# 简单使用

## 基本用途

* 可以用来声明一大堆方法（不能声明成员变量）
* 只要某个类遵守了这个协议，就相当于拥有这个协议中的所有方法声明
* 只要父类遵守了某个协议，就相当于子类也遵守了

## 格式

* 协议的编写

@protocol 协议名称

// 方法声明列表

@end

* 某个类遵守协议

@interface 类名 : 父类 <协议名称>

@end

## 关键字

协议中有2个关键字可以控制方法是否要实现（默认是@required），在大多数情况下，用途在于程序员之间的交流

* @required：这个方法必须要实现（若不实现，编译器会发出警告）
* @optional：这个方法不一定要实现

## 协议遵守协议

* 一个协议可以遵守其他多个协议，多个协议之间用逗号 , 隔开
* 一个协议遵守了其他协议，就相当于拥有了其他协议中的方法声明

@protocol 协议名称 <协议1, 协议2>

@end

## 基协议

* NSObject是一个基类，最根本最基本的类，任何其他类最终都要继承它
* 其实还有一个协议，名字也叫NSObject，它是一个基协议，最根本最基本的协议
* NSObject协议中声明很多最基本的方法，比如description、retain、release等
* 建议每个新的协议都要遵守NSObject协议

## 定义变量时指定协议

// NSObject类型的对象，并且要遵守NSCopying协议

NSObject<NSCopying> \*obj;

// 任何OC对象，并且要遵守NSCoding协议

id<NSCoding> obj2;

# 代理设计模式

## 设计原理

* 有些麻烦的事情不想自己亲自做，就可以找个人帮忙做，即交给代理对象去做

## 设计原则

* 首先得拥有某个代理对象属性
* 其次要很清楚代理有哪些方法
* 最后要保证能解耦

## 实现方案

* 定义一个protocol，在其中声明一些和代理沟通的方法
* 拥有一个代理属性id<protocol> delegate
* 让代理遵守protocol