继承

@M了个J

https://github.com/CoderMJLee

http://cnblogs.com/mjios

码拉松

继承（Inheritance）

- 值类型（枚举、结构体）不支持继承，只有类支持继承

- 没有父类的类，称为：基类
- Swift并没有像OC、Java那样的规定：任何类最终都要继承自某个基类

> NOTE
>
> Swift classes do not inherit from a universal base class. Classes you define without specifying a superclass automatically become base classes for you to build upon.

- 子类可以重写父类的下标、方法、属性，重写必须加上`override`关键字

小码哥教育 SEEMYGO

内存结构

```
class Animal {
    var age = 0
}
class Dog : Animal {
    var weight = 0
}
class ErHa : Dog {
    var iq = 0
}
```

```
let a = Animal()
a.age = 10
// 32
print(Mems.size(ofRef: a))
/*
 0x00000001000073e0
 0x000000000000002
 0x000000000000000a
 0x0000000000000000
 */
print(Mems.memStr(ofRef: a))
```

```
let d = Dog()
d.age = 10
d.weight = 20
// 32
print(Mems.size(ofRef: d))
/*
 0x0000000100007490
 0x000000000000002
 0x000000000000000a
 0x0000000000000014
 */
print(Mems.memStr(ofRef: d))
```

```
let e = ErHa()
e.age = 10
e.weight = 20
e.iq = 30
// 48
print(Mems.size(ofRef: e))
/*
 0x0000000100007560
 0x000000000000002
 0x000000000000000a
 0x0000000000000014
 0x000000000000001e
 0x0000000000000000
 */
print(Mems.memStr(ofRef: e))
```

小码哥教育 SEEMYGO

重写实例方法、下标

```
class Animal {
    func speak() {
        print("Animal speak")
    }
    subscript(index: Int) -> Int {
        return index
    }
}
```

```
var anim: Animal
anim = Animal()
// Animal speak
anim.speak()
// 6
print(anim[6])
```

```
class Cat : Animal {
    override func speak() {
        super.speak()
        print("Cat speak")
    }
    override subscript(index: Int) -> Int {
        return super[index] + 1
    }
}
```

```
anim = Cat()
// Animal speak
// Cat speak
anim.speak()
// 7
print(anim[6])
```

重写类型方法、下标

- 被class修饰的类型方法、下标，允许被子类重写
- 被static修饰的类型方法、下标，不允许被子类重写

```
class Animal {
    class func speak() {
        print("Animal speak")
    }
    class subscript(index: Int) -> Int {
        return index
    }
}
// Animal speak
Animal.speak()
// 6
print(Animal[6])
```

```
class Cat : Animal {
    override class func speak() {
        super.speak()
        print("Cat speak")
    }
    override class subscript(index: Int) -> Int {
        return super[index] + 1
    }
}
// Animal speak
// Cat speak
Cat.speak()
// 7
print(Cat[6])
```

重写属性

- 子类可以将父类的属性（存储、计算）重写为计算属性
- 子类不可以将父类属性重写为存储属性
- 只能重写var属性，不能重写let属性
- 重写时，属性名、类型要一致
- 子类重写后的属性权限 不能小于 父类属性的权限
  - 如果父类属性是只读的，那么子类重写后的属性可以是只读的、也可以是可读写的
  - 如果父类属性是可读写的，那么子类重写后的属性也必须是可读写的

小码哥教育 SEEMYGO

重写实例属性

```
class Circle {
    var radius: Int = 0
    var diameter: Int {
        set {
            print("Circle setDiameter")
            radius = newValue / 2
        }
        get {
            print("Circle getDiameter")
            return radius * 2
        }
    }
}
```

```
var circle: Circle
circle = Circle()
circle.radius = 6
// Circle getDiameter
// 12
print(circle.diameter)
// Circle setDiameter
circle.diameter = 20
// 10
print(circle.radius)
```

小码哥教育 SEEMYGO

重写实例属性

```
class SubCircle : Circle {
    override var radius: Int {
        set {
            print("SubCircle setRadius")
            super.radius = newValue > 0 ? newValue : 0
        }
        get {
            print("SubCircle getRadius")
            return super.radius
        }
    }
    override var diameter: Int {
        set {
            print("SubCircle setDiameter")
            super.diameter = newValue > 0 ? newValue : 0
        }
        get {
            print("SubCircle getDiameter")
            return super.diameter
        }
    }
}
```

```
circle = SubCircle()

// SubCircle setRadius
circle.radius = 6

// SubCircle getDiameter
// Circle getDiameter
// SubCircle getRadius
// 12
print(circle.diameter)

// SubCircle setDiameter
// Circle setDiameter
// SubCircle setRadius
circle.diameter = 20

// SubCircle getRadius
// 10
print(circle.radius)
```

重写类型属性

- 被class修饰的计算类型属性，可以被子类重写
- 被static修饰的类型属性（存储、计算），不可以被子类重写

```
class Circle {
    static var radius: Int = 0
    class var diameter: Int {
        set {
            print("Circle setDiameter")
            radius = newValue / 2
        }
        get {
            print("Circle getDiameter")
            return radius * 2
        }
    }
}
```

```
class SubCircle : Circle {
    override static var diameter: Int {
        set {
            print("SubCircle setDiameter")
            super.diameter = newValue > 0 ? newValue : 0
        }
        get {
            print("SubCircle getDiameter")
            return super.diameter
        }
    }
}
```

```
Circle.radius = 6
// Circle getDiameter
// 12
print(Circle.diameter)
// Circle setDiameter
Circle.diameter = 20
// 10
print(Circle.radius)
```

```
SubCircle.radius = 6
// SubCircle getDiameter
// Circle getDiameter
// 12
print(SubCircle.diameter)
// SubCircle setDiameter
// Circle setDiameter
SubCircle.diameter = 20
// 10
print(SubCircle.radius)
```

小码哥教育 SEEMYGO

属性观察器

- 可以在子类中为父类属性（除了只读计算属性、let属性）增加属性观察器

```
class Circle {
    var radius: Int = 1
}
class SubCircle : Circle {
    override var radius: Int {
        willSet {
            print("SubCircle willSetRadius", newValue)
        }
        didSet {
            print("SubCircle didSetRadius", oldValue, radius)
        }
    }
}
var circle = SubCircle()
// SubCircle willSetRadius 10
// SubCircle didSetRadius 1 10
circle.radius = 10
```

小码哥教育 SEEMYGO

属性观察器

```
class Circle {
    var radius: Int = 1 {
        willSet {
            print("Circle willSetRadius", newValue)
        }
        didSet {
            print("Circle didSetRadius", oldValue, radius)
        }
    }
}
class SubCircle : Circle {
    override var radius: Int {
        willSet {
            print("SubCircle willSetRadius", newValue)
        }
        didSet {
            print("SubCircle didSetRadius", oldValue, radius)
        }
    }
}
```

```
var circle = SubCircle()
// SubCircle willSetRadius 10
// Circle willSetRadius 10
// Circle didSetRadius 1 10
// SubCircle didSetRadius 1 10
circle.radius = 10
```

小码哥教育 SEEMYGO

属性观察器

```
class Circle {
    var radius: Int {
        set {
            print("Circle setRadius", newValue)
        }
        get {
            print("Circle getRadius")
            return 20
        }
    }
}
class SubCircle : Circle {
    override var radius: Int {
        willSet {
            print("SubCircle willSetRadius", newValue)
        }
        didSet {
            print("SubCircle didSetRadius", oldValue, radius)
        }
    }
}
```

```
var circle = SubCircle()
// Circle getRadius
// SubCircle willSetRadius 10
// Circle setRadius 10
// Circle getRadius
// SubCircle didSetRadius 20 20
circle.radius = 10
```

小码哥教育 SEEMYGO

属性观察器

```
class Circle {
    class var radius: Int {
        set {
            print("Circle setRadius", newValue)
        }
        get {
            print("Circle getRadius")
            return 20
        }
    }
}
class SubCircle : Circle {
    override static var radius: Int {
        willSet {
            print("SubCircle willSetRadius", newValue)
        }
        didSet {
            print("SubCircle didSetRadius", oldValue, radius)
        }
    }
}
```

```
// Circle getRadius
// SubCircle willSetRadius 10
// Circle setRadius 10
// Circle getRadius
// SubCircle didSetRadius 20 20
SubCircle.radius = 10
```

小码哥教育 SEEMYGO

final

- 被final修饰的方法、下标、属性，禁止被重写
- 被final修饰的类，禁止被继承