```
平时在对象.h/.m文件添加属性
                                                          和方法时,其实就是扩展
                                                说明:
                                                          比如我们继承了UIView, 我们@interface XXX ()里面添加属性
                                                          和方法就是为了扩展其功能,满足我们页面功能多样的需求
                                                                     私有(.m默认私有)
                                                        开放程度:
                                                                     公有(.h默认公用)
                                                                     通过设置@private 私有
                                                                 对象属性
                                                                             @property()
                       类扩展(Class Extension)
                                                                                       .h文件定义方法
                                                                 对象方法
                                                                             -()xxx
                                                        类型
                                                                                       .m实现方法
                                                                                      .h文件定义方法
                                                                 类方法
                                                                           + ()xxx
                                                                                      .m实现方法
                                                         一般 .h定义 的方法一旦调
                                                                                 除非运用运行时做消
                                                注意:
                                                         用, .m必须实现该方法
                                                                                 息转发,不然会崩溃!
                                                          当扩展越多,协调开发越频繁,
                                                                                     此时分类方法就能更好的解决
                                                          代码就会越臃肿,耦合度高,难
                                                                                     这个问题
                                                          以维护,代码冗余
                                                                                     加入需要扩展一些功能,在一个
                                                    系统类文件我们只能继承, 不能直接修改
                                                                                     地方修改,其他所有子类均能使
                                                                                     用就好了
                                          说明:
                                                                                              默认不能添加对象方法和属性
                                                    分类的创建, 默认我们添加新的类方法 让所有子类使
                                                    用,而无需引入头文件(等于给系统文件添加类方法)
                                                                                              可以通过运行时变相实现
                                                   分类能更好的解耦
                                                   Category 是表示一个指向分类的结构体的指针,其定义如下:
                                                   typedef struct objc_category *Category;
                                                   struct objc_category {
                                                    char *category_name
                                                                                 OBJC2_UNAVAILABLE; // 分类名
                                          定义
                                                    char *class name
                                                                                OBJC2_UNAVAILABLE; // 分类所属的类名
                                                    struct objc_method_list *instance_methods OBJC2_UNAVAILABLE; // 实例方法列表
                                                    struct objc_method_list *class_methods OBJC2_UNAVAILABLE; // 类方法列表
                                                    struct objc_protocol_list *protocols
                                                                                  OBJC2_UNAVAILABLE; // 分类所实现的协议列表
                                                               @interface 新扩展的名称(分类的名称)
                                                               @end
                                                   分类格式
                                                               @implementation 新扩展的名称(分类的名称)
                                                               @end
                                                               @interface Programmer (Category)
                                                   分类方法:
                                                                + (void) programCategoryMethod; //分类方法
                                                                @end
                                                                    @interface Programmer (Category)
                                                                    @property(nonatomic,copy) NSString *name;
                                                   属性和对象方法:
                                                                    - (void) programCategoryMethod; //对象方法
                                                                    @end
                                                                    分类中的可以写@property,
                                                                                               运行直接调用会导致崩溃
                                                                    但不会生成setter/getter方法
                                                                 如果分类中有和原有类同名的方法, 会
                                                                                                   iOS只会校验方法名称,不校验
                                                                 优先调用分类中的方法,覆盖原类方法
                                                                                                   参数名称及类型
                                                   类方法重名:
                                                                 如果多个分类中都有和原有类中同名的方法,那么调用该方法的时候执行谁由编译
                                                                 器决定;编译器会执行最后一个参与编译的分类中的方法
                                          使用:
                                                                   分类声明了属性,但是系统不会
                                                                                             那么手动添加setter/getter方法,达到
分类(category)类扩展
                                                                   为其生成setter/getter方法
                                                                                              使用点语法获取和修改属性的目的
    (extension)
                                                                            1.属性: @property(nonatomic,copy) NSString *name;
                                                                            2.实现setter/getter方法(只能点语法,_name不能用):
                                                                            #import <objc/runtime.h>
                                                                            //定义一个key值
                                                                            static NSString *nameWithSetterGetterKey = @"nameWithSetterGetterKey";
                                                                            @implementation Programmer (Category)
                                                                            //运行时实现setter方法
                       分类(Category):
                                                                            – (void)setName:(NSString *)nameWithSetterGetter {
                                                                   代码:
                                                   动态添加属性:
                                                                                objc_setAssociatedObject(self, &nameWithSetterGetterKey, name,
                                                                            OBJC_ASSOCIATION_COPY);
                                                                            //运行时实现getter方法
                                                                            - (NSString *)name {
                                                                              return objc_getAssociatedObject(self, &nameWithSetterGetterKey);
                                                                            @end
                                                                             set/get实现中objc_setAssociatedObject 和 objc_getAssociatedObject
                                                                   提问?
                                                                             含义是什么?
                                                       iOS中有一种功能是关联对象
                                                                                 我们知道,运行时方法能转发消息给其他对象,同样也可以绑定对象
                                                                                        objc_setAssociatedObject(id object, const void *key, id value, objc_AssociationPolicy policy)
                                                                               绑定:
                                                                                        通过key把对象value关联到对象object上
                                                                                        objc_getAssociatedObject(id object, const void *key)
                                                                               获取:
                                                                                         通过key把关联到object上的对象value取出来
                                                       objc_setAssociatedObject
                                                                                        使用绑定的方法objc_setAssociatedObject()
                                                                               解绑:
                                                                                        id value 处传值nil,使得该绑定无效,也即是解绑
                                                                                                   objc_removeAssociatedOjbects
                                                                               移除绑定(一般不用):
                                                                                                   会移除对象所有绑定事件,如果是button则会无法点击等等
                                                                          在UITableView中点击某一个cell,这时候弹出一个
                                                                 需求:
                                                                                                                           把毫不相干的两
                                                                          UIAlertView,然后在UIAlertView消失的时候获取此cell的信
                                                                                                                           个对象关联起来
                                                                          息,我们就获取cell的indexPath
                                                                                  #import <objc/runtime.h>
                                                                                  static char kUITableViewIndexKey;
                                          关联对象:
                                                                                   - (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
                                                                                  UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"提示"
                                                                          绑定:
                                                                                                            message:@"这里是xx楼"
                                                                                                            delegate:self
                                                       实例:
                                                                                                       cancelButtonTitle:@"好的"
                                                                                                       otherButtonTitles:nil];
                                                                                   //然后这里设定关联,此处把indexPath关联到alert上
                                                                1代码:
                                                                                  objc_setAssociatedObject(alert, &kUITableViewIndexKey, indexPath, OBJC_ASSOCIATION_RETAIN_NONATOMIC);
                                                                                  [alert show];
                                                                                   – (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex
                                                                                     if (buttonIndex == 0) {
                                                                          获取:
                                                                                     NSIndexPath *indexPath = objc_getAssociatedObject(alertView, &kUITableViewIndexKey);
                                                                                     NSLog(@"%@", indexPath);
```