## 类

## 8.1 类的概念

Swift中的类和结构体一样都是人们构建代码所用的一种通用且灵活的构造体。我们可以使用相似的语法规则来为类和结构体定义属性和添加方法，从而拓展类和结构体的功能。上一章，我们介绍了结构体，这一章我们就来详细介绍功能更加强大的类。

在Swift中，用“类”来描述“对象”，所谓的对象是指现实世界中的一切事物，那么类就可以看做是对相似事物的抽象。只说概念可能是很抽象的，让我们来举个实际的例子

假如你要买一台电脑，你会考虑哪些因素？按照常见思维，我们首先会想，电脑有哪些部件？电脑应该能完成那些工作？我们会想到电脑有型号，CPU，显卡，内存，主板，屏幕......，然后电脑可以看电影，浏览网页，玩游戏......。随后你打开电商网站，选了一台这样的电脑，它的型号是ABC，CPU是i7，显卡是 gtx1060，内存是16g等等，2天之后你收到了新的电脑，你很开心。在这次购物过程中，按照面向对象的思维如何理解呢？什么是类？什么是对象？“类”就是你心中的的电脑，你知道一台电脑应该有哪些部件，应该有哪些功能。它是高度抽象的，而你买的型号为ABC的电脑则是一个”对象“，它是按照”电脑“这个抽象概念制造出来的，换句话说对象（型号为ABC的电脑）是类（”电脑“）的一个实例，是具体的。电脑的CPU，显卡等等称作属性，而能完成的工作称作行为，也就是方法。

从这里可以看出，类和结构体是极为相似的，都拥有属性，都可以定义方法。但两者也是有区别的，例如类是引用类型，而结构体是值类型。我们为你简单归纳了一下两者的区别：

|  |  |  |
| --- | --- | --- |
| 属性特征 | 类 | 结构体 |
| 定义属性用于存储值 | √ | √ |
| 定义方法用于提供功能 | √ | √ |
| 定义下标脚本用于允许使用下标语法访问值 | √ | √ |
| 定义初始化器用于初始化状态 | √ | √ |
| 可以被拓展来实现默认没有的功能 | √ | √ |
| 遵循协议来针对特定类型提供标准功能 | √ | √ |
| 继承允许一个类继承另一个类的特征 | √ |  |
| 类型转换允许你在运行时检查和解释一个类的实例类型 | √ |  |
| 反初始化器允许一个实例释放任何被其占用的资源 | √ |  |
| 引用计数器允许不止一个对类的实例的引用 | √ |  |

## 8.2 类的创建