JAVA字符谜题8:字符串奶酪 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_AD_97_E7_AC_A6_c97_138558.htm 下面的程序从一个字节序列创建了一个字符串，然后迭代遍历字符串中的字符，并将它们作为数字打印。请描述一下程序打印出来的数字序列： public class StringCheese { public static void main(String[] args) { byte bytes[] = new byte[256]. for (int i = 0. i bytes[i] = (byte)i. String str = new String(bytes). for (int i = 0, n = str.length(). i System.out.println((int)str.charAt(i) " "). } } 首先，byte数组用从0到255每一个可能的byte数值进行了初始化，然后这些byte数值通过String构造器被转换成了char数值。最后，char数值被转型为int数值并被打印。打印出来的数值肯定是非负整数，因为char数值是无符号的，因此，你可能期望该程序将按顺序打印出0到255的整数。 如果你运行该程序，可能会看到这样的序列。但是在运行一次，可能看到的就不是这个序列了。我们在四台机器上运行它，会看到四个不同的序列，包括前面描述的那个序列。这个程序甚至都不能保证会正常终止，比打印其他任何特定字符串都要缺乏这种保证。它的行为完全是不确定的。 这里的罪魁祸首就是String(byte[])构造器。有关它的规范描述道：“ 在通过解码使用平台缺省字符集的指定byte数组来构造一个新的String时，该新String的长度是字符集的一个函数，因此，它可能不等于byte数组的长度。当给定的所有字节在缺省字符集中并非全部有效时，这个构造器的行为是不确定的 ” [Java-API]。 到底什么是字符集？从技术角度上讲，它是“ 被编码的字符集合和字符编码模式的结

合物 ”[Java-API]。换句话说，字符集是一个包，包含了字符、表示字符的数字编码以及在字符编码序列和字节序列之间来回转换的方式。转换模式在字符集之间存在着很大的区别：某些是在字符和字节之间做一对一的映射，但是大多数都不是这样。ISO-8859-1是唯一能够让该程序按顺序打印从0到255的整数的缺省字符集，它更为大家所熟知的名字是Latin-1[ISO-8859-1]。 J2SE运行期环境（JRE）的缺省字符集依赖于底层的操作系统和语言。如果你想知道你的JRE的缺省字符集，并且你使用的是5.0或更新的版本，那么你可以通过调用java.nio.charset.Charset.defaultCharset()来了解。如果你使用的是较早的版本，那么你可以通过阅读系统属性 “ file.encoding ”来了解。 幸运的是，你没有被强制要求必须去容忍各种稀奇古怪的缺省字符集。当你在char序列和byte序列之间做转换时，你可以且通常是应该显式地指定字符集。除了接受byte数字之外，还可以接受一个字符集名称的String构造器就是专为此目的而设计的。如果你用下面的构造器去替换在最初的程序中的String构造器，那么不管缺省的字符集是什么，该程序都保证能够按照顺序打印从0到255的整数：String str = new String(bytes, "ISO-8859-1"). 这个构造器声明会抛出UnsupportedEncodingException异常，因此你必须捕获它，或者更适宜的方式是声明main方法将抛出它，要不然程序不能通过编译。尽管如此，该程序实际上不会抛出异常。Charset的规范要求Java平台的每一种实现都要支持某些种类的字符集，ISO-8859-1就位列其中。 这个谜题的教训是：每当你要将一个byte序列转换成一个String时，你都在使用某一个字符集，不管你是否显式地指定了它。如果你想让你的程

序的行为是可预知的，那么就请你在每次使用字符集时都明确地指定。对API的设计者来说，提供这么一个依赖于缺省字符集的String(byte[])构造器可能并非是一个好主意。 