# **快速入门 Kotlin 编程**

## 1.变量与函数

val：用于声明不可变的变量，这种变量在初始赋值之后就再也不能重新赋值，对应 Java 中的 final 变量。

var：用于声明一个可变的变量，这种变量在初始赋值之后仍然可以再被重新赋值，对应 Java 中的非 final 变量。

Kotlin 在赋值时会进行自动推导，可以根据值的类型推导出变量的类型，如果使用下面这种****延迟赋值****的方式，那么 Kotlin 将无法推到值得类型，这样程序就变报错

永远优先使用 val 来声明变量，当 val 无法满足你的需求时再使用 var，这样设计出来的程序更加健壮，也更加符合高质量的编码规范。

#### 使用主构造函数

class Student(val sno: String, val grade: Int) : Person() {}

val student = Student("a123", 5)复制代码

构造函数的参数直接写在类后面即可，如果想在主构造函数中编写一些逻辑的话，可以使用 init 声明结构体，

class Student(val sno: String, val grade: Int) : Person() {

// 将主构造函数的逻辑写在 init 结构体中

init {

println("sno is " + sno)

println("grade is " + grade)

}

}

val student = Student("a123", 5)复制代码

这样书写后可以在初始化 Student 类时打印 sno 和 grade 的值，这里的一个规范与 Java 中相同，就是在初始化子类时必须调用父类的构造函数。但是这么写会调用父类的哪个构造方法呢，这取决于 Person() 中的括号中有几个参数，这里没有传入参数，所以会调用父类的无参构造函数。

#### 使用次构造函数

Kotlin 提供了一个给函数设定参数默认值的功能，基本上可以替代次构造函数的作用，但是考虑到知识结构的完整性，还是说一下此构造函数的相关知识并探讨一下括号的问题在次构造函数上的区别。

**一个类只能有一个主构造函数，但是可以有多个次构造函数**，次构造函数也可以用于实例化一个类，这一点和主构造函数没有什么不同，只不过它**有函数体**。

Kotlin 规定，当一个类既有主构造函数又有次构造函数时，所有的次构造函数都必须调用主构造函数（包括间接调用），这里通过一个例子进行简单的阐明。

class Student(val sno: String, val grade: Int, name: String, age: Int)

: Person(name, age) {

constructor(name: String, age: Int) : this("", 0, name, age) {

}

constructor() : this("", 0){

}

}复制代码

这里定义了两个次构造函数，第一个次构造函数接收 name 和 age 参数，然后又通过 this 调用主构造函数，并将 sno 和 grade 参数赋值，第二个次构造函数不接收任何参数，通过 this 调用了上面的次构造函数，并将 name 和 age 参数也成功进行了赋值，由于第二个次构造函数间接的调用了主构造函数，所以这也是合法的。

这么写完之后就拥有了三种初始化 Student 类的方式

val student1 = Student()

val student2 = Student("Jack", 19)

val student3 = Student("a123", 5, "Jack", 19)

在一个类中显式的设置了次构造函数并且没有显式的设置主构造函数，此时是没有主构造函数的，这种操作在 Kotlin 中是允许的。

class Student : Person {

constructor(name: String, age: Int) : super(name, age) {

}

}复制代码

这里的 Student 类的后面没有显式的定义主构造函数，同时又因为定义了次构造函数，所以现在 Student 类是没有主构造函数的，那么在继承 Person 类是就不需要再添加括号了，另外由于没有主构造函数，次构造函数只能显式的调用父类的构造函数，所以可以将 this 换成 super。

。