Scala 类和对象 Scala 模式匹配

Scala Trait(特征)

Scala Trait(特征) 相当于 Java 的接口，实际上它比接口还功能强大。

与接口不同的是，它还可以定义属性和方法的实现。

一般情况下Scala的类只能够继承单一父类，但是如果是 Trait(特征) 的话就可以继承多个，从结果来看就是实现了多重继承。

Trait(特征) 定义的方式与类类似，但它使用的关键字是 **trait**，如下所示：

```
trait Equal {
  def isEqual(x: Any): Boolean
  def isNotEqual(x: Any): Boolean = !isEqual(x)
}
```

以上Trait(特征)由两个方法组成：**isEqual** 和 **isNotEqual**。isEqual 方法没有定义方法的实现，isNotEqual定义了方法的实现。

子类继承特征可以实现未被实现的方法。所以其实 Scala Trait(特征)更像 Java 的抽象类。

以下演示了特征的完整实例：

```
/* 文件名: Test.scala
 * author:菜鸟教程
 * url:www.runoob.com
 */
trait Equal {
  def isEqual(x: Any): Boolean
  def isNotEqual(x: Any): Boolean = !isEqual(x)
}

class Point(xc: Int, yc: Int) extends Equal {
  var x: Int = xc
  var y: Int = yc
  def isEqual(obj: Any) =
    obj.isInstanceOf[Point] &&
    obj.asInstanceOf[Point].x == x
}

object Test {
   def main(args: Array[String]) {
      val p1 = new Point(2, 3)
      val p2 = new Point(2, 4)
      val p3 = new Point(3, 3)

      println(p1.isNotEqual(p2))
      println(p1.isNotEqual(p3))
      println(p1.isNotEqual(2))
```

```
    }
}
```

执行以上代码，输出结果为：

```
$ scalac Test.scala
$ scala Test
false
true
true
```

特征构造顺序

特征也可以有构造器，由字段的初始化和其他特征体中的语句构成。这些语句在任何混入该特征的对象在构造时都会被执行。

构造器的执行顺序：

- 调用超类的构造器；
- 特征构造器在超类构造器之后、类构造器之前执行；
- 特征由左到右被构造；
- 每个特征当中，父特征先被构造；
- 如果多个特征共有一个父特征，父特征不会被重复构造
- 所有特征被构造完毕，子类被构造。

构造器的顺序是类的线性化的反向。线性化是描述某个类型的所有超类型的一种技术规格。

← Scala 类和对象 Scala 模式匹配 →