



冰凌天 Lv2

2018年08月03日 阅读 166

[关注](#)

## 小码哥iOS学习笔记第二天: OC对象的分类

- Objective-C中的对象, 简称OC对象, 主要可以分为3种
  - `instance` 对象(实例对象)
  - `class` 对象(类对象)
  - `meta-class` 对象(元类对象)

### 一、instance

- `instance` 对象就是通过类 `alloc` 出来的对象, 每次调用 `alloc` 都会产生新的 `instance` 对象

ObjectiveC

```
NSObject *obj1 = [[NSObject alloc] init];
NSObject *obj2 = [[NSObject alloc] init];

NSLog(@"obj1 - %p", obj1);      // 打印: obj1 - 0x100500e90
NSLog(@"obj2 - %p", obj2);      // 打印: obj2 - 0x1005061b0
```

- `obj1`、`obj2` 是 `NSObject` 的 `instance` 对象(实例对象), 它们是不同的两个对象, 分别占据着两块不同的内存
- instance对象在内存中存储的信息包括
  - `isa` 指针
  - 其他 成员变量
- 例如, 一个 `Person` 类, 继承自 `NSObject`, 有一个 `_age` 的成员变量

ObjectiveC

```
@interface Person: NSObject {
    @public
    int _age;
}

@end
```

```
@implementation Person
@end
```

- 创建 `Person` 类的 `instance` 对象, 并给成员变量 `_age` 赋值

ObjectiveC

```
Person *p1 = [[Person alloc] init];
p1->_age = 3;

Person *p2 = [[Person alloc] init];
p2->_age = 4;
```

- 因为 `p1` 和 `p2` 是两个 `alloc` 创建的对象, 所以在内存中分别占用一块内存, 类似于下图



## 二、class

- `class` 类对象, 可以通过 `alloc` 创建出 `instance` 对象
- 有三种方式, 可以获取一个类对象
  - `-(Class)class`
  - `+(Class)class`
  - `object_getClass(实例对象)`

ObjectiveC

```
NSObject *obj1 = [[NSObject alloc] init];
NSObject *obj2 = [[NSObject alloc] init];
// -(Class)class
Class objectClass1 = [obj1 class];
Class objectClass2 = [obj2 class];
// +(Class)class
Class objectClass3 = [NSObject class];
// object_getClass(实例对象)
Class objectClass4 = object_getClass(obj1);
Class objectClass5 = object_getClass(obj2);
```

- `objectClass1 ~ objectClass5` 都是 `NSObject` 的 `class` 对象 (类对象)

- 它们是同一个对象, 每个类在内存中有且只有一个 `class` 对象, 可以通过打印内存证明:

ObjectiveC

```
NSLog(@"%p %p %p %p %p",
      objectClass1,
      objectClass2,
      objectClass3,
      objectClass4,
      objectClass5);

// 打印结果: 0x7fff955aa140 0x7fff955aa140 0x7fff955aa140 0x7fff955aa140 0x7fff955aa140
```

- 通过打印结果可以证明, `objectClass1 ~ objectClass5` 指向这同一块内存地址, 即在内存中会存在一个类的 `class` 对象
- `class`对象在内存中存储的信息主要包括:
  - `isa` 指针
  - `superclass` 指针
  - 类的 `属性` 信息(@property)
  - 类的 `对象方法` 信息(instance method)
  - 类的 `协议` 信息(protocol)
  - 类的 `成员变量` 信息(ivar, 描述成员变量的类型和名字, 而不是如同实例一般具体的值)
  - ...
- 如下图:



### 三、 meta-class

- 每个类在内存中有且只有一个 meta-class 对象
- 可以通过运行时的 object\_getClass(类对象) 方法获取类的元类型

ObjectiveC

```
// Runtime API  
Class objectMetaClass = object_getClass([NSObject class]);
```

- objectMetaClass 是 NSObject 的 meta-class 对象(元类对象)
- meta-class 对象和 class 对象的内存结构是一样的，但是用途不一样，在内存中存储的信息主要包括：
  - isa 指针
  - superclass 指针
  - 类的 类方法 信息(class method)



注意: meta-class对象 和 class对象 拥有相同的结构, 意思如下图



class对象 中，类方法信息 为空，meta-class 方法中 属性信息 、 对象方法信息 、 协议信息 、 成员变量信息 为空

注意: - (Class)class 和 + (Class)class 方法只能获取 class对象 , 不能获取 meta-class对象

meta-class对象 只能通过 Runtime 的 object\_getClass(类对象) 获取

- 可以通过 Runtime 的 class\_isMetaClass(对象) 函数, 来判断对象是否是 元类型

ObjectiveC

```
Class objectClass = [NSObject class];
NSLog(@"%d", class_isMetaClass(objectClass));           // 打印: 0

Class objectMetaClass = object_getClass([NSString class]);
NSLog(@"%d", class_isMetaClass(objectMetaClass));       // 打印: 1
```

- 下面是 objc\_getClass 、 object\_getClass 、 - (Class)class、 + (Class)class 的区别

ObjectiveC

```
1.Class objc_getClass(const char *aClassName)
1> 传入字符串类名
2> 返回对应的类对象
```

2. `Class object_getClass(id obj)`

1> 传入的obj可能是instance对象、**class**对象、**meta-class**对象

2> 返回值

a) 如果是instance对象，返回**class**对象

b) 如果是**class**对象，返回**meta-class**对象

c) 如果是**meta-class**对象，返回**NSObject**（基类）的**meta-class**对象

3. `-(Class)class`、`+(Class)class`

1> 返回的就是类对象

```
- (Class) {  
    return self->isa;  
}
```

```
+ (Class) {  
    return self;  
}
```

关注下面的标签，发现更多相似文章

iOS

Objective-C



冰凌天 **Lv2** iOS

获得点赞 189 · 获得阅读 9,688

关注

## 安装掘金浏览器插件

打开新标签页发现好内容，掘金、GitHub、Dribbble、ProductHunt 等站点内容轻松获取。快来安装掘金浏览器插件获取高质量内容吧！

评论

输入评论...