1. Java是编译型语言还是解释型语言?

是解释型

定义： 编译型语言：把做好的源程序全部编译成二进制代码的可运行程序。然后，可直接运行这个程序。 解释型语言：把做好的源程序翻译一句，然后执行一句，直至结束！ 区别： 编译型语言，执行速度快、效率高；依靠编译器、跨平台性差些。 解释型语言，执行速度慢、效率低；依靠解释器、跨平台性好。 个人认为，java是解释型的语言，因为虽然java也需要编译，编译成.class文件， 但是并不是机器可以识别的语言，而是字节码，最终还是需要 jvm的解释，才能在各个平台执行， 这同时也是java跨平台的原因。所以可是说java即是编译型的，也是解释型， 但是假如非要归类的话，恐怕java应该归到解释型的语言中。 附： 编译型的语言包括：C、C++、Delphi、Pascal、Fortran 解释型的语言包括：Java、Basic、javascript

Java语言是跨平台的，先编译后解释，编译是通过编译器完成的，解释执行是通过JVM完成的， JVM是java虚拟机，跨平台用的，主要作用是解释编译后生成的字节码文件--class，

2. 我想知道JVM是以什么形式存在的，是包含在jdk中吗？他的文件叫什么名字?

安装bin文件中的java.exe，操作系统装入jvm是通过jdk中java.exe来完成在jdk里面,是javaw.exe,因为每次eclipse启动,都有javaw.exe这个进程,简单来说Sun java提供的JDK中包含有JVM，是其中的一个组成部分，更详细的看下面：

在我们运行和调试java程序的时候,经常会提到一个jvm的概念.jvm是java程序运行的环境,但是他同时一个操作系统的一个应用程序一个进程,因此他也有他自己的运行的生命周期,也有自己的代码和数据空间. 首先来说一下jdk这个东西,不管你是初学者还是高手,是j2ee程序员还是j2se程序员,jdk总是在帮我们做一些事情.我们在了解java之前首先大师们会给我们提供说jdk这个东西.它在java整个体系中充当着什么角色呢?我很惊叹sun大师们设计天才,能把一个如此完整的体系结构化的如此完美.jdk在这个体系中充当一个生产加工中心,产生所有的数据输出,是所有指令和战略的执行中心.本身它提供了java的完整方案,可以开发目前java能支持的所有应用和系统程序.这里说一个问题,大家会问,那为什么还有j2me,j2ee这些东西,这两个东西目的很简单,分别用来简化各自领域内的开发和构建过程.jdk除了jvm之外,还有一些核心的API,集成API,用户工具,开发技术,开发工具和API等组成 好了,废话说了那么多,来点于主题相关的东西吧.jvm在整个jdk中处于最底层,负责于操作系统的交互,用来屏蔽操作系统环境,提供一个完整的java运行环境,因此也就虚拟计算机. 操作系统装入jvm是通过jdk中java.exe来完成,通过下面4步来完成jvm环境.

3. **操作系统装入jvm是通过jdk中java.exe来完成,通过下面4步来完成jvm环境.**
    1. 创建jvm装载环境和配置
    2. 装载jvm.dll
    3. 初始化jvm.dll并挂界到JNIENV(JNI调用接口)实例
    4. 调用JNIEnv实例装载并处理class类。

一. jvm装入环境

jvm提供的方式是操作系统的动态连接文件. 既然是文件那就一个装入路径的问题，java是怎么找这个路径的呢？当你在调用java test的时候，操作系统会在path下在你的java.exe程序，java.exe就通过下面一个过程来确定jvm的路径和相关的参数配置了. 下面基于windows的实现的分析. 首先查找jre路径，java是通过GetApplicationHome api来获得当前的java.exe绝对路径，c:\j2sdk1.4.2_09\bin\java.exe,那么它会截取到绝对路径c:\j2sdk1.4.2_09\，判断c:\j2sdk1.4.2_09\bin\java.dll文件是否存在，如果存在就把c:\j2sdk1.4.2_09\作为jre路径，如果不存在则判断c:\j2sdk1.4.2_09\jre\bin\java.dll是否存在，如果存在这

c:\j2sdk1.4.2_09\jre作为jre路径．如果不存在调用GetPublicJREHome查HKEY_LOCAL_MACHINE\Software\JavaSoft\Java Runtime Environment\“当前JRE版本号”\JavaHome的路径为jre路径。 然后装载jvm.cfg文件JRE路径+\lib+\ARCH（CPU构架）+\jvm.cfgARCH（CPU构架）的判断是通过java_md.c中GetArch函数判断的，该函数中windows平台只有两种情况：WIN64的‘ia64’，其他情况都为‘i386’。以我的为例：C:\j2sdk1.4.2_09\jre\lib\i386\jvm.cfg.主要的内容如下：

- client KNOWN
- server KNOWN
- hotspot ALIASED_TO -client
- classic WARN
- native ERROR
- green ERROR

在我们的jdk目录中jre\bin\server和jre\bin\client都有jvm.dll文件存在，而java正是通过jvm.cfg配置文件来管理这些不同版本的jvm.dll的．通过文件我们可以定义目前jdk中支持那些jvm,前面部分（client）是jvm名称，后面是参数，KNOWN表示jvm存在，ALIASED_TO表示给别的jvm取一个别名，WARN表示不存在时找一个jvm替代，ERROR表示不存在抛出异常．在运行java XXX时，java.exe会通过CheckJvmType来检查当前的jvm类型，java可以通过两种参数的方式来指定具体的jvm类型，一种按照jvm.cfg文件中的jvm名称指定，第二种方法是直接指定，它们执行的方法分别是“java -J”、“java -XXaltjvm=”或“java -J-XXaltjvm=”。如果是第一种参数传递方式，CheckJvmType函数会取参数‘-J’后面的jvm名称，然后从已知的jvm配置参数中查找如果找到同名的则去掉该jvm名称前的‘-’直接返回该值；而第二种方法，会直接返回“-XXaltjvm=”或“-J-XXaltjvm=”后面的jvm类型名称；如果在运行java时未指定上面两种方法中的任一一种参数，CheckJvmType会取配置文件中第一个配置中的jvm名称，去掉名称前面的‘-’返回该值。CheckJvmType函数的这个返回值会在下面的函数中汇同jre路径组合成jvm.dll的绝对路径。如果没有指定这会使用jvm.cfg中第一个定义的jvm.可以通过set _JAVA_LAUNCHER_DEBUG=1在控制台上测试． 最后获得jvm.dll的路径，JRE路径+\bin+\jvm类型字符串+\jvm.dll就是jvm的文件路径了，但是如果在调用java程序时用-XXaltjvm=参数指定的路径path,就直接用path+\jvm.dll文件做为jvm.dll的文件路径．

二．**装载jvm.dll**

通过第一步已经找到了jvm的路径，java通过LoadJavaVM来装入jvm.dll文件．装入工作很简单就是调用windows API函数： LoadLibrary装载jvm.dll动态连接库．然后把jvm.dll中的导出函数JNI_CreateJavaVM和JNI_GetDefaultJavaVMInitArgs挂接到InvocationFunctions变量的CreateJavaVM和GetDefaultJavaVMInitArgs函数指针变量上。jvm.dll的装载工作宣告完成。

三．**初始化jvm**

获得本地调用接口，这样就可以在java中调用jvm的函数了．调用InvocationFunctions -> CreateJavaVM也就是jvm中JNI_CreateJavaVM方法获得JNIEnv结构的实例．

四．**运行java程序** java程序有两种方式一种是jar包，一种是class．运行jar,java -jar XXX.jar运行的时候，java.exe调用GetMainClassName函数，该函数先获得JNIEnv实例然后调用java类java.util.jar.JarFileJNIEnv中方法getManifest()并从返回的Manifest对象中取getAttributes("Main-Class")的值即jar包中文件：META-INF/MANIFEST.MF指定的Main-Class的主类名作为运行的主类。之后main函数会调用java.c中LoadClass方法装载该主类（使用JNIEnv实例的FindClass）。main函数直接调用java.c中LoadClass方法装载该类。如果是执行class方法。main函数直接调用java.c中LoadClass方法装载该类。

然后main函数调用JNIEnv实例的GetStaticMethodID方法查找装载的class主类中 "public static void main(String[] args)"方法，并判断该方法是否为public方法，然后调用JNIEnv实例的CallStaticVoidMethod方法调用该java类的main方法