基础入门：关于java数组的深度思考 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022__E5_9F_BA_E7_A1_80_E5_85_A5_E9_c104_145109.htm 刚刚开始接触java数组的人都会听到一句类似的话：java是纯面向对象的语言，他的数组也是一个对象。 于是乎，我就按照一个对象的方式来使用数组，心安理得。直到我接触到C的数组后，才发现将数组作为一个类来使用在实现上是多么的“ 不自然 ”。 首先我们看一下表面现象，数组创建的时候采用的是如下语句：MyClass[] arr = new MyClass[9]；而普通类采用的是如下语句：MyClass obj = new MyClass（）；就是说，创建数组的时候不使用小括号传参。使得数组和普通类看起来就有很多不同，因为小括号里的参数是传递给构造方法的，进而让人感觉数组类是没有构造方法的。 再往深了想，还有很多让人感觉不自然的东西。可以肯定的是，java确实将数组作为了一个类来处理。还是用上面的例子说明：可以通过以下方法得到MyClass[]的Class实例：arr.getClass（）或MyClass[].class.这样，我就可以向数组类里面“ 窥探 ”了。 Class clazz = MyClass[].class；System.out.println（clazz.getConstructors（）。length）；打印出来的结果是0；证明数组类确实没有构造方法。 如果强行执行clazz.newInstance（）；就会得到下面的错误。 java.lang.InstantiationException：[Larraytest.MyClass；证明数组类不能够通过普通的反射方式来创建一个实例。 再看看数组类的“ 庐山真面目 ”：System.out.println（clazz）；输出是：[Larraytest.MyClass 对Java Class文件结构稍有了结就知道，这个字符串的意思就是一个元素类型

为arraytest.MyClass的一维数组。也就是说，数组类型不是和普通类一样，以一个全限定路径名 类名来作为自己的唯一标示的，而是以[一个或者多个L 数组元素类全限定路径 类来最为唯一标示的。这个（）也是数组和普通类的区别。而这个区别似乎在某种程度上说明数组和普通java类在实现上有很大区别。因为java虚拟机（java指令集）在处理数组类和普通类的时候，肯定会做出区分。我猜想，可能会有专门的java虚拟机指令来处理数组。 既然我们可以得到数组的Class类实例，就说明肯定需要调用ClassLoader的defineClass（不一定非要是loadClass方法）方法，来构造一个Class实例。java虚拟机规范规定，任何一个可以被加载的类，如果其类文件存储在文件系统上，那么一个*.class文件只能存储一个类信息，也就是说，数组类的信息不可能以类文件的形式存储在本地磁盘上（否则任意一个类都要配有255个数组类了……），既然这样，那就说明java虚拟机肯定内置了一块用来声明数组类的数据（不管是几级数组）。这是符合java虚拟机规范的，规范规定class类数据可以来自任意介质，包括本地磁盘、网络、数据库、内存等等。 分析到这里，我基本上可以肯定：java对数组对象化的操作的支持是指令级的，也就是说java虚拟机有专门针对数组的指令。数组的Class类实例是java虚拟机动态创建动态加载的，其结构与普通java类的Class实例有一些不同。JDK API中有一个java.lang.reflect.Array类，这个类提供了很多方法（绝大多数是native方法，这在另一个方面证明了java对数组的支持是专用指令支持的，否则用本地方法干嘛^_^），用来弥补我们对数组操作的局限性。