成员变量

```
const char *name = "_height";
size_t size = sizeof(float);
unsigned char uint8_t = log2(sizeof(float));
bool isOk = class_addIvar([Person class], name, size, uint8_t, @encode(float));
NSLog(@"%d",isOk);
```

注意:在OC中不支持往已存在的类中添加实例变量，所以不管是系统的类还是自定的类都是无法添加的，但是，如果通过运行时创建的类，还是可以通过该方法来添加的。

方法

```
IMP methodImp = class_getMethodImplementation([ViewController class], @selector(printTest));
bool isOK = class_addMethod([Person class], @selector(printTest), methodImp, "v@:");
NSLog(@"%d",isOK);
Person *p1 = [[Person alloc] init];
[p1 performSelector:@selector(printTest)];
```

3、动态交换两个方法的实现(method Swizzing)

```objc
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

```objc
- (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];
    }
}
```