熟悉 Java 的人一定知道，Java 中继承使用的关键字是 extends，实现接口使用的关键字是 implements，而 Kotlin 中统一使用冒号，中间用逗号进行分隔。上述代码就表示 Student 类继承了 Person 类，同时还实现了 Study 接口。另外接口的后面不用加上括号，因为它没有构造函数可以去调用。

Study 接口中定义了 readBooks()和 doHomework()这两个待实现函数，因此 Student 类必须实现这两个函数。Kotlin 中使用 override 关键字来重写父类或者实现接口中的函数，这里我们只是简单地在实现的函数中打印了一行日志。

现在我们可以在 main()函数中编写如下代码来调用这两个接口中的函数：

```
fun main() {
    val student = Student("Jack", 19)
    doStudy(student)
}

fun doStudy(study: Study) {
    study.readBooks()
    study.doHomework()
}
```

这里为了向你演示一下多态编程的特性，我故意将代码写得复杂了一点。首先创建了一个 Student 类的实例，本来是可以直接调用该实例的 readBooks()和 doHomework()函数的，但是我没有这么做，而是将它传入到了 doStudy()函数中。doStudy()函数接收一个 Study 类型的参数，由于 Student 类实现了 Study 接口，因此 Student 类的实例是可以传递给 doStudy()函数的，接下来我们调用了 Study 接口的 readBooks()和 doHomework()函数，这种就叫作面向接口编程，也可以称为多态。

现在运行一下代码，结果如图 2.20 所示。

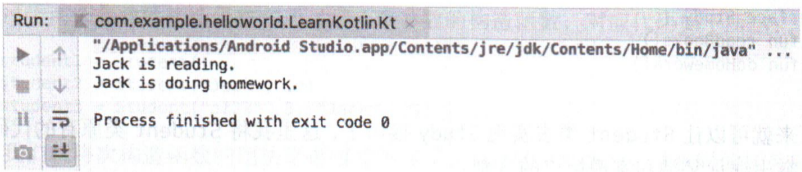

图 2.20　调用接口中的函数

这样我们就将 Kotlin 中接口的用法基本学完了，是不是很简单？不过为了让接口的功能更加灵活，Kotlin 还增加了一个额外的功能：允许对接口中定义的函数进行默认实现。其实 Java 在 JDK 1.8 之后也开始支持这个功能了，因此总体来说，Kotlin 和 Java 在接口方面的功能仍然是一模一样的。

下面我们学习一下如何对接口中的函数进行默认实现，修改 Study 接口中的代码，如下所示：