Swift 下标脚本 Swift 构造过程

Swift 继承

继承我们可以理解为一个类获取了另外一个类的方法和属性。

当一个类继承其它类时，继承类叫子类，被继承类叫超类（或父类)

在 Swift 中，类可以调用和访问超类的方法，属性和下标脚本，并且可以重写它们。

我们也可以为类中继承来的属性添加属性观察器。

基类

没有继承其它类的类，称之为基类（Base Class）。

以下实例中我们定义了基类 StudDetails ，描述了学生（stname）及其各科成绩的分数(mark1、mark2、mark3)：

```
class StudDetails {
    var stname: String!
    var mark1: Int!
    var mark2: Int!
    var mark3: Int!
    init(stname: String, mark1: Int, mark2: Int, mark3: Int) {
        self.stname = stname
        self.mark1 = mark1
        self.mark2 = mark2
        self.mark3 = mark3
    }
}
let stname = "swift"
let mark1 = 98
let mark2 = 89
let mark3 = 76

let sds = StudDetails(stname:stname, mark1:mark1, mark2:mark2, mark3:mark3);

print(sds.stname)
print(sds.mark1)
print(sds.mark2)
print(sds.mark3)
```

以上程序执行输出结果为：

```
swift
98
89
76
```

子类

子类指的是在一个已有类的基础上创建一个新的类。

为了指明某个类的超类，将超类名写在子类名的后面，用冒号(:)分隔,语法格式如下

```
class SomeClass: SomeSuperclass {
    // 类的定义
}
```

实例

以下实例中我们定义了超类 StudDetails，然后使用子类 Tom 继承它：

```
class StudDetails
{
    var mark1: Int;
    var mark2: Int;

    init(stm1:Int, results stm2:Int)
    {
        mark1 = stm1;
        mark2 = stm2;
    }

    func show()
    {
        print("Mark1:\(self.mark1), Mark2:\(self.mark2)")
    }
}

class Tom : StudDetails
{
    init()
    {
        super.init(stm1: 93, results: 89)
    }
}

let tom = Tom()
tom.show()
```

以上程序执行输出结果为：

```
Mark1:93, Mark2:89
```

重写（Overriding）

子类可以通过继承来的实例方法，类方法，实例属性，或下标脚本来实现自己的定制功能，我们把这种行为叫重写（overriding）。

我们可以使用 override 关键字来实现重写。

访问超类的方法、属性及下标脚本

你可以通过使用super前缀来访问超类的方法，属性或下标脚本。

重写	访问方法，属性，下标脚本
方法	super.somemethod()
属性	super.someProperty()
下标脚本	super[someIndex]

重写方法和属性

重写方法

在我们的子类中我们可以使用 override 关键字来重写超类的方法。

以下实例中我们重写了 show() 方法：

```
class SuperClass {
    func show() {
        print("这是超类 SuperClass")
    }
}

class SubClass: SuperClass  {
    override func show() {
        print("这是子类 SubClass")
    }
}

let superClass = SuperClass()
superClass.show()

let subClass = SubClass()
subClass.show()
```

以上程序执行输出结果为：

```
这是超类 SuperClass
这是子类 SubClass
```

重写属性

你可以提供定制的 getter（或 setter）来重写任意继承来的属性，无论继承来的属性是存储型的还是计算型的属性。

子类并不知道继承来的属性是存储型的还是计算型的，它只知道继承来的属性会有一个名字和类型。所以你在重写一个属性时，必需将它的名字和类型都写出来。

注意点：

- 如果你在重写属性中提供了 setter，那么你也一定要提供 getter。
- 如果你不想在重写版本中的 getter 里修改继承来的属性值，你可以直接通过super.someProperty来返回继承来的值，其中someProperty是你要重写的属性的名字。

以下实例我们定义了超类 Circle 及子类 Rectangle, 在 Rectangle 类中我们重写属性 area：

```
class Circle {
    var radius = 12.5
    var area: String {
        return "矩形半径 \(radius) "
    }
}

// 继承超类 Circle
class Rectangle: Circle {
    var print = 7
    override var area: String {
        return super.area + " ，但现在被重写为 \(print)"
    }
}

let rect = Rectangle()
rect.radius = 25.0
rect.print = 3
print("Radius \(rect.area)")
```

以上程序执行输出结果为：

```
Radius 矩形半径 25.0  ，但现在被重写为 3
```

重写属性观察器

你可以在属性重写中为一个继承来的属性添加属性观察器。这样一来，当继承来的属性值发生改变时，你就会监测到。

注意：你不可以为继承来的常量存储型属性或继承来的只读计算型属性添加属性观察器。

```
class Circle {
    var radius = 12.5
```

```
    var area: String {
        return "矩形半径为 \(radius) "
    }
}

class Rectangle: Circle {
    var print = 7
    override var area: String {
        return super.area + " ，但现在被重写为 \(print)"
    }
}


let rect = Rectangle()
rect.radius = 25.0
rect.print = 3
print("半径: \(rect.area)")

class Square: Rectangle {
    override var radius: Double {
        didSet {
            print = Int(radius/5.0)+1
        }
    }
}


let sq = Square()
sq.radius = 100.0
print("半径: \(sq.area)")
```

```
半径: 矩形半径为 25.0  ，但现在被重写为 3
半径: 矩形半径为 100.0  ，但现在被重写为 21
```

防止重写

我们可以使用 final 关键字防止它们被重写。

如果你重写了final方法，属性或下标脚本，在编译时会报错。

你可以通过在关键字class前添加final特性（final class）来将整个类标记为 final 的，这样的类是不可被继承的，否则会报编译错误。

```
final class Circle {
    final var radius = 12.5
    var area: String {
        return "矩形半径为 \(radius) "
    }
```

```
}
class Rectangle: Circle {
    var print = 7
    override var area: String {
        return super.area + " ，但现在被重写为 \(print)"
    }
}

let rect = Rectangle()
rect.radius = 25.0
rect.print = 3
print("半径: \(rect.area)")

class Square: Rectangle {
    override var radius: Double {
        didSet {
            print = Int(radius/5.0)+1
        }
    }
}

let sq = Square()
sq.radius = 100.0
print("半径: \(sq.area)")
```

由于以上实例使用了 final 关键字不允许重写，所以执行会报错：

```
error: var overrides a 'final' var
    override var area: String {
                 ^
note: overridden declaration is here
    var area: String {
        ^
error: var overrides a 'final' var
    override var radius: Double {
                 ^
note: overridden declaration is here
    final var radius = 12.5
              ^
error: inheritance from a final class 'Circle'
class Rectangle: Circle {
      ^
```

← Swift 下标脚本 Swift 构造过程 →

点我分享笔记