从C 转到Java如何把握关键 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/237/2021_2022__E4_BB_8EC___E8_BD_AC_E5_c104_237665.htm 本文将提供一个对这些概念的简明的解释，而不是提供一些深入的或者如何使用的问题。 Java在虚拟机上运行 Java源代码并不是被编译成为普通的机器代码。而是被翻译成为虚拟机可以执行的代码。一个Java解释器最终执行这些代码。这其中没有连接的过程；解释在需要的时候动态的加载一些类； Java是完全面向对象的 Java是一种完全面向对象的语言。这意味着你对任何一个Java对象所做的动作都是通过一个方法实现的。第一点就是，再也没有没有主函数这样的孤立的东西了。取而代之的是，你必须开始用一个对象的看法看待一个程序，一个类的对象。但是这个对象又什么对象呢？大多数Java程序只是简单的通过继承Java基础类Object来实现所需要的东西，但是你可以通过创建程序基础类用于多个特性相似的应用程序来节省时间。 严格的面向对象的规定意味着理用原有的C/C 代码不可以直接不加改动的使用；系统调用也是这样的。C 中，你可以通过在C 正常的命名空间外声明extern"C"来使用原有的C的过程调用，包括系统调用。 在Java中，只有一个类似的安全回溯的方法，但是并不是十分简单的方法。你必须定义一个本地方法，其目的是为C语言提供接口，然后提供连接的介质。Java环境提供了完成这种任务的工具，但是整个过程和C 中提供的extern比微不足道，完成使用C 类的过程则更加复杂，因为这样会引入对C的借口和C函数和C 成员函数的问题。 幸运的是，许多常用的系统实用工具函数已经在系统类中的方法中

提供出来，但是这些明显没有包含经过许多年来你所创建的那些类和过程。所以，在你需要的时候你应该去钻研一下。
Java中没有独立的头文件 在Java中，关于类的一切东西都被放到一个单独的文件中。方法的位置只可能在一个地方出现，一个方法的实现必须在它的定义过程中同时进行。这样做得优点是在实现程序的时候不容易因为文件的非同步错误而失败，或者获取到一个没有实现的声明。类的声明可以被Java解释器利用甚至是从一个编译过的单元中获取，所以不再需要有头文件，只要有编译过的文件。 这样做的缺点与我们编程的过程有关。许多C 程序员喜欢用头文件来代替文档。要看一个成员函数的接口参数，只需要看头文件中的声明即可。你可以经常的看头文件即可了解怎样去使用这个类。在Java中，没有这样的总结。因为实现类方法的代码必须在方法定义的时候出现，而且，对于一个单独的函数的代码来说就经常占据了一整页乃至更多。这样，很难通过看Java的代码就初步了解类是怎样使用的。你必须为你需要的类准备足够多的文档。不言而喻，再处理非商业类库的时候文档是极度缺乏的。 在当先的Java环境中提供了两个工具来补偿这些，javap来打印类标识，javadoc为嵌入式程序提供HTML文档。
用Package来分解Java命名空间 在大的C 工程中经常遇到的一个问题是命名空间--怎样保证工程的一些程序员不会创建和另一些程序员一样名字的类？更糟糕的是，供应商可能会提供一个包含和你的类一样名字的类的库。有许多方法可以解决这一问题，但是很可能在问题发现之前工程已经启动，改正错误是需要付出许多痛苦的。 Java通过"Package"这个概念解决了这个问题，Package有效地通过通过集合类划分了命名

空间。在不同包内的两个同名的类仍然是不同的。关键问题就变成了类是否放置到相应的包中。 记住，Java并没有解决命名冲突的问题。扩展一个基类而引起了派生类的冲突。比如说，如果你最喜欢的供应商提供了一些类，然后你把它们用做基类并且派生有一个foo方法的类，当供应商提供一个新版本的类的时候就可能出现，如果供应商业也在新类中提供了一个foo的方法。 异常是Java的重要特性 在C 中，异常和异常处理是十分深奥的事情；许多C 程序员从没有处理过它们甚至不知道它们是何物。异常是在正常的过程中出现的未预料的错误，因此，它们不会从方法中返回，或者作为参数传入；但是，它们不能被忽略！这里的一个例子是计算一个书的方根的方法。正常的接口形式是将一个正数作为参数传入方法，然后方法会返回一个正实数作为结果，方法可以检验这些并且在异常产生的时候抛出异常。在大多数系统中，程序员并不是必须这样做，这样，一个没有考虑到的异常可以使程序不正常的退出。 在Java中，异常已经成为语言中非常成熟的部分。方法的说明中就包含了异常的信息，程序处理器也强制检验如果你使用了一个能够产生异常的方法，你就必须检查异常是否发生。几乎所有的Java程序员都会遇到异常的情况，因为许多非常有用的库中的类都会抛出异常。处理异常并不难，但是在一些时候是需要注意的。一个方法的文档会指明方法抛出的异常的类型。如果你忘了，不要紧，编译器会提醒你的。 字符串不再是字符数组 Java中包括了一个字符串的对象，并且是个常量。字符串不像字符数组一样，虽然可以简单的从一个字符数组构造一个字符串。你应该尽可能的用字符串代替字符数组，因为他们不会因为误操作而

被覆盖。 Java限制了常量对象和方法 在C 中，你可以正式的声明一个函数参数或者函数返回值为const类型，这样可以有效的防止对参数或者返回值的不正当修改。另外，你可以声明一个成员函数为const，表明它不可以修改任何他操作的对象。 Java支持常量操作符，只读变量，这些通过final关键字实现。但是Java没有支持强制的使一个可写变量在函数传递、返回的过程中变为只读。或者定义一个不操作修改对象的常量方法。 在Java中，这个省略带来的影响和在C 中相比就非常小了，这很大程度上因为字符串变量和字符数组的不同，但是这也带来一个引起错误的隐患。特别地，没有办法检验一个方法是否可以改动对象。 Java没有指针 理解指针的概念是一个C或C 程序员最难应付的问题。指针也是错误产生的一大根源。Java中没有指针，对象的句柄直接作为参数传递，而不是传递指针。另外，你必须通过索引使用数组。这些都不是什么大问题。然而，没有指针是在写含有函数指针或者成员函数指针的系统的时候引起很大麻烦。这个问题在处理回调函数的时候更加显著。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com