# 继承

## 继承的基本用法

* 设计两个类Bird、Dog

// Bird的声明

@interface Bird : NSObject

{

@public

int weight;

}

- (void)eat;

@end

// Bird的定义

@implementation Bird

- (void)eat {

NSLog(@"吃吃吃-体重：%d", weight);

}

@end

// Dog的声明

@interface Dog : NSObject

{

@public

int weight;

}

- (void)eat;

@end

// Dog的定义

@implementation Dog

- (void)eat {

NSLog(@"吃吃吃-体重：%d", weight);

}

@end

* 有相同的属性和行为，抽出一个父类Animal（先抽取weight属性，再抽取eat方法）

// Animal的声明

@interface Animal : NSObject

{

@public

int weight;

}

- (void)eat;

@end

// Animal的定义

@implementation Animal

- (void)eat {

NSLog(@"吃吃吃-体重：%d", weight);

}

@end

* 子类在父类的基础上拓充属性和方法

// Bird的声明

@interface Bird : Animal

{

@public

int height;

}

- (void)fly;

@end

// Bird的定义

@implementation Bird

- (void)fly {

NSLog(@"飞飞飞-高度：%d", height);

}

@end

// Dog的声明

@interface Dog : Animal

{

@public

int speed;

}

- (void)run;

@end

// Dog的定义

@implementation Dog

- (void)run {

NSLog(@"跑跑跑-高度：%d", speed);

}

@end

* 子类方法和属性的访问过程：如果子类没有，就去访问父类的
* 父类被继承了还是能照常使用的
* 父类的静态方法
* 画继承结构图，从子类抽取到父类
* NSObject的引出：全部OC类的最终父类，包含了一些常用方法，比如+new

## 继承的专业术语

* 父类\超类 superclass
* 子类 subclass\subclasses

## 继承的细节

* 单继承
* 子类和父类不能有相同的成员变量
* 方法的重写

## super关键字

* 分别调用父类的对象方法和类方法

## 继承的好处

* 不改变原来模型的基础上，拓充方法
* 建立了类与类之间的联系
* 抽取了公共代码
* 坏处：耦合性强

## 继承的使用场合

* 它的所有属性都是你想要的，一般就继承
* 它的部分属性是你想要的，可以抽取出另一个父类

# 多态

## 多态的基本概念

* 某一类事物的多种形态
* OC对象具有多态性

## 多态的体现

Person \*p = [Student new];

p->age = 100;

[p walk];

* 子类对象赋值给父类指针
* 父类指针访问对应的属性和方法

## 多态的好处

* 用父类接收参数，节省代码

## 多态的局限性

* 不能访问子类的属性（可以考虑强制转换）

## 多态的细节

* 动态绑定：在运行时根据对象的类型确定动态调用的方法

# NSString的简单使用

## 字符串的快速创建

NSStirng \*str = @“Hello”;

## 使用静态方法创建

## 使用%@输出字符串

NSString \*name = @”mj”;

NSLog(@“我的名字是%@”, name);