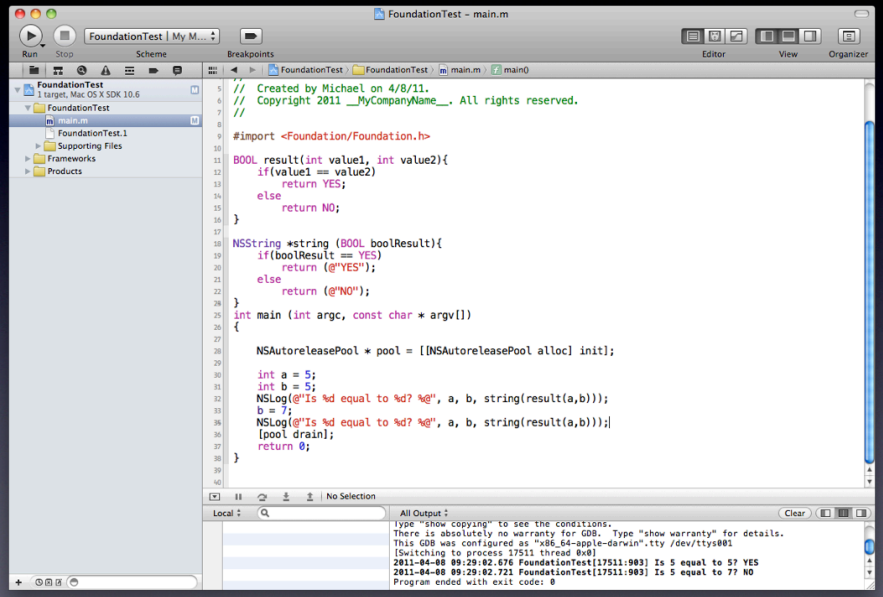
# Run


FoundationTest – main.m
FoundationTest | My M...
Run
Stop
Scheme
Breakpoints
Editor
View
Organizer
FoundationTest
1 target, Mac OS X SDK 10.6
FoundationTest
main.m
FoundationTest.1
Supporting Files
Frameworks
Products
FoundationTest › FoundationTest › main.m › No Selection
//
// main.m
// FoundationTest
//
// Created by Michael on 4/8/11.
// Copyright 2011 __MyCompanyName__. All rights reserved.
//
#import <Foundation/Foundation.h>
int main (int argc, const char * argv[])
{
NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
// insert code here...
NSLog(@"Hello, World!");
[pool drain];
return 0;
}
No Selection
Local
All Output
Clear
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
[Switching to process 17454 thread 0x0]
2011-04-08 09:25:57.596 FoundationTest[17454:903] Hello, World!
Program ended with exit code: 0

# Function call

```
#import <Foundation/Foundation.h>

BOOL result(int value1, int value2){
   if(value1 == value2)
      return YES;
   else
      return NO;
}

NSString *string (BOOL boolResult){
   if(boolResult == YES)
      return (@"YES");
   else
      return (@"NO");
}

int main (int argc, const char * argv[]) {
   int a = 5;
   int b = 5;
   NSLog(@"Is %d equal to %d? %@", a, b, string(result(a,b)));
   b = 7;
   NSLog(@"Is %d equal to %d? %@", a, b, string(result(a,b)));
   return 0;
}
```

# Result

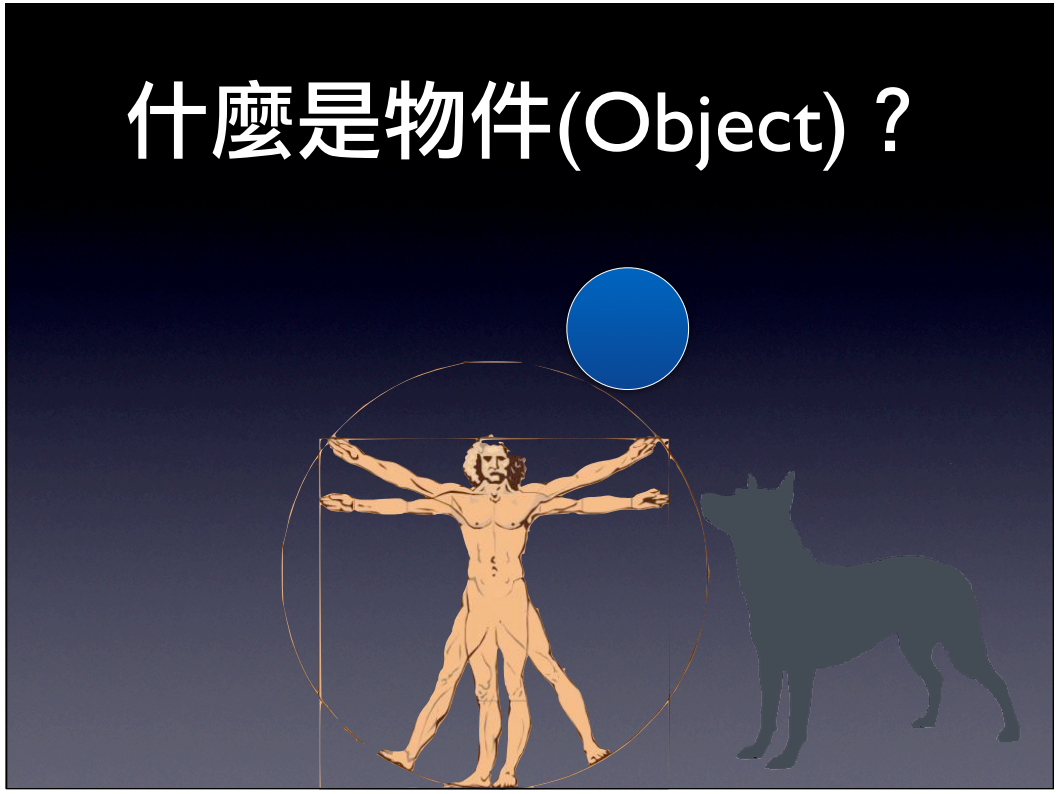

FoundationTest – main.m
FoundationTest | My M...
Run
Stop
Scheme
Breakpoints
Editor
View
Organizer
FoundationTest
1 target, Mac OS X SDK 10.6
FoundationTest
main.m
FoundationTest.1
Supporting Files
Frameworks
Products
FoundationTest › FoundationTest › main.m › main()
// Created by Michael on 4/8/11.
// Copyright 2011 __MyCompanyName__. All rights reserved.
//
#import <Foundation/Foundation.h>
BOOL result(int value1, int value2){
if(value1 == value2)
return YES;
else
return NO;
}
NSString *string (BOOL boolResult){
if(boolResult == YES)
return (@"YES");
else
return (@"NO");
}
int main (int argc, const char * argv[])
{
NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
int a = 5;
int b = 5;
NSLog(@"Is %d equal to %d? %@", a, b, string(result(a,b)));
b = 7;
NSLog(@"Is %d equal to %d? %@", a, b, string(result(a,b)));
[pool drain];
return 0;
}
No Selection
Local
All Output
Clear
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "x86_64-apple-darwin".tty /dev/ttys001
[Switching to process 17511 thread 0x0]
2011-04-08 09:29:02.676 FoundationTest[17511:903] Is 5 equal to 5? YES
2011-04-08 09:29:02.721 FoundationTest[17511:903] Is 5 equal to 7? NO
Program ended with exit code: 0

# Recap

```
#import <Foundation/Foundation.h>

void modifiedPointer(int * a){
   int * b;
   b =a ;
   *b = 3;
}

int main (int argc, const char * argv[]) {
  NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];

  int * a;
  int b = 9 ;
  a = &b;

   modifiedPointer(a);
   NSLog(@"Result is %d", *a);
  [pool drain];
  return 0;
}
```

Result is ?

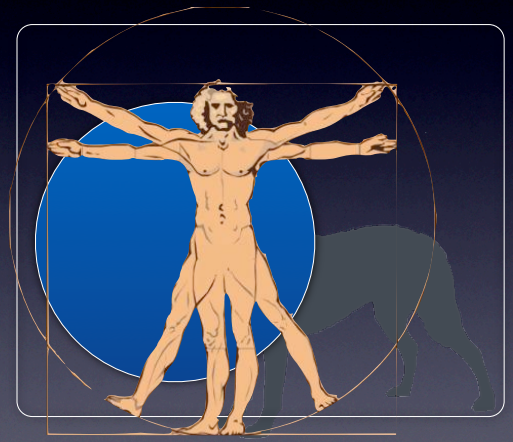
# 什麼是物件(Object)？



# 任何東西都是

# 來看看物件的特色

# 任一個物件都有



名字

屬性

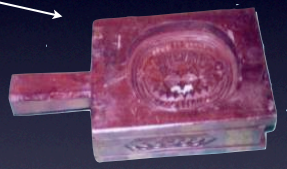
行為

# 什麼是Class(類別)

一切都要從創世紀開始說明

# 話說第六日，神要造烏龜

拿



拍!打!

吹氣

# 簡化來看

Class

Object

以上故事純屬虛構

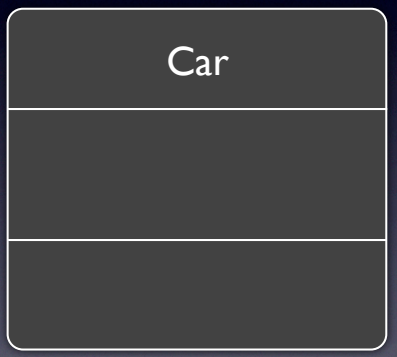
# Class

- 是個抽象的概念
- 是個模版
- 用來製造object
- 如果要製造大量相同的object只需要一種class

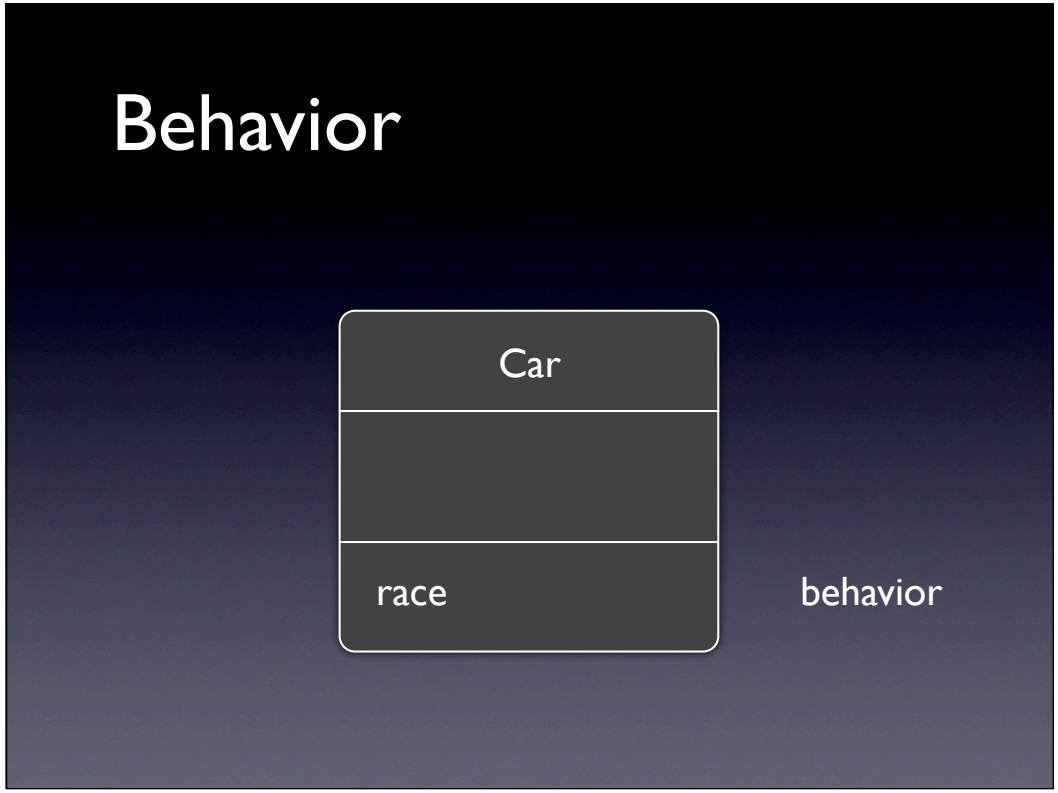
# 紅龜粿模板不也是Object？

- 是的
- Class 指的是用來製造烏龜的紋路

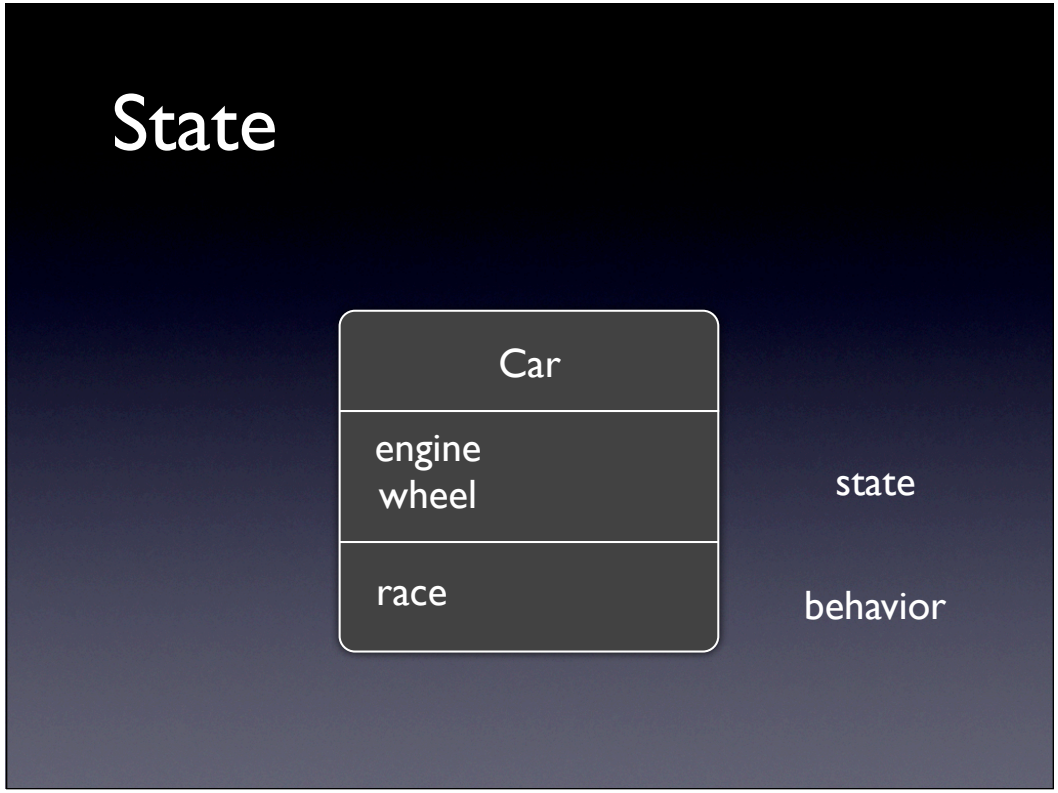
# Object



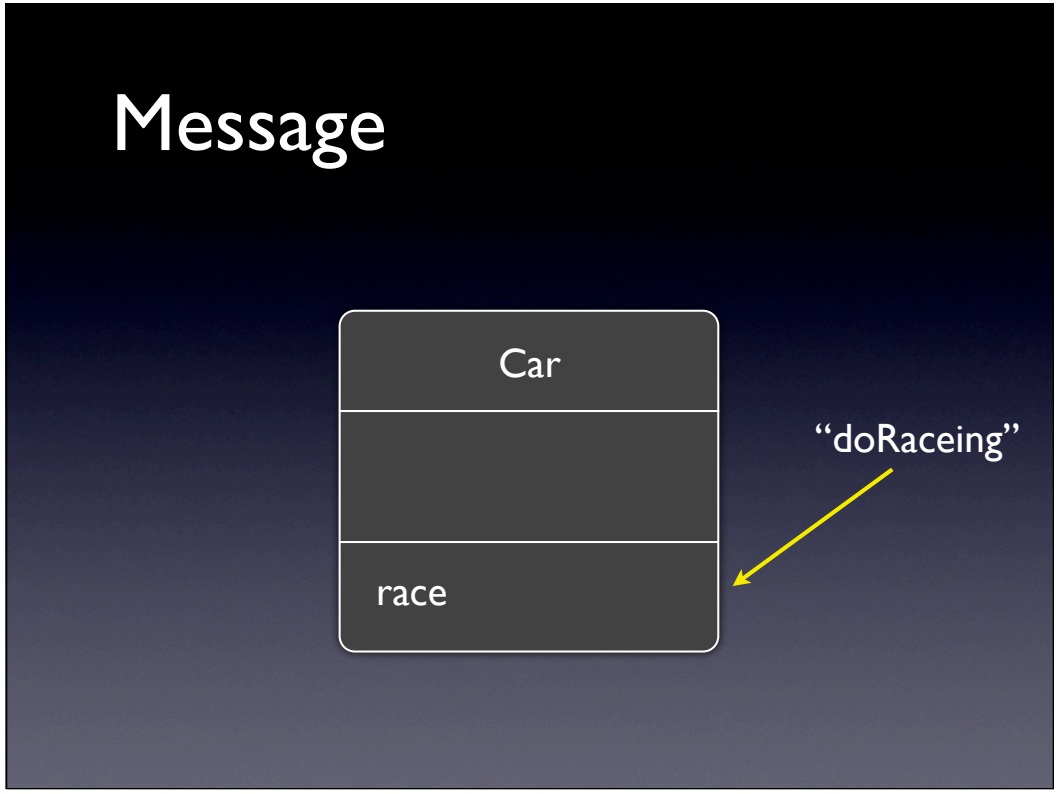
# Behavior



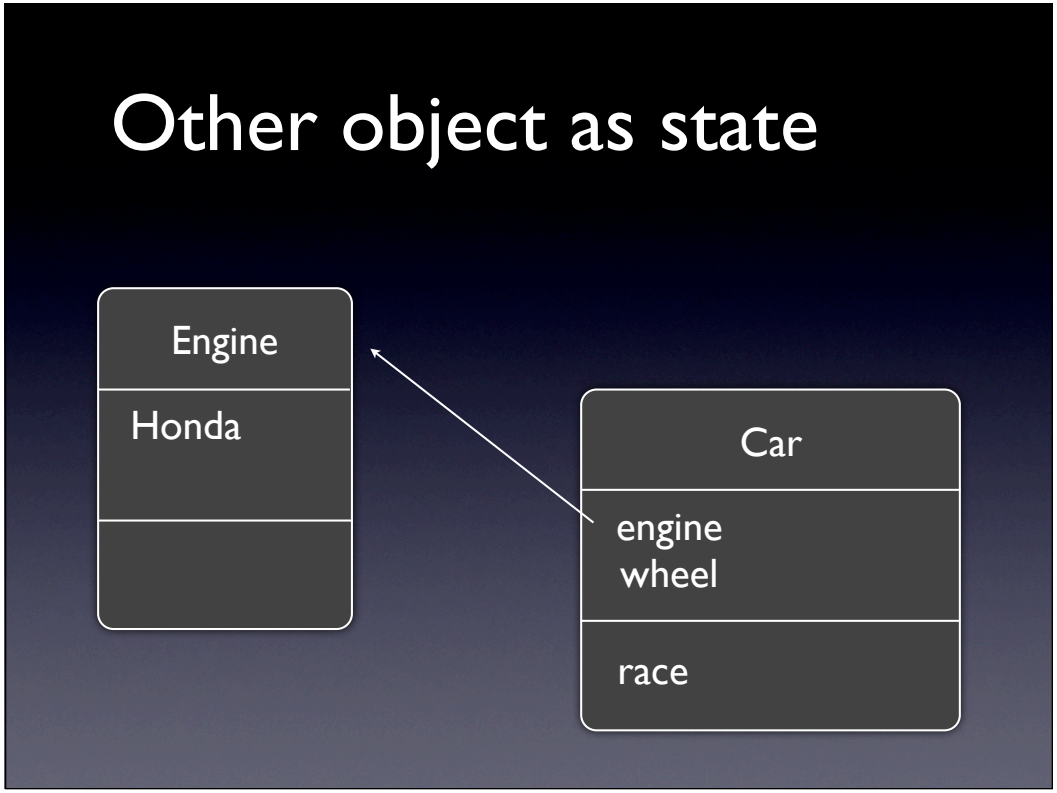
# State



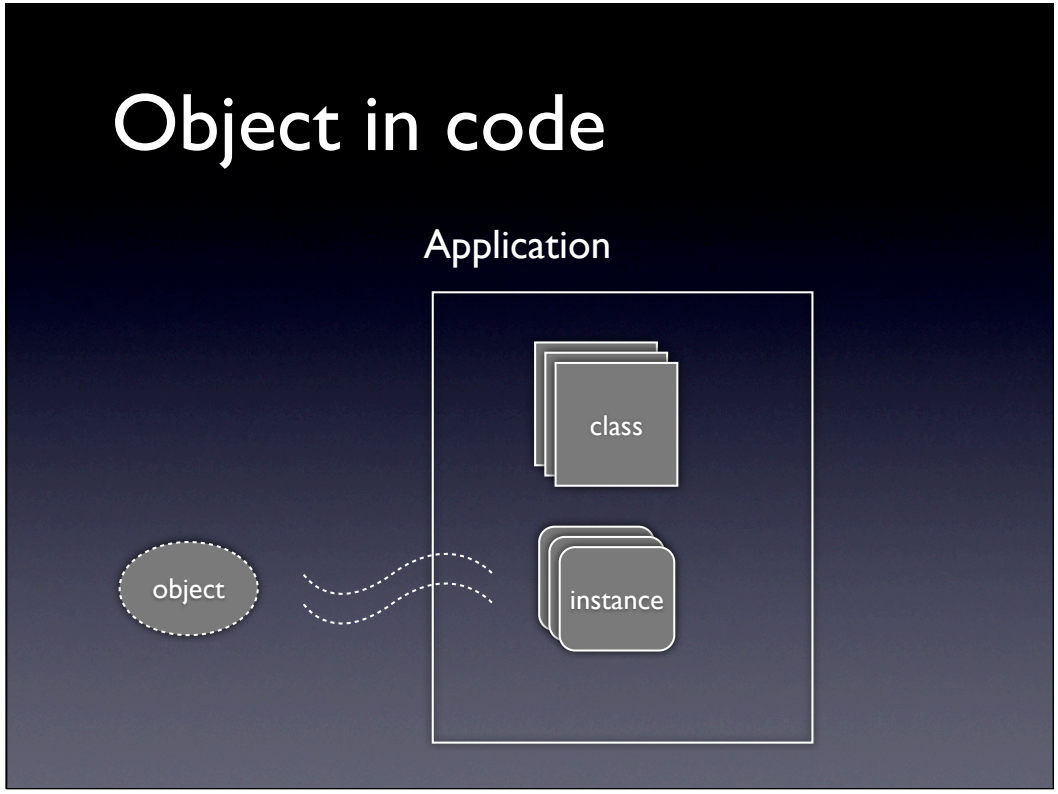
# Message



# Other object as state



# Object in code



# 名詞整理

Class - 類別

Object - instance, 物件, 實體

- Name - 名字
- State - attribute, member, 屬性,狀態,成員
- Behavior - method, action,行為, 方法

# Class & Instance in C

class - 類別
instance - 實體

```
struct BookInfo {
   int rating;
   char title[titleLength];
   char author[authorLength];
};
```

類別

```
int main (int argc, const char * argv[]) {

  struct BookInfo myBook;

  }
```

實體

# 物件概念延申

int a ;

// a 也是個物件，只有名字叫整數

struct BookInfo b ;

// b 也是個物件，名字和屬性

物件導向語言：
就是用語法把屬性 (int a;) 和 function整合在一起，或者說把 c 的 struct 加上 function

# **Class** in Objective-C

宣告 - interface

實作 - implementation

還記得寫程式，就是定義？Class？Object？

# Class - 宣告

```
@interface Shape : NSObject {
    int color;
}
- (void) setColor: (int) Color;
- (void) draw;
@end
```

名字

狀態

行為

參數型別

參數名

```
- (void) setColor: (ShapeColor) Color withNum:(int) aNum;
```

# Class - 定義

```
@implementation Shape
- (void) setColor: (int) myColor {
  color = myColor;
}

@end
```

定義行為內容

# Function - 宣告與定義

```
void myPlus(int a, int b);
```

宣告

```
int main (int argc, const char * argv[]) {

    myPlus(4, 5);

}
```

呼叫

```
void myPlus(int a, int b){
    printf("a + b is %d\n", a+b);
}
```

定義

# Method - 宣告與定義

```
@interface Shape : NSObject {
  ShapeColor color;
}
- (void) setColor: (ShapeColor) Color;
- (void) draw;
@end
```

宣告

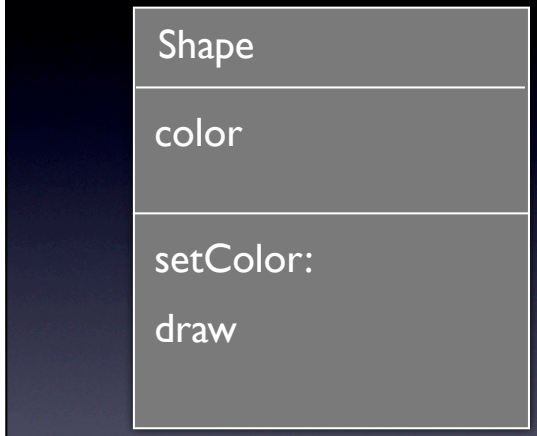
```
int main (int argc, const char * argv[]) {
  Shape * s = [Shape new];
  [s setColor:blue];
  [s draw];
}
```

呼叫

```
@implementation Shape
- (void) setColor: (ShapeColor) myColor {
  color = myColor;
}
@end
```

定義

# Class Diagram



# Recap - My Class

```
typedef enum {
   red,
   green,
   blue,
}ShapeColor;


@interface Shape : NSObject {
   ShapeColor color;
}
- (void) setColor: (ShapeColor) Color;
- (void) draw;
@end


@implementation Shape
- (void) setColor: (ShapeColor) myColor {
   color = myColor;
}

- (void) draw {
    NSLog(@"draw some shape %d", color);
}
@end
```

# Recap - Usage

```
#import <Foundation/Foundation.h>

// 上一頁寫在這
int main (int argc, const char * argv[]) {
   NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];


  Shape * s = [Shape new];
  [s setColor:blue];
  [s draw];

  [pool drain];
  return 0;
}
```

Result is ?

# Objective-C method & C function

Obj-C style : infix notation

```
[s setColor: yellow ]

[someObj setTitle: @"someTitle" setKey:556677 ]
```

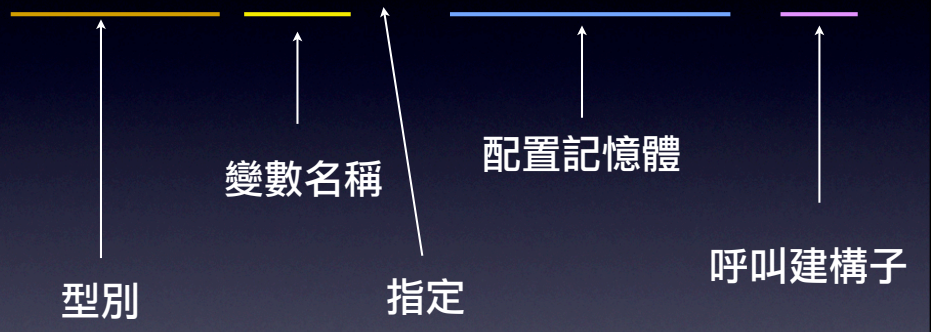
Method name - (setTitle:setKey:)

C style

```
setTitleAndKey(someObj, @"someTitle", 556677);
```

Function name - setTitleAndKey

## Variable - 宣告定義

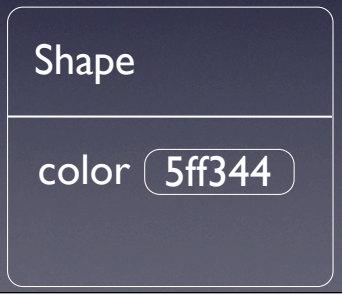



[[NSString alloc ]init] 等同於[NSString new]

[aObj something] 代表的意思是傳something這個訊息給aObj

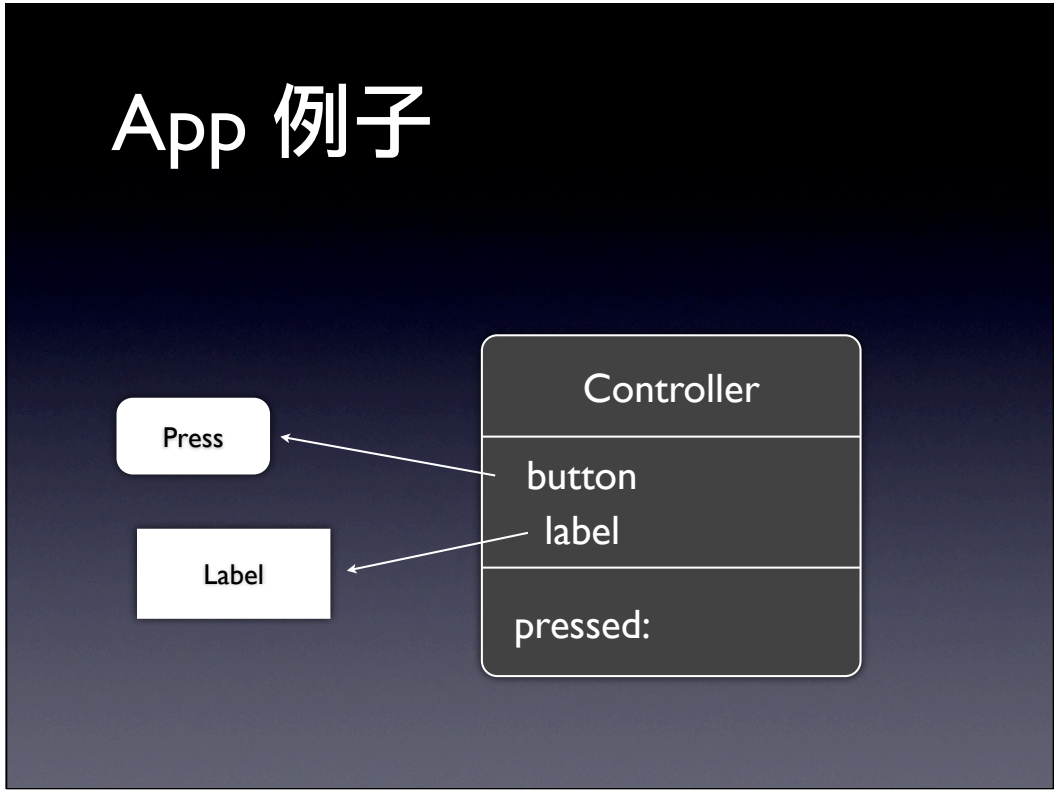
# 配置記憶體與初始化


初始化

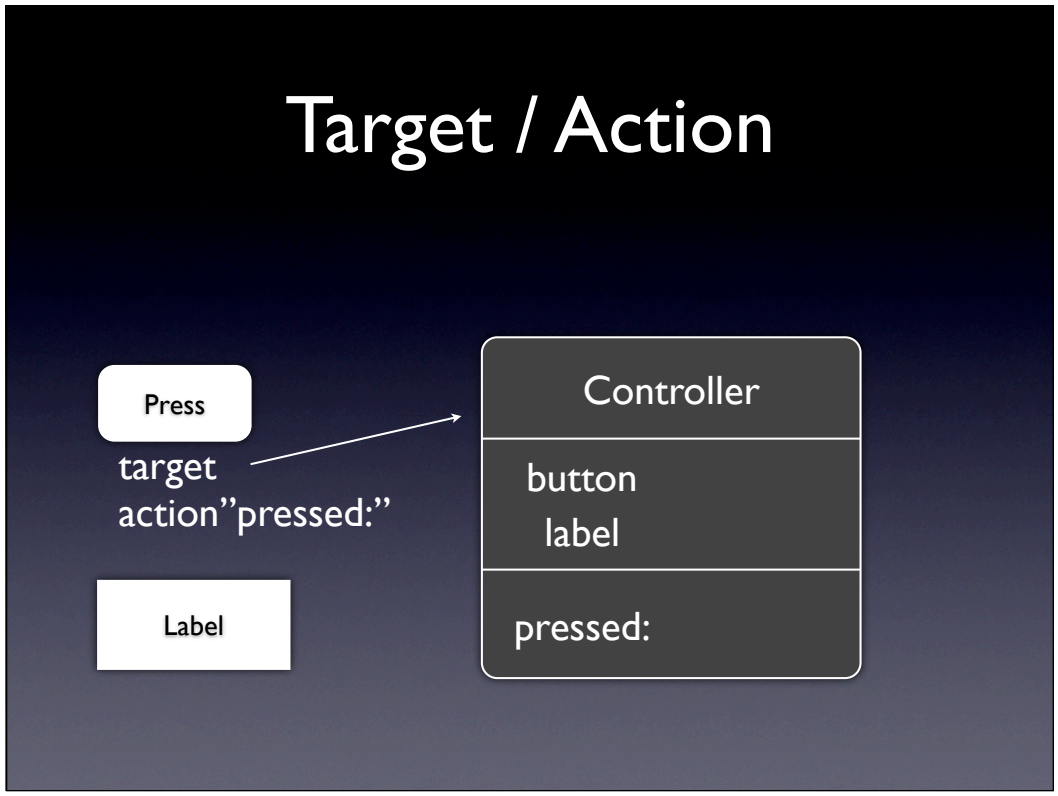



初始化的行為寫在建構子，
建構子也是一個 method

# App 例子



# Target / Action



# Demo HelloXcode

# Basic Terms

**Class** : Object 的類型，包含很多資料

**Instance** : Class 的實體, 常也叫做object

**Method** : 定義Object要如何去動作

**Instance Variable (ivar)**: 屬於Object的一些特別的資料

# OO Terms

Encapsulation

隱藏實作的部分和介面有所區隔

Polymorphism

有許多不同的object但有著相同的介面

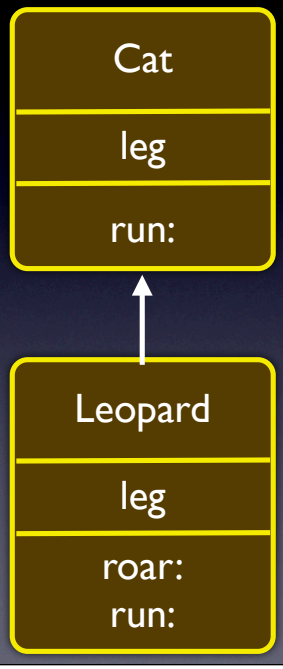
Inheritance

hierarchical organization, share code, customize or extend behaviors

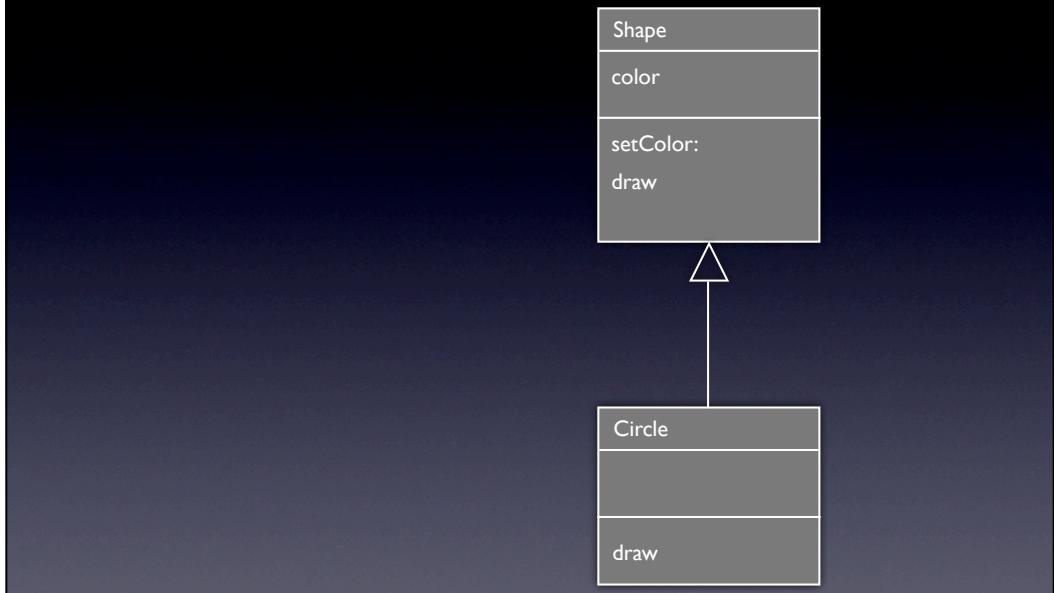
# Inheritance

- Subclass inherits states and behaviors from superclass

- Subclass may augment, replace the superclass methods



# Class Diagram



# Recap - Inheritance

```
typedef enum {
   red,
   green,
   blue,
}ShapeColor;

@interface Shape : NSObject {
   ShapeColor color;
}
- (void) setColor: (ShapeColor) Color;
- (void) draw;
@end


@implementation Shape
- (void) setColor: (ShapeColor) myColor {
   color = myColor;
}

- (void) draw {
NSLog(@"draw some shape %d", color);
}

@end
```

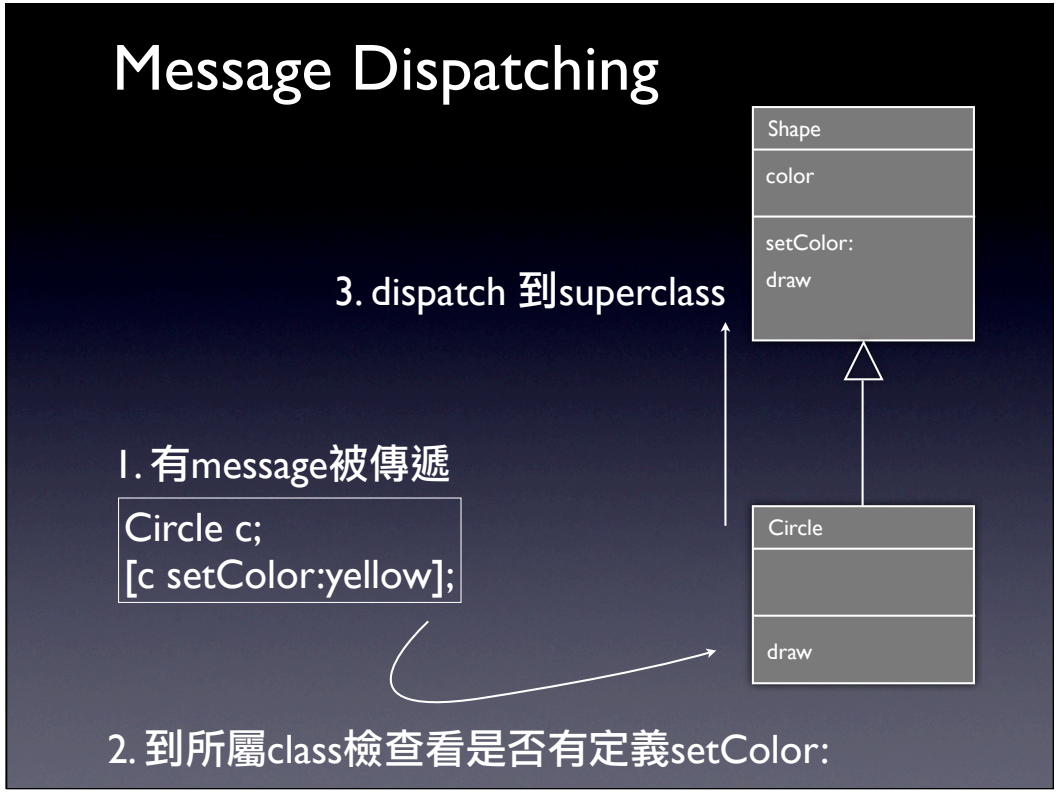
```
@interface Circle : Shape

@end

@implementation Circle
- (void) draw {
  NSLog (@"Drawing circle... color is %d", color);
}
@end
```

# Recap - Inheritance

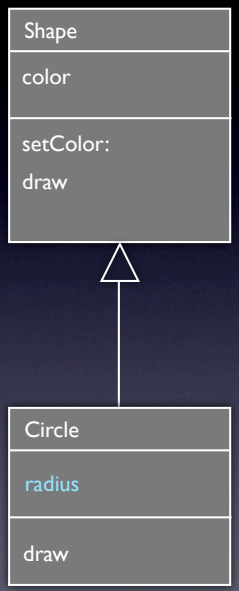
```
// 上一頁寫在這裡
int main (int argc, const char * argv[]) {
    NSAutoreleasePool * pool = [[NSAutoreleasePool
alloc] init];


  Circle * c = [Circle new];
  [c setColor:green];
  [c draw];

  [pool drain];
  return 0;
}
```

Result ?

# Message Dispatching



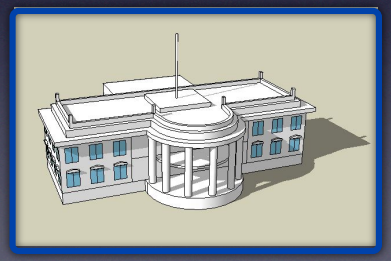
# Inherited Instance




Memory Layout of Circle object

# Class & Instance

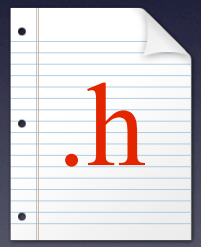
Blueprint



White house

# Declare & Implement

Hello.h




Hello.m

# Hello.h

```
@interface Hello : NSObject {

 int age;

}
-(void) greeting:(NSString *) words;
@end
```

# Hello.m

```
#import “Hello.h”
@implementation Hello


-(void) greeting:(NSString *) words{
 NSLog(words);
}
@end
```

# Message Syntax

# Message Syntax

```
[object signal]

[object signal:argument]

[object signal:arg1 withOther:arg2]
```

# Message Syntax - More

```
[object signal]
```

// 和object說要溝通的訊息是signal

```
[object signal:argument]
```

// 和object說要溝通的訊息是signal並傳了argument這個資料給object

```
[object signal:arg1 withOther:arg2]
```

// 傳了arg1和arg2兩筆資料給object
// 此時訊息叫signal:withOther:

全部要溝通的訊息都定義在object裡所以object會知道

# Terminology

**Message Expression**

```
[object signal:argument]
```

**Message**

```
[object signal:argument]
```

Receiver

**Selector**

```
[object signal:argument]
```

**Method**

message 所選擇的codes

# Examples of Message

```
Consumer *customer; //假設存在

[customer buyOnLine];

int card = [customer creditCard];

if([customer isCardValid]){
 //確認卡號合法
}

[customer buyItem:@"Macbook" quantity:1];

NSString *address = [[customer bill] address];
```

# Class & Instance Method

實體呼應instance method

```
- (id) init;  // e.g. [[Hello alloc ] init ]
- (void) greeting:(NSString *) word;
```

類別呼應class method

```
+ (id) alloc;   // e.g. [Hello alloc]
+ imageNamed:(UIImage) image;
// [UIImage imageNamed:@"hello.png"]
```

# Class & Instance Example

```
[NSString  string];
 // string is  +/- ?
[@"500" intValue ];
 // intValue is  +/- ?
[[NSArray alloc ] initWithObject:@"fine"];
 // initWithObject: is  +/- ?
[NSArray arrayWithObject:@"ok"];
 // arrayWithObject: is  +/- ?
```

# Convenient Way

- Since Objective-C 2.0
- Dot Syntax

```
NSString *name = person.name;
//         name = [person name]

     person.name = @"Michael";
//  [person setName:@"Michael"]
```

- Cascade

```
      person.bill.name = @"Andy";
// [[person bill] setName:@"Andy"]
      person.bill.name ;
// [[person bill] name]
```

# Demo ModifyClass

Getter & Setter

# Types

# Dynamic & Static

- Dynamic

```
id someObject; // 任何的型別
```

```
id * - usually use id only
```

- Static

```
Customer *person;
```

- Objective-C checks type at compile-time, usually uses run-time binding

# Null pointer to Object

Check exsiting

```
if(person != nil){}
if(!person){}
```

Usage

```
person = nil; // assignment
[child setFather:nil]; // as argument
```

# Boolean Type

Declare

```
BOOL lock;
```

Usage

```
lock = YES; // true
lock = NO;  // false
if(lock != YES)
if(lock)
if(!lock)
```

# Selector

- choose method by name
- Type Define

```
SEL sel;
```

- Usage

```
SEL sel = [object work];
SEL sel = @selector(work:);
SEL sel = @selector(work:who:);
[object setAction:@selector(work:)];
```

# Object works with selector

```
id obj;
SEL action = @selector(work);
```

- Check response

```
if([obj responseToSelector:action])
```

- Perform Selector

```
[obj performSelector:action];
// [obj work]
[obj performSelector:action withObject:who];
```

# Demo Selector

performSelector

# Operation

# With Class

- Feature

```
Class me = [obj class];
NSString *name = [obj className];
```

- Relationship

```
if([obj isKindOfClass:[UIView class]]){}
// if obj inherit UIView
if([obj isMemberOfClass:[NSString class]]){}
// if obj is the instance of NSString
```

# With Object

- Check identity

```
if(obj1 == obj2){
    // the same object instance
}
```

- Check equality

```
if([obj1 isEqual: obj2]){
    // the same content, define by user
}
```

# Overwrite -description

**- (NSString *) description;**

- Usage

```
NSLog([@"test " stringByAppendingFormat:@"%@", obj]);

// call [obj description];
```

- User can overwrite description to get the specific information

```
NSLog([obj description]);
```

# Numeric Type

# 依執行環境選擇

- 32-bit
- 64-bit

- NSInteger (int or long)
- NSUInteger
- CGFloat (float or double)

Question?