```
fun main() {
    var a: Int = 10
    a = a * 10
    println("a = " + a)
}
```

现在编译器就不会再报错了，重新运行一下代码，结果如图 2.7 所示。

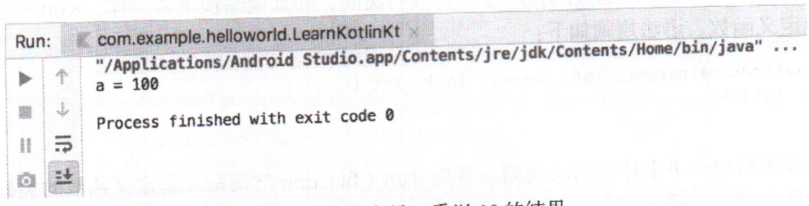


图 2.7 打印变量 a 乘以 10 的结果

可以看到，a 的值变成了 100，这说明我们的数学运算操作成功了。

这里你可能会产生疑惑：既然 val 关键字有这么多的束缚，为什么还要用这个关键字呢？干脆全部用 var 关键字不就好了。其实 Kotlin 之所以这样设计，是为了解决 Java 中 final 关键字没有被合理使用的问题。

在 Java 中，除非你主动在变量前声明了 final 关键字，否则这个变量就是可变的。然而这并不是一件好事，当项目变得越来越复杂，参与开发的人越来越多时，你永远不知道一个可变的变量会在什么时候被谁给修改了，即使它原本不应该被修改，这就经常会导致出现一些很难排查的问题。因此，一个好的编程习惯是，除非一个变量明确允许被修改，否则都应该给它加上 final 关键字。

但是，不是每个人都能养成这种良好的编程习惯。我相信至少有 90%的 Java 程序员没有主动在变量前加上 final 关键字的意识，仅仅因为 Java 对此是不强制的。因此，Kotlin 在设计的时候就采用了和 Java 完全不同的方式，提供了 val 和 var 这两个关键字，必须由开发者主动声明该变量是可变的还是不可变的。

那么我们应该什么时候使用 val，什么时候使用 var 呢？这里我告诉你一个小诀窍，就是永远优先使用 val 来声明一个变量，而当 val 没有办法满足你的需求时再使用 var。这样设计出来的程序会更加健壮，也更加符合高质量的编码规范。

### 2.3.2 函数

不少刚接触编程的人对于函数和方法这两个概念有些混淆，不明白它们有什么区别。其实，函数和方法就是同一个概念，这两种叫法都是从英文翻译过来的，函数翻译自 function，方法翻译自 method，它们并没有什么区别，只是不同语言的叫法习惯不一样而已。而因为 Java 中方法的叫法更普遍一些，Kotlin 中函数的叫法更普遍一些，因此本书里可能会交叉使用两种叫法，你