Java 有用论 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/212/2021_2022_Java__E6_9C_89_E7_94_c107_212326.htm 1.我们老祖宗说过，合久必分，分久必合。软件工程本身是希望从工程学的角度来控制工程的进度，混杂各种技术不是什么好现象，C 恐怕种类也不少，目前还有谁只写ansi C 的？看到漫天飞的makefile，各个版本的lib，不知名的头文件()，遍地的define。。。，恐怕晕的不止我一个，多有耐心的人都要消耗时间来协调各个部分的程序(呜呜，我是低手，别臭我)，写了许久，还是某个平台的(确实有人可以写出移植方便的程序，但我辈低手尚我此种眼界和经验)。提到IA32真考倒了，java和jvm最好还是分开讲，信任一群公司的联盟可以提供更好的jvm，好象不只是我这么想。当然，java很多东西做不了，可以尝试通过JNI、CORBA、Socket等技术来调用万能的C 啊。十八班武器，那个没有自己长短处？不过现在做底层应用的，还是以C为主。不信，看看那些做通讯，电子的，哪个是在用C，而且将来也不一定会转到C 。 2.java是什么？这个问题问的好，三年前，两年前，一年前，半年前，大家的理解都不同，本人第一次对这个问题有感觉有触发是在回答一份ibm调查问卷的时候。 java.sun.com/products下面那些东西，有谁看完了？我是看了几个，但起码知道跑java的设备目前已经存在了，虽然是手持设备或通讯类产品，而且不是只能跑java，但起码那也叫硬件。将硬件跟软件支持分开，这才是适应不断升级硬件、软件的最优结合方式，拘泥于技术的细节，忽略方向，这样子很被动的。 3.被他的例子击中要害。没话讲，一是不熟悉，二是我也基本同意他的看法，除了3.1.3，小小细节就不

必拖出来讲了。 4. 4.1是个老话题了。的确，java效率通常要低于c ，但是它是挂在jvm上运行的，比通常的C 运行多了个壳，而且用循环来比较的话，恐怕这种举例更不公平，显然C 对于y = x * z优化跟java的处理有不同。如果一味强调编译器优化的某项特性，那java这边也可以做优化啊。运行态编译就那么不可行？别忘了，服务器上的程序没有哪个是运行一次就被重新执行的，而执行过一次，运行态编译就会体现它的优点，一个C 程序运行一次3秒，一个java程序运行一次6秒，这就说java慢？好，那么那个c 程序运行10000次就是30000秒，那个java还是这么简单乘法吗？请考虑统计数据采样的重要性。统计本身就是们学问，加减乘除都要有凭据的。对于两者效率差多少，个人意见：在不同的应用场合下来测试，这样可能客观点。 4.2预编译java谁说没有？SQLJ白出来那么久了。运算符重载在java就有一个" "，如果C 中有反射的话，相信灵活性会有更大程度的提高。模板类都是C 的精华，我也非常喜欢，那是C 中的奇迹，ATL和STL活在一群热爱C 的程序员里面，但是其它语言又有谁支持了？类型约束，呵呵，如果程序员连类型都写错，那么在C 不需要改吗？多继承是个老话题，不过从com/dcom、corba这些技术，恐怕接口比多继承更受欢迎吧。毕竟超过二层的多继承在C 应用实例也不常见，主要原因是没必要，而且程序难调。AWT的确不是很受欢迎，4年前，人们都以为AWT就是java全部，结果多少人放弃java，但现在情况就完全不同了。java可以将程序压缩放在jar文件中，以减少文件尺寸和方便管理，不过pe的也有压缩加壳技术，但是终究不是系统级支持。 4.3扯多点东西，那位老大一定对信号量等同步技术非常重视，在java中实现非

常简单明朗，C 也麻烦，那么关于资源如何释放对于程序员就只是个编程的习惯，相信在java中忘记关掉stream的人在C也可能犯同样的错，而且结果也会相同。允许C 程序员在析构函数多写几行关闭，为什么一定认为java程序员会忘了呐？如果java是骗局，那么这是一场跨国多巨头联合超级大骗局。5.java发展时间不长，比c 是年青太多了，年青到基本框架到处都在搭建，jvm的版本也是很多，我现在使用的是1.2.2_005，对个人影响也许不大，但这个对公司的影响确实比较大，没有哪个公司愿意为jdk1.1.x和jdk1.2.x写两套软件，但这个就好象c 的库版本管理混乱一样，没有哪种技术一开始不需要伤筋动骨的改造，perl的下一版都在重写呐。 6. 6.1好象软件工程要求你在写代码前先干点什么，这个应该不算java的问题。是采用如何的方式学习和应用语言开发的问题。在工程里面说明一遍，然后再在header里面写一清遍，恐怕算是浪费体力吧。javadoc强烈推荐大家使用，可以根据代码生成象jdk那种结构清晰的文档，写完程序，这种文档也可以作为补充。 6.2呵呵，又是多继承。实例举多了，不利于帮助了解事物的本质。 6.3析构，呵呵，析构中再调用其它的析构，好处可以自己定制析构，但是对于调试程序恐怕也会带来多一些麻烦，这个世界上的程序员还是要靠debug来吃饭的。析构到底有那么神奇，令仁兄接二连三提起？ 6.4许多人都讨论过一个大对象好，还是一堆小对象好，那么本人的看法是如果一个大对象完成所有工作，那么就将问题局部化，调试范围集中在一个文件，但不利于多人开发，而且功能升级也不方便，重用的可能更小；如果多个小对象，可以分工明确，功能升级有可能减少代码的修改量，最大可能重用组件，对于项目管理

的确是个挑战，看似简单的小对象之间的架构才是component比oo多出来的精髓，但使用不当会令工作事倍功半。这个问题对c 和java都成立，但是由于语言的风格不同，所以受重视的程度有不同，许多人在学习java的过程里面都会思索、比较这两种模式。 6.5移植的东西就很难讲。ibm和oracle都要在palm上搞数据库，ibm是改写的，oracle是重写的，那么哪个服务器软件不是一大堆for各个版本的不同操作系统。这些就是摆在眼前d的实际问题。不知不觉，64位又要大行其道了。 Fortran77和basic现在也有人用，甚至cobol用的人都不少，为什么？商业社会不允许随便放弃过去，推C 的时候面临的是同样问题，怎么就这么快忘了呐？ java的基本类型比较少，这个是事实，之间转换也不是非常灵活，但是c 中，类似整除和取余更是通过函数实现的(顺便提下，基本上c 中每一个内置函数都有功能相同名字、类型不同的，这些恐怕也不是什么好现象吧)。细节问题，不至于影响对java发展方向的评价。 全局变量这种东西可以放在一个类中包装啊，c 也是希望减少使用全局变量、宏编译，联合体出现的年代，String都不是数据类型，操纵起来也要处处小心。现在用联合体的场合一般是厂家提供的api需要，自己写程序用联合体的不多。用联合的语言就一定要支持指针，这是java所不愿走的路。instanceof确实非常有用，因为java可以通过非常多的形式拿到类的实例，如果不知道是什么类或者实现了什么接口(别咬文嚼字，接口本身就是一种特殊的类)，可以让java的灵活程度大大加强。比起dll的动态加载，java的动态加载就是天生的，不需要那么多乱七八糟的规矩，而且可以通过jar来集中管理同属一类的java class。 反射是java的一大特色。 6.6呵呵，java

可以骑着c/c 啊，jni是吃白饭的？通过jdbc可以调用数据库中的东西，jndi可以访问目录，jms可以访问消息队列，jts可以访问交易系统，javacard可以在sim卡上跑，corba可以调用任何支持corba的东西，socket可以调用任何支持ip的语言，rmi虽然只能用java，但是可以非常透明的得到实例，这是其它语言不具备的优势，适合网络分布计算。呵呵，这些只是非常普及的java技术之一小部分，它虽然不是什么崇高的理论，但是起码它能让许多计算机在一起工作，为应用服务，这才是我们最关心的问题。 7.偏系统底层开发，我宁可用C，中间层开发倒是java，界面用些简单工具就算了。至于c ，恐怕只是当初在接触面向对象时看到的一个诱人的梦了。 出国留学移民教育考试出国,留学,移民,澳洲,澳大利亚,加拿大,英国,美国,法国,日本,新西兰 1.我们老祖宗说过，合久必分，分久必合。软件工程本身是希望从工程学的角度来控制工程的进度，混杂各种技术不是什么好现象，C 恐怕种类也不少，目前还有谁只写ansi C 的？看到漫天飞的makefile，各个版本的lib，不知名的头文件()，遍地的define。。。，恐怕晕的不止我一个，多有耐心的人都要消耗时间来协调各个部分的程序(呜呜，我是低手，别臭我)，写了许久，还是某个平台的(确实有人可以写出移植方便的程序，但我辈低手尚我此种眼界和经验)。提到IA32真考倒了，java和jvm最好还是分开讲，信任一群公司的联盟可以提供更好的jvm，好象不只是我这么想。当然，java很多东西做不了，可以尝试通过JNI、CORBA、Socket等技术来调用万能的C 啊。十八班武器，那个没有自己长短处？不过现在做底层应用的，还是以C为主。不信，看看那些做通讯，电子的，哪个是在用C，而且将来也不一定会转

到C。 2.java是什么？这个问题问的好，三年前，两年前，一年前，半年前，大家的理解都不同，本人第一次对这个问题有感觉有触发是在回答一份ibm调查问卷的时候。java.sun.com/products下面那些东西，有谁看完了？我是看了几个，但起码知道跑java的设备目前已经存在了，虽然是手持设备或通讯类产品，而且不是只能跑java，但起码那也叫硬件。将硬件跟软件支持分开，这才是适应不断升级硬件、软件的最优结合方式，拘泥于技术的细节，忽略方向，这样子很被动的。 3.被他的例子击中要害。没话讲，一是不熟悉，二是我也基本同意他的看法，除了3.1.3，小小细节就不必拖出来讲了。 4. 4.1是个老话题了。的确，java效率通常要低于c，但是它是挂在jvm上运行的，比通常的C 运行多了个壳，而且用循环来比较的话，恐怕这种举例更不公平，显然C 对于y = x * z优化跟java的处理有不同。如果一味强调编译器优化的某项特性，那java这边也可以做优化啊。运行态编译就那么不可行？别忘了，服务器上的程序没有哪个是运行一次就被重新执行的，而执行过一次，运行态编译就会体现它的优点，一个C 程序运行一次3秒，一个java程序运行一次6秒，这就说java慢？好，那么那个c 程序运行10000次就是30000秒，那个java还是这么简单乘法吗？请考虑统计数据采样的重要性。统计本身就是们学问，加减乘除都要有凭据的。对于两者效率差多少，个人意见：在不同的应用场合下来测试，这样可能客观点。 4.2预编译java谁说没有？SQLJ白出来那么久了。运算符重载在java就有一个" "，如果C 中有反射的话，相信灵活性会有更大程度的提高。模板类都是C 的精华，我也非常喜欢，那是C 中的奇迹，ATL和STL活在一群热爱C 的程序员

里面，但是其它语言又有谁支持了？类型约束，呵呵，如果程序员连类型都写错，那么在C 不需要改吗？多继承是个老话题，不过从com/dcom、corba这些技术，恐怕接口比多继承更受欢迎吧。毕竟超过二层的多继承在C 应用实例也不常见，主要原因是没必要，而且程序难调。AWT的确不是很受欢迎，4年前，人们都以为AWT就是java全部，结果多少人放弃java，但现在情况就完全不同了。java可以将程序压缩放在jar文件中，以减少文件尺寸和方便管理，不过pe的也有压缩加壳技术，但是终究不是系统级支持。 4.3扯多点东西，那位老大一定对信号量等同步技术非常重视，在java中实现非常简单明朗，C 也麻烦，那么关于资源如何释放对于程序员就只是个编程的习惯，相信在java中忘记关掉stream的人在C 也可能犯同样的错，而且结果也会相同。允许C 程序员在析构函数多写几行关闭，为什么一定认为java程序员会忘了呐？如果java是骗局，那么这是一场跨国多巨头联合超级大骗局。 5.java发展时间不长，比c 是年青太多了，年青到基本框架到处都在搭建，jvm的版本也是很多，我现在使用的是1.2.2_005，对个人影响也许不大，但这个对公司的影响确实比较大，没有哪个公司愿意为jdk1.1.x和jdk1.2.x写两套软件，但这个就好象c 的库版本管理混乱一样，没有哪种技术一开始不需要伤筋动骨的改造，perl的下一版都在重写呐。 6. 6.1好象软件工程要求你在写代码前先干点什么，这个应该不算java的问题。是采用如何的方式学习和应用语言开发的问题。在工程里面说明一遍，然后再在header里面写一清遍，恐怕算是浪费体力吧。javadoc强烈推荐大家使用，可以根据代码生成象jdk那种结构清晰的文档，写完程序，这种文档也可以作为补充。 6.2

呵呵，又是多继承。实例举多了，不利于帮助了解事物的本质。 6.3析构，呵呵，析构中再调用其它的析构，好处可以自己定制析构，但是对于调试程序恐怕也会带来多一些麻烦，这个世界上的程序员还是要靠debug来吃饭的。析构到底有那么神奇，令仁兄接二连三提起？ 6.4许多人都讨论过一个大对象好，还是一堆小对象好，那么本人的看法是如果一个大对象完成所有工作，那么就将问题局部化，调试范围集中在一个文件，但不利于多人开发，而且功能升级也不方便，重用的可能更小；如果多个小对象，可以分工明确，功能升级有可能减少代码的修改量，最大可能重用组件，对于项目管理的确是个挑战，看似简单的小对象之间的架构才是component比oo多出来的精髓，但使用不当会令工作事倍功半。这个问题对c 和java都成立，但是由于语言的风格不同，所以受重视的程度有不同，许多人在学习java的过程里面都会思索、比较这两种模式。 6.5移植的东西就很难讲。ibm和oracle都要在palm上搞数据库，ibm是改写的，oracle是重写的，那么哪个服务器软件不是一大堆for各个版本的不同操作系统。这些就是摆在眼前d的实际问题。不知不觉，64位又要大行其道了。 Fortran77和basic现在也有人用，甚至cobol用的人都不少，为什么？商业社会不允许随便放弃过去，推C 的时候面临的是同样问题，怎么就这么快忘了呐？ java的基本类型比较少，这个是事实，之间转换也不是非常灵活，但是c 中，类似整除和取余更是通过函数实现的(顺便提下，基本上c 中每一个内置函数都有功能相同名字、类型不同的，这些恐怕也不是什么好现象吧)。细节问题，不至于影响对java发展方向的评价。 全局变量这种东西可以放在一个类中包装啊，c 也是希

望减少使用全局变量、宏编译，联合体出现的年代，String都不是数据类型，操纵起来也要处处小心。现在用联合体的场合一般是厂家提供的api需要，自己写程序用联合体的不多。用联合的语言就一定要支持指针，这是java所不愿走的路。instanceof确实非常有用，因为java可以通过非常多的形式拿到类的实例，如果不知道是什么类或者实现了什么接口(别咬文嚼字，接口本身就是一种特殊的类)，可以让java的灵活程度大大加强。比起dll的动态加载，java的动态加载就是天生的，不需要那么多乱七八糟的规矩，而且可以通过jar来集中管理同属一类的java class。 反射是java的一大特色。 6.6呵呵，java可以骑着c/c 啊，jni是吃白饭的？通过jdbc可以调用数据库中的东西，jndi可以访问目录，jms可以访问消息队列，jts可以访问交易系统，javacard可以在sim卡上跑，corba可以调用任何支持corba的东西，socket可以调用任何支持ip的语言，rmi虽然只能用java，但是可以非常透明的得到实例，这是其它语言不具备的优势，适合网络分布计算。呵呵，这些只是非常普及的java技术之一小部分，它虽然不是什么崇高的理论，但是起码它能让许多计算机在一起工作，为应用服务，这才是我们最关心的问题。 7.偏系统底层开发，我宁可用C，中间层开发倒是java，界面用些简单工具就算了。至于c ，恐怕只是当初在接触面向对象时看到的一个诱人的梦了。 