```
object Singleton {
    fun singletonTest() {
        println("singletonTest is called.")
    }
}
```

可以看到，在 Kotlin 中我们不需要私有化构造函数，也不需要提供 getInstance()这样的静态方法，只需要把 class 关键字改成 object 关键字，一个单例类就创建完成了。而调用单例类中的函数也很简单，比较类似于 Java 中静态方法的调用方式：

```
Singleton.singletonTest()
```

这种写法虽然看上去像是静态方法的调用，但其实 Kotlin 在背后自动帮我们创建了一个 Singleton 类的实例，并且保证全局只会存在一个 Singleton 实例。

这样我们就将 Kotlin 面向对象编程最主要的知识掌握了，这也是非常充实的一节内容，希望你能好好掌握和消化。要知道，你往后的编程工作基本上是建立在面向对象编程的基础之上的。

2.6 Lambda 编程

可能很多 Java 程序员对于 Lambda 编程还比较陌生，但其实这并不是什么新鲜的技术。许多现代高级编程语言在很早之前就开始支持 Lambda 编程了，但是 Java 却直到 JDK 1.8 之后才加入了 Lambda 编程的语法支持。因此，大量早期开发的 Java 和 Android 程序其实并未使用 Lambda 编程的特性。

而 Kotlin 从第一个版本开始就支持了 Lambda 编程，并且 Kotlin 中的 Lambda 功能极为强大，我甚至认为 Lambda 才是 Kotlin 的灵魂所在。不过，本章只是 Kotlin 的入门章节，我不可能在这短短一节里就将 Lambda 的方方面面全部覆盖。因此，这一节我们只学习一些 Lambda 编程的基础知识，而像高阶函数、DSL 等高级 Lambda 技巧，我们会在本书的后续章节慢慢学习。

2.6.1 集合的创建与遍历

集合的函数式 API 是用来入门 Lambda 编程的绝佳示例，不过在此之前，我们得先学习创建集合的方式才行。

传统意义上的集合主要就是 List 和 Set，再广泛一点的话，像 Map 这样的键值对数据结构也可以包含进来。List、Set 和 Map 在 Java 中都是接口，List 的主要实现类是 ArrayList 和 LinkedList，Set 的主要实现类是 HashSet，Map 的主要实现类是 HashMap，熟悉 Java 的人对这些集合的实现类一定不会陌生。

现在我们提出一个需求，创建一个包含许多水果名称的集合。如果是在 Java 中你会怎么实现？可能你首先会创建一个 ArrayList 的实例，然后将水果的名称一个个添加到集合中。当然，在 Kotlin 中也可以这么做：