学到这里，我们就将 Kotlin 的主构造函数基本掌握了，是不是觉得继承时的这对括号问题也不是那么难以理解？但是，Kotlin 在括号这个问题上的复杂度并不仅限于此，因为我们还没涉及 Kotlin 构造函数中的另一个组成部分——次构造函数。

其实你几乎是用不到次构造函数的，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，我们会在本章最后学习这部分内容。但是考虑到知识结构的完整性，我决定还是介绍一下次构造函数的相关知识，顺便探讨一下括号问题在次构造函数上的区别。

你要知道，任何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，只不过它是有函数体的。

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）。这里我通过一个具体的例子就能简单阐明，代码如下：

```
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) {
    }
}
```

次构造函数是通过 constructor 关键字来定义的，这里我们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，然后它又通过 this 关键字调用了主构造函数，并将 sno 和 grade 这两个参数赋值成初始值；第二个次构造函数不接收任何参数，它通过 this 关键字调用了我们刚才定义的第一个次构造函数，并将 name 和 age 参数也赋值成初始值，由于第二个次构造函数间接调用了主构造函数，因此这仍然是合法的。

那么现在我们就拥有了 3 种方式来对 Student 类进行实体化，分别是通过不带参数的构造函数、通过带两个参数的构造函数和通过带 4 个参数的构造函数，对应代码如下所示：

```
val student1 = Student()
val student2 = Student("Jack", 19)
val student3 = Student("a123", 5, "Jack", 19)
```

这样我们就将次构造函数的用法掌握得差不多了，但是到目前为止，继承时的括号问题还没有进一步延伸，暂时和之前学过的场景是一样的。

那么接下来我们就再来看一种非常特殊的情况：类中只有次构造函数，没有主构造函数。这种情况真的十分少见，但在 Kotlin 中是允许的。当一个类没有显式地定义主构造函数且定义了次构造函数时，它就是没有主构造函数的。我们结合代码来看一下：

```
class Student : Person {
    constructor(name: String, age: Int) : super(name, age) {
    }
}
```