```
class Student {
    var sno = ""
    var grade = 0
}
```

现在 Student 和 Person 这两个类之间是没有任何继承关系的，想要让 Student 类继承 Person 类，我们得做两件事才行。

第一件事，使 Person 类可以被继承。可能很多人会觉得奇怪，尤其是有 Java 编程经验的人。一个类本身不就是可以被继承的吗？为什么还要使 Person 类可以被继承呢？这就是 Kotlin 不同的地方，在 Kotlin 中任何一个非抽象类默认都是不可以被继承的，相当于 Java 中给类声明了 final 关键字。之所以这么设计，其实和 val 关键字的原因是差不多的，因为类和变量一样，最好都是不可变的，而一个类允许被继承的话，它无法预知子类会如何实现，因此可能就会存在一些未知的风险。*Effective Java* 这本书中明确提到，如果一个类不是专门为继承而设计的，那么就应该主动将它加上 final 声明，禁止它可以被继承。

很明显，Kotlin 在设计的时候遵循了这条编程规范，默认所有非抽象类都是不可以被继承的。之所以这里一直在说非抽象类，是因为抽象类本身是无法创建实例的，一定要由子类去继承它才能创建实例，因此抽象类必须可以被继承才行，要不然就没有意义了。由于 Kotlin 中的抽象类和 Java 中并无区别，这里我就不再多讲了。

既然现在 Person 类是无法被继承的，我们得让它可以被继承才行，方法也很简单，在 Person 类的前面加上 open 关键字就可以了，如下所示：

```
open class Person {
    ...
}
```

加上 open 关键字之后，我们就是在主动告诉 Kotlin 编译器，Person 这个类是专门为继承而设计的，这样 Person 类就允许被继承了。

第二件事，要让 Student 类继承 Person 类。在 Java 中继承的关键字是 extends，而在 Kotlin 中变成了一个冒号，写法如下：

```
class Student : Person() {
    var sno = ""
    var grade = 0
}
```

继承的写法如果只是替换一下关键字倒也挺简单的，但是为什么 Person 类的后面要加上一对括号呢？Java 中继承的时候好像并不需要括号。对于初学 Kotlin 的人来讲，这对括号确实挺难理解的，也可能是 Kotlin 在这方面设计得太复杂了，因为它还涉及主构造函数、次构造函数等方面的知识，这里我尽量尝试用最简单易懂的讲述来让你理解这对括号的意义和作用，同时顺便学习一下 Kotlin 中的主构造函数和次构造函数。