```
interface Study {
    fun readBooks()

    fun doHomework() {
        println("do homework default implementation.")
    }
}
```

可以看到，我们给 doHomework()函数加上了函数体，并且在里面打印了一行日志。如果接口中的一个函数拥有了函数体，这个函数体中的内容就是它的默认实现。现在当一个类去实现 Study 接口时，只会强制要求实现 readBooks()函数，而 doHomework()函数则可以自由选择实现或者不实现，不实现时就会自动使用默认的实现逻辑。

现在回到 Student 类当中，你会发现如果我们删除了 doHomework()函数，代码是不会提示错误的，而删除 readBooks()函数则不行。当删除了 doHomework()函数之后，重新运行 main()函数，结果如图 2.21 所示。可以看到，程序正如我们所预期的那样运行了。

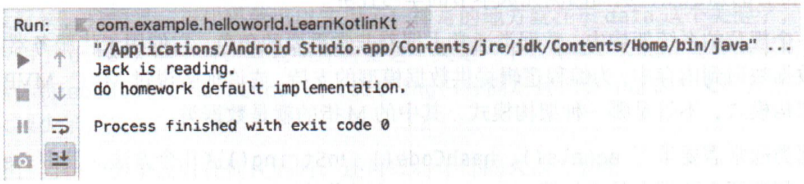

图 2.21 调用接口的默认实现函数

现在你已经掌握了 Kotlin 面向对象编程中最主要的一些内容，接下来我们再学习一个和 Java 相比变化比较大的部分——函数的可见性修饰符。

熟悉 Java 的人一定知道，Java 中有 public、private、protected 和 default（什么都不写）这 4 种函数可见性修饰符。Kotlin 中也有 4 种，分别是 public、private、protected 和 internal，需要使用哪种修饰符时，直接定义在 fun 关键字的前面即可。下面我详细介绍一下 Java 和 Kotlin 中这些函数可见性修饰符的异同。

首先 private 修饰符在两种语言中的作用是一模一样的，都表示只对当前类内部可见。public 修饰符的作用虽然也是一致的，表示对所有类都可见，但是在 Kotlin 中 public 修饰符是默认项，而在 Java 中 default 才是默认项。前面我们定义了那么多的函数，都没有加任何的修饰符，所以它们默认都是 public 的。protected 关键字在 Java 中表示对当前类、子类和同一包路径下的类可见，在 Kotlin 中则表示只对当前类和子类可见。Kotlin 抛弃了 Java 中的 default 可见性（同一包路径下的类可见），引入了一种新的可见性概念，只对同一模块中的类可见，使用的是 internal 修饰符。比如我们开发了一个模块给别人使用，但是有一些函数只允许在模块内部调用，不想暴露给外部，就可以将这些函数声明成 internal。关于模块开发的内容，我们会在本书的最后一章学习。