在这里，Person 类后面的一对空括号表示 Student 类的主构造函数在初始化的时候会调用 Person 类的无参数构造函数，即使在无参数的情况下，这对括号也不能省略。

2.5 面向对象编程 53

在这里，Person 类后面的一对空括号表示 Student 类的主构造函数在初始化的时候会调用 Person 类的无参数构造函数，即使在无参数的情况下，这对括号也不能省略。

而如果我们将 Person 改造一下，将姓名和年龄都放到主构造函数当中，如下所示：

```
open class Person(val name: String, val age: Int) {
    ...
}
```

此时你的 Student 类一定会报错，当然，如果你的 main()函数还保留着之前创建 Person 实例的代码，那么这里也会报错，但是它和我们接下来要讲的内容无关，你可以自己修正一下，或者干脆直接删掉这部分代码。

现在回到 Student 类当中，它一定会提示如图 2.19 所示的错误。

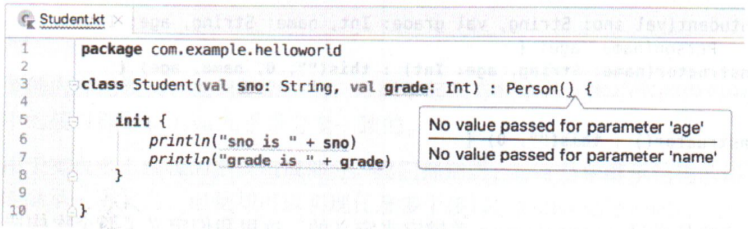

图 2.19 Student 类提示错误

这里出现错误的原因也很明显，Person 类后面的空括号表示要去调用 Person 类中无参的构造函数，但是 Person 类现在已经没有无参的构造函数了，所以就提示了上述错误。

如果我们想解决这个错误的话，就必须给 Person 类的构造函数传入 name 和 age 字段，可是 Student 类中也没有这两个字段呀。很简单，没有就加呗。我们可以在 Student 类的主构造函数中加上 name 和 age 这两个参数，再将这两个参数传给 Person 类的构造函数，代码如下所示：

```
class Student(val sno: String, val grade: Int, name: String, age: Int) :
        Person(name, age) {
    ...
}
```

注意，我们在 Student 类的主构造函数中增加 name 和 age 这两个字段时，不能再将它们声明成 val，因为在主构造函数中声明成 val 或者 var 的参数将自动成为该类的字段，这就会导致和父类中同名的 name 和 age 字段造成冲突。因此，这里的 name 和 age 参数前面我们不用加任何关键字，让它的作用域仅限定在主构造函数当中即可。

现在就可以通过如下代码来创建一个 Student 类的实例：

```
val student = Student("a123", 5, "Jack", 19)
```