3、动态交换两个方法的实现(method Swizzing)

```
Method method1 = class_getInstanceMethod([ViewController class], @selector(replaceMethod1));
Method method2 = class_getInstanceMethod([ViewController class], @selector(replaceMethod2));
method_exchangeImplementations(method1, method2);

[self replaceMethod1];
[self replaceMethod2];
```

4、动态实现 NSCoding 的自动归档、解档

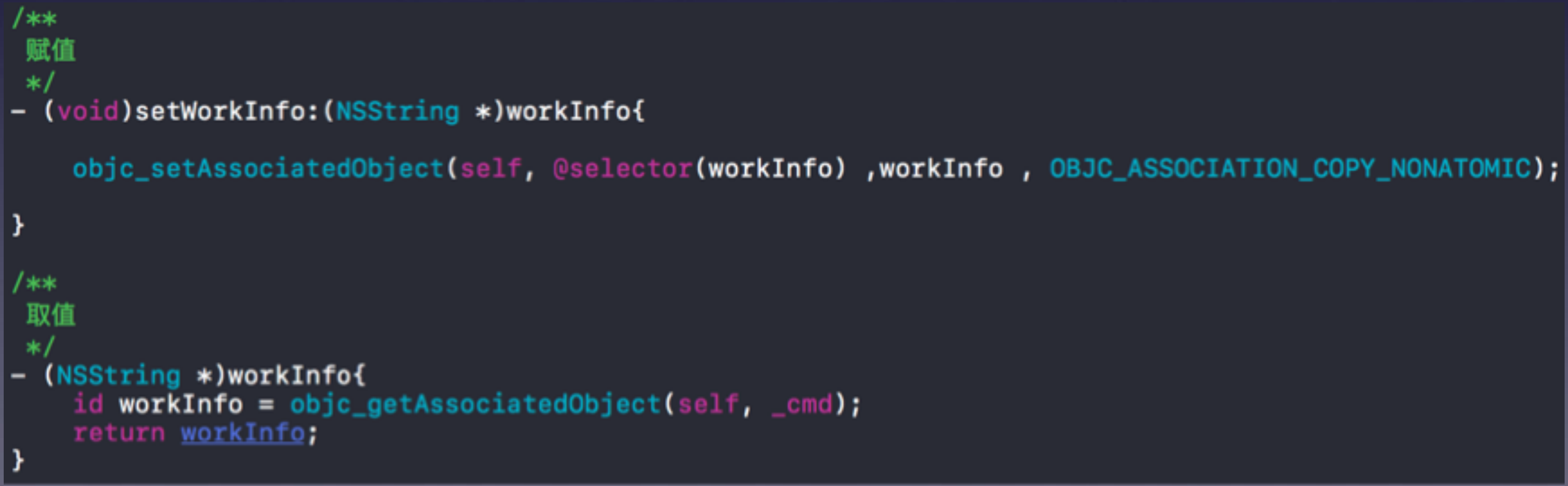
其实就是在获取属性或者实例变量的基础上使用kvc赋值

```
- (void)encodeWithCoder:(NSCoder *)aCoder{

    unsigned int propertysCount;
    objc_property_t *propety =  class_copyPropertyList([self class], &propertysCount);
    for (int i = 0; i < propertysCount; i++) {
        const char *name = property_getName(propety[i]);
        NSString *propertyName = [NSString stringWithUTF8String:name];
        [aCoder setValue:[self valueForKeyPath:propertyName] forKeyPath:propertyName];
    }
}
```

5、动态为Category添加属性

一般在使用 Category 时，对已经存在的类(主要是系统类)进行扩展时，一般只能添加实例、类方法，而无法直接添加属性，即使在.h中声明一个 property ，在.m中也不会生成对应的实例变量和相关属性访问方法。这时通过 runtime 来动态绑定属性。具体操作如下:



```
/**
 赋值
 */
- (void)setWorkInfo:(NSString *)workInfo{

    objc_setAssociatedObject(self, @selector(workInfo) ,workInfo , OBJC_ASSOCIATION_COPY_NONATOMIC);

}

/**
 取值
 */
- (NSString *)workInfo{
    id workInfo = objc_getAssociatedObject(self, _cmd);
    return workInfo;
}
```