该文章引用自 http://www.cnblogs.com/ziyi--caolu/p/4864160.html
原型模式：用原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象。原型模式其实就是从一个对象再创建另一个可定制的对象，而且不需知道任何创建的细节。(也是引用改文章的定义)
一、 原型模式的使用：
在iOS中使用原型模式依赖使用 NSCopying 协议。
我们自定义一个person类，有name，age，sex，year ，和可以添加朋友的方法。
在.h 文件中的定义：

```
#import <Foundation/Foundation.h>
@interface Person :NSObject
- (instancetype)initWithName:(NSString *)name age:(NSString *)age sex:(NSString *)sex year:(NSString *)year;
- (void)addFriend:(Person *)person;
@end
```

在.m文件中定义：

```
#import "Person.h"
@interface Person () <NSCopying>
{
    NSMutableArray *_friends;
}
@property (nonatomic) NSString *name;
@property (nonatomic) NSString *age;
@property (nonatomic) NSString *sex;
@property (nonatomic) NSString *year;
@end

@implementation Person
- (instancetype)initWithName:(NSString *)name age:(NSString *)age sex:(NSString *)sex year:(NSString *)year
{
    if (self = [super init]) {
        self.name = name;
        self.age = age;
        self.sex = sex;
        self.year = year;
        _friends = [NSMutableArray array];
    }
    return self;
}
- (void)addFriend:(Person *)person
{
    [_friends addObject:person];
}
- (id)copyWithZone:(nullable NSZone *)zone
{
    Person *person = [[Person class] allocWithZone:zone];
    person->_friends = [_friends mutableCopy];
    return person;
}
@end
```

在viewController 中调用person类创建对象：

```
    Person *person1 = [[Person alloc] initWithName:@"张三" age:@"17" sex:@"男" year:@"1991-10-10"];
    Person *person2 = [[Person alloc] initWithName:@"李四" age:@"22" sex:@"女" year:@"1988-05-10"];
    [person1 addFriend:person2];
    // 如果现在有person3 和person1除了姓名不同外 其他属性都一样 这个时候通过使用原型模式会非常方便
    Person *person3 = [person1 copy];
    // 这个时候使用copy 可以直接通过person1对象获取到一个属性值一样的一个独立的person3对象 不需要我们手动去重新写一遍初始化方法。
```

二、 引申出这样一个问题（面试题）：怎样使用copy关键字？ (文章作者提出的问题和答案 我自己写代码验证了下)

一般使用retain或者strong修饰属性时，是使引用对象的指针指向同一个对象，即为同一块内存地址。只要其中有一个指针变量被修改时所有其他引用该对象的变量都会被改变。

而copy关键字修饰时，如果新的对象是不可变的，那么它是直接引用新对象的内存地址，并不重新分配内存地址，如果新对象是可变的，那么在赋值时是释放旧对象，拷贝新对象内容。重新分配了内存地址。以后该指针变量被修改时就不会影响旧对象的内容了。

copy只有实现NSCopying协议的对象类型才有效。
常用于NSString和Block。

验证实例：

```
    NSMutableArray *arr1 = [[NSMutableArray alloc] initWithObjects:@"hello", @"nihao", nil];
    NSMutableArray *copyArr = [arr1 copy];

    [arr1 addObject:@"add"];
    NSLog(@"%@ -- %p, copyArr %@--- %p", arr1, arr1, copyArr, copyArr);
    // (hello, nihao, add ) -- 0x7f9c63725f00, copyArr (hello, nihao)--- 0x7f9c63737560
    // 地址不一样了 因为为可变对象
    NSArray *array1 =@[@"haha", @"smile"];
    NSArray *array2 = [array1 copy];
    NSLog(@"%p, %p", array1, array2);
    //    0x7f9c6370dcd0, 0x7f9c6370dcd0
    // 地址一样 因为都是不可变的对象。
```

作者在最后写道：朋友们，虽然这个世界日益浮躁起来，只要能够为了当时纯粹的梦想和感动坚持努力下去，不管其它人怎么样，我们也能够保持自己的本色走下去。

新增一个面试题： 出自:http://www.cocoachina.com/ios/20150803/12872.html
这个写法会出什么问题： **@property (copy) NSMutableArray *array;**
两个问题：
1、添加,删除,修改数组内的元素的时候,程序会因为找不到对应的方法而崩溃.因为copy就是复制一个不可变NSArray的对象；
2、使用了atomic属性会严重影响性能。