Java库谜题56：大问题 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/138/2021_2022_Java_E5_BA_93_E8_B0_9C_c97_138648.htm 作为一项热身活动，我们来测试一下你对BigInteger的了解程度。下面这个程序将打印出什么呢？ import java.math.BigInteger. public class BigProblem { public static void main(String[ ] args) { BigInteger fiveThousand = new BigInteger("5000"). BigInteger fiftyThousand = new BigInteger("50000"). BigInteger fiveHundredThousand = new BigInteger("500000"). BigInteger total = BigInteger.ZERO. total.add(fiveThousand). total.add(fiftyThousand). total.add(fiveHundredThousand). System.out.println(total). } } 你可能会认为这个程序会打印出555000。毕竟，它将total设置为用BigInteger表示的0，然后将5,000、50,000和500,000加到了这个变量上。如果你运行该程序，你就会发现它打印的不是555000，而是0。很明显，所有这些加法对total没有产生任何影响。 对此有一个很好理由可以解释：BigInteger实例是不可变的。String、BigDecimal以及包装器类型：Integer、Long、Short、Byte、Character、Boolean、Float和Double也是如此，你不能修改它们的值。我们不能修改现有实例的值，对这些类型的操作将返回新的实例。起先，不可变类型看起来可能很不自然，但是它们具有很多胜过与其向对应的可变类型的优势。不可变类型更容易设计、实现和使用；它们出错的可能性更小，并且更加安全[EJ Item 13]。 为了在一个包含对不可变对象引用的变量上执行计算，我们需要将计算的结果赋值给该变量。这样做就会产生下面的程序，它将打印出我

们所期望的555000： import java.math.BigInteger. public class BigProblem { public static void main(String[] args) { BigInteger fiveThousand = new BigInteger("5000"). BigInteger fiftyThousand = new BigInteger("50000"). BigInteger fiveHundredThousand = new BigInteger("500000"). BigInteger total = BigInteger.ZERO. total = total.add(fiveThousand). total = total.add(fiftyThousand). total = total.add(fiveHundredThousand). System.out.println(total). } } 本谜题的教训是：不要被误导，认为不可变类型是可变的。这是一个在刚入门的Java程序员中很常见的错误。公正地说，Java不可变类型的某些方法名促使我们走上了歧途。像add、subtract和negate之类的名字似乎是在暗示这些方法将修改它们所调用的实例。也许plus、minus和negation才是更好的名字。 对API设计来说，其教训是：在命名不可变类型的方法时，应该优选介词和名词，而不是动词。介词适用于带有参数的方法，而名词适用于不带参数的方法。对语言设计者而言，其教训与谜题2相同，那就是应该考虑对操作符重载提供有限的支持，这样算数操作符就可以作用于诸如BigInteger这样的数值型的引用类型。由此，即使是初学者也不会认为计算表达式total fiveThousand将会对total的值产生任何影响。