```
        public void onClick(View v) {
        }
    });
```

而用 Kotlin 代码实现同样的功能，就可以使用函数式 API 的写法来对代码进行简化，结果如下：

```
button.setOnClickListener {
}
```

可以看到，使用这种写法，代码明显精简了很多。这段给按钮注册点击事件的代码，我们在正式开始学习 Android 程序开发之后将会经常用到。

最后提醒你一句，本小节中学习的 Java 函数式 API 的使用都限定于从 Kotlin 中调用 Java 方法，并且单抽象方法接口也必须是用 Java 语言定义的。你可能会好奇为什么要这样设计。这是因为 Kotlin 中有专门的高阶函数来实现更加强大的自定义函数式 API 功能，从而不需要像 Java 这样借助单抽象方法接口来实现。关于高阶函数的用法，我们会在本书的第 6 章进行学习。

2.7 空指针检查

我之前看过某国外机构做的一个统计，Android 系统上崩溃率最高的异常类型就是空指针异常（NullPointerException）。相信不只是 Android，其他系统上也面临着相同的问题。若要分析其根本原因的话，我觉得主要是因为空指针是一种不受编程语言检查的运行时异常，只能由程序员主动通过逻辑判断来避免，但即使是最出色的程序员，也不可能将所有潜在的空指针异常全部考虑到。

我们来看一段非常简单的 Java 代码：

```
public void doStudy(Study study) {
    study.readBooks();
    study.doHomework();
}
```

这是我们在 2.5.3 小节编写过的一个 doStudy()方法，我将它翻译成了 Java 版。这段代码没有任何复杂的逻辑，只是接收了一个 Study 参数，并且调用了参数的 readBooks()和 doHomework()方法。

这段代码安全吗？不一定，因为这要取决于调用方传入的参数是什么，如果我们向 doStudy()方法传入了一个 null 参数，那么毫无疑问这里就会发生空指针异常。因此，更加稳妥的做法是在调用参数的方法之前先进行一个判空处理，如下所示：

```
public void doStudy(Study study) {
    if (study != null) {
        study.readBooks();
        study.doHomework();
    }
}
```