# Swift新特性介绍（四） 构造、析构、继承

# 构造函数

在创建一个类型的实例的时候，构造函数中的参数需要写全参数名：

```
struct Color {
    let red = 0.0, green = 0.0, blue = 0.0
    init(red: Double, green: Double, blue: Double) {
        self.red   = red
        self.green = green
        self.blue  = blue
    }
}
let magenta = Color(red: 1.0, green: 0.0, blue: 1.0)
let veryGreen = Color(0.0, 1.0, 0.0)
// this reports a compile-time error - external names are required
```

在构造函数中，还可以使用闭包来初始化属性：

```
struct Checkerboard {
    let boardColors: Bool[] = {
        var temporaryBoard = Bool[]()
        var isBlack = false
        for i in 1...10 {
            for j in 1...10 {
                temporaryBoard.append(isBlack)
                isBlack = !isBlack
            }
            isBlack = !isBlack
        }
        return temporaryBoard
        }()
    func squareIsBlackAtRow(row: Int, column: Int) -> Bool {
        return boardColors[(row * 10) + column]
    }
}
```

上面例子中的boardColors就是使用闭包来初始化的，间隔将棋盘格填色。

# 析构函数

Swift通过自动引用计数`ARC`来处理实例的内存管理，自动释放不再需要的实例以释放资源。当代码需要额外的操作时，可以自定义析构函数。需要注意的是，每个类最多只能有一个析构函数，析构函数不带任何参数和括号：

```
deinit {
// 执行析构过程
}
```

# 继承

任何一个不继承于其他类的类被称作基类。Swift的类不是从一个全局的基类继承而来，所有在类的定义中没有继承其他父类的类都是基类：

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

继承父类的子类在重写函数的时候，必须要加`override`关键词以标示该方法为重写方法，这样的好处是可以检查父类中是否有相对应的方法，不会因名称错误而造成一些不可预知的错误。

```
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"
    }
}
```

除了函数的重载，类的属性在子类中也可以重载，比如`getter`和`setter`方法：

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

使用`@final`标记类，可以禁止重写一个类，或者一个类的方法，属性，如：`@final class`

在子类的构造函数中需要注意的问题：

1、自定义构造函数要先调用自己类默认构造函数，自己重写默认构造函数要先调用父类默认构造函数

2、应该要先调用父类的构造函数或者自身的默认构造函数，以防止先给属性赋值了然后才调用父类或者自身的默认构造函数而把之前的赋值覆盖了