The Swift Programming Language in Chinese

https://github.com/letsswift

http://letsswift.com/

June 15, 2014

目录

1 Swift 中文教程（十三）继承

一个类可以从另外一个类中继承方法，属性或者其它的一些特性。当一个类继承于另外一个类时，这个继承的类叫子类，被继承的类叫父类。继承是 Swift 中类区别于其它类型的一个基本特征。

Swift 中的类可以调用父类的方法，使用父类的属性和下标，还可以根据需要使用重写方法或者属性来重新定义和修改他们的一些特性。Swift 可以帮助你检查重写的方法和父类的方法定义是相符的。

类还可以为它继承的属性添加观察者，这样可以能够让它在一个属性变化的时候得到通知。属性观察者可以被添加给任何属性，不管它之前是存储属性还是计算属性。

1.1 定义一个基类

任何一个不继承于其它类的类被称作基类

> 注意：Swift 的类不是从一个全局基类继承而来。在你编写代码的时，只要是在类的定义中没有继承自父类的类都是基类。

下面的例子定义了一个叫 Vehicle 的基类。基类包含两个所有交通工具通用的属性 numberOfWheels 和 maxPassengers。这两个属性被一个叫 description 的方法使用，通过返回一个 String 描述来作为这个交通工具的特征：

```
class Vehicle {
    var numberOfWheels: Int
    var maxPassengers: Int
    func description() -> String {
     return "\(numberOfWheels) wheels; up to \(maxPassengers) passengers"
    }
    init() {
        numberOfWheels = 0
        maxPassengers = 1
    }
}
```

这个交通工具类 Vehicle 还定义了一个构造函数来设置它的属性。构造函数更多的解释在 Initialization 一章，但是为了说明子类如何修改继承的属性，这里需要简要解释一下什么叫构造函数。

通过构造函数可以创建一个类型的实例。尽管构造函数不是方法，但是它们在编码的时候使用了非常相似的语法。构造函数通过确保所有实例的属性都是有效的来创建一个新的实例。

构造函数最简单的形式是使用 init 关键词的一个类似方法的函数，并且没有任何参数：

```
init() {
    // perform some initialization here
}
```

使用构造函数语法 TypeName 和空的两个小括号来完成一个 Vehicle 实例的创建：

```
let someVehicle = Vehicle()
```

Vehicle 的构造函数为属性设置了一些初始值 (numberOfWheels = 0 然后 maxPassengers = 1)。

Vehicle 类定义的是一个通用的交通工具特性，它本身没有太多意义，所以就需要冲定义它的一些属性或者方法来让它具有实际的意义。

1.2 产生子类

产生子类就是根据一个已有的类产生新类的过程。子类继承了父类的一些可以修改的特性。还可以为子类添加一些新的特性。

为了表明一个类是继承自一个父类，需要将父类的名称写在子类的后面，并且用冒号分隔：

```
class SomeClass: SomeSuperclass {
    // class definition goes here
}
```

下面的例子定义了一种特定叫 Bicycle 的交通工具。这个新类是基于已有的类 Vehicle 产生的。书写方式是在类名 Bicycle 后加冒号加父类 Vehicle 名。

可以理解为：

定义一个新的类叫 Bicycle，它继承了 Vehicle 的特性：

```
class Bicycle: Vehicle {
    init() {
        super.init()
        numberOfWheels = 2
    }
}
```

Bicycle 是 Vehicle 的子类，Vehicle 是 Bicycle 的父类。Bicycle 类继承了 Vehicle 所有的特征，比如 maxPassengers 和 numberOfWheels 属性。你还可以为 Bicycle 类添加心的属性。

Bicycle 类也定义了构造函数，在这个构造函数中调用了父类的构造函数 super.init()，这样可以确保在 Bicycle 修改他们之前，父类已经初始化了。

> 注意：跟 Objective-C 不同的是，Swift 中的构造函数没有默认继承。
> 更多信息可以参考 Initializer Inheritance and Overriding 这一章节。

maxPassengers 属性在继承自父类的时候已经被初始化了，对于 Bicycle 来说是正确的，因此不需要再做更改。然后 numberOfWheels 是不对的，所以被替换成了 2.

不仅属性是继承于 Vehicle 的，Bicycle 还继承了父类的方法。如果你创建一个实例，然后调用了已经继承的 description 方法，可以得到该交通工具的描述并且看到它的属性已经被修改：

```
let bicycle = Bicycle()
println("Bicycle: \(bicycle.description())")
// Bicycle: 2 wheels; up to 1 passengers
```

子类本身也可以作为父类被再次继承：

```
class Tandem: Bicycle {
    init() {
        super.init()
        maxPassengers = 2
    }
}
```

上面的例子创建了 Bicycle 的子类，叫做 tandem，也就可以两个人一起骑的自行车。所以 Tandem 没有修改 numberOfWheels 属性，只是更新了 maxPassengers 属性。

注意：子类只能够在构造的时候修改变量的属性，不能修改常量的属性。

创建一个 Tandem 的实例，然后调用 description 方法检查属性是否被正确修改：

```
let tandem = Tandem()
println("Tandem: \(tandem.description())")
// Tandem: 2 wheels; up to 2 passengers
```

注意到 description 方法也被 Tandem 继承了。

1.3 重写方法

子类可以提供由父类继承来的实例方法，类方法，实例属性或者下标的个性化实现。这个特性被称为重写。

重写一个由继承而来的方法需要在方法定义前标注 override 关键词。通过这样的操作可以确保你所要修改的这个方法确实是继承而来的，而不会出现重写错误。错误的重写会造成一些不可预知的错误，所以如果如果不标记 override 关键词的话，就会被在代码编译时报错。

override 关键词还能够让 Swift 编译器检查该类的父类是否有相符的方法，以确保你的重写是可用的，正确的。

1.3.1 访问父类方法，属性和下标

当在重写子类继承自父类的方法，属性或者下标的时候，需要用到一部分父类已有的实现。比如你可以重定义已知的一个实现或者在继承的变量中存储一个修改的值。

适当的时候，可以通过使用 super 前缀来访问父类的方法，属性或者下标：

- 叫 someMethod 的重写方法可以在实现的时候通过 super.someMethod() 调用父类的 someMethod 方法。
- 叫 someProperty 的重写属性可以在重写实现 getter 或者 setter 的时候通过 super.someProperty 调用父类的 someProperty。
- 叫 someIndex 的重写下标可以在实现下标的时候通过 super[someIndex] 来访问父类的下标。

1.3.2 复写方法

你可以在你的子类中实现定制的继承于父类的实例方法或者类方法。

下面的例子演示的就是一个叫 Car 的 Vehicle 子类，重写了继承自 Vehicle 的 description 方法。

```
class Car: Vehicle {
    var speed: Double = 0.0
    init() {
        super.init()
        maxPassengers = 5
        numberOfWheels = 4
    }
    override func description() -> String {
        return super.description() + "; "
            + "traveling at \(speed) mph"
    }
}
```

Car 中定义了一个新的 Double 类型的存储属性 speed。这个属性默认值是 0.0，意思是每小时 0 英里。Car 还有一个自定义的构造函数，设置了最大乘客数为 5，轮子数量是 4.

Car 重写了继承的 description 方法，并在方法名 description 前标注了 override 关键词。

在 description 中并没有给出了一个全新的描述实现,还是通过 super.description 使用了 Vehicle 提供的部分描述语句，然后加上了自己定义的一些属性，如当前速度。

如果你创建一个 Car 的实例，然后调用 description 方法，会发现描述语句变成了这样:

```
let car = Car()
println("Car: \(car.description())")
// Car: 4 wheels; up to 5 passengers; traveling at 0.0 mph
```

1.3.3 复写属性

你还可以提供继承自父类的实例属性或者类属性的个性化 getter 和 setter 方法，或者是添加属性观察者来实现重写的属性可以观察到继承属性的变动。

1.3.4 重写属性的 Getters 和 Setters

不管在源类中继承的这个属性是存储属性还是计算属性，你都可以提供一个定制的 getter 或者 setter 方法来重写这个继承属性。子类一般不会知道这个继承的属性本来是存储属性还是计算属性，但是它知道这个属性有特定的名字和类型。在重写的时候需要指明属性的类型和名字，好让编译器可以检查你的重写是否与父类的属性相符。

你可以将一个只读的属性通过提那家 getter 和 setter 继承为可读写的，但是反之不可。

> 注意：如果你为一个重写属性提供了 setter 方法，那么也需要提供 getter 方法。如果你不想在 getter 中修改继承的属性的值，可以在 getter 中使用 super.someProperty 即可，在下面 SpeedLimitedCar 例子中也是这样。

下面的例子定义了一个新类 SpeedLimitedCar，是 Car 的一个子类。这个类表示一个显示在 40 码一下的车辆。通过重写继承的 speed 属性来实现：

```
class SpeedLimitedCar: Car {
    override var speed: Double  {
    get {
        return super.speed
    }
    set {
        super.speed = min(newValue, 40.0)
    }
    }
}
```

每当你要设置 speed 属性的时候，setter 都会检查新值是否比 40 大，二者中较小的值会被设置给 SpeedLimitedCar。

如果你尝试为 speed 设置超过 40 的值，description 的输出依然还是 40：

```
let limitedCar = SpeedLimitedCar()
limitedCar.speed = 60.0
println("SpeedLimitedCar: \(limitedCar.description())")
// SpeedLimitedCar: 4 wheels; up to 5 passengers; traveling at 40.0 mph
```

1.3.5 重写属性观察者

你可以使用属性重写为继承的属性添加观察者。这种做法可以让你无论这个属性之前是如何实现的，在继承的这个属性变化的时候都能得到提醒。更多相关的信息可以参考 Property Observers 这章。

> 注意：不能为继承的常量存储属性或者是只读计算属性添加观察者。这些属性值是不能被修改的，因此不适合在重写实现时添加 willSet 或者 didSet 方法。

> 注意：不能同时定义重写 setter 和重写属性观察者，如果想要观察属性值的变化，并且又为该属性给出了定制的 setter，那只需要在 setter 中直接获得属性值的变化就行了。

下面的代码演示的是一个新类 AutomaticCar，也是 Car 的一个子类。这个类表明一个拥有自动变速箱的汽车，可以根据现在的速度自动选择档位，并在 description 中输出当前档位：

```
class AutomaticCar: Car {
    var gear = 1
    override var speed: Double {
    didSet {
        gear = Int(speed / 10.0) + 1
    }
    }
    override func description() -> String {
        return super.description() + " in gear \(gear)"
    }
}
```

这样就可以实现，每次你设置 speed 的值的时候，didSet 方法都会被调用，来看档位是否需要变化。gear 是由 speed 除以 10 加 1 计算得来，所以当速度为 35 的时候，gear 档位为 4：

```
let automatic = AutomaticCar()
automatic.speed = 35.0
println("AutomaticCar: \(automatic.description())")
// AutomaticCar: 4 wheels; up to 5 passengers; traveling at 35.0 mph in gear 4
```

1.4 禁止重写

你可以通过标记 final 关键词来禁止重写一个类的方法，属性或者下标。在定义的关键词前面标注 @final 属性即可。

在子类中任何尝试重写父类的 final 方法，属性或者下标的行为都会在编译时报错。同样在扩展中为类添加的方法，属性或者下标也可以被标记为 final。

还可以在类关键词 class 前使用 @final 标记一整个类为 final(@final class)。任何子类尝试继承这个父类时都会在编译时报错。

参考文献