帮你分析android与java的关系

Android是否相当于Java? 请注意，我并没有说**相等**，我说的是**相当**，就像P = NP里的那样。

相当的类/字节码格式　在很多层面上，Android和Java都有明显的相当。Android应用程序是用Java(TM)语言写成的，使用JDK的*javac*(或等效工具，例如ECJ)来编译。这个过程产生标准的Java字节码（*.class*文件）。这些文件再转化成Android的*.dex*文件，从使用的角度来看，它就是一种不同格式的Java class文件。不错，这是一种更优秀的格式；对Sun自从1994年以来的设计有了很大的改进。但就如你可以把一个GIF格式的图片转换成更高级的完美的完全等效的PNG格式，尽管它们的字节流完全的不同。

等效的文件格式在细节的实现上非常的不同，主要是为了优化。就好比，如果我们简单的满足于低效率的视频数据流，没有采用高端的、跨不同框架的压缩技术，那我们就可以避免跟MPEGLA视频解码专利做斗争的麻烦了。Android特异的classfile设计有好几种动机；而为了避免和Sun的知识产权保护冲突显然是一个主要的因素。不管怎样，Google并没有走的离Java足够远。两种文件格式非常的类似。它们在特定的底层数据结构上有区别，但这些结构体在语法上一致的，存储完全相同的信息。我相信在JavaSE或JavaME VM里可以轻易的在它们的系统classloader里添加一个*.dex*分析器来加载"Android classes"。

Android SDK **依赖于***.java -> .class -> .dex* 转换的事实情况既微不足道也毫无损失。"毫无损失"的事实很重要：当GIF = PNG 时，跟受损的JPG文件就不等了 —— 它解码不出完全相同的信息。如果JVM和Dalvik都各自独立，你很难写出一个相对简单的工具将一种编译过的代码转换成另一种 —— 而且不做任何妥协：不丢失信息，不使用冗余来补偿某种特征在一种VM中是first-class而在另一种中却不是的情况，不需要额外的runtime层 上在一种VM中实现另一种VM的核心API。

*(*我知道dx转换器有多么的复杂。我看过它的源代码。那个字节码转换器是一个巨大的，全功能的反编译/重编译器，通 过*SSA*构造完成。但是这个转换器在概念上仍然是无足轻重的；从*Java*字节码到*Dalvik*字节码的映射在设计上是很平滑的。堆栈相对于寄存器架构中细 节上进行了优化；而重要的东西，例如*VM*层的类型系统是完全一致的。*)*

VM相当　这Dalvik 和 JVM 的相当也是很容易看清楚的。并不只是源代码或字节码格式上的问题：它们的runtime对等物上也一样。一但一个"Android class"被加载到Dalvik VM里，**它就会像Java class一样运行，像Java class一样工作。** 如果你懂得Java编程(深入到高级的，底层的细节)，你也就懂得Android编程。你只需要学一些新的API和框架概念。他们是对等的系统。

是否记得微软的.NET? 当.NET刚出世时，Java阵营迅速的反击指责.NET是对Java的剽窃。我也是其中的一份子，但今天我看问题更清楚了。是的，它过去是个严重的剽窃产品；C# 1.0 就是一个... 区分一种语言和另一种语言最简单的方法就是看它们的惯用风格 —— 例如*toString()* 相对于 *ToString()*。 但在最重要的VM规范里，微软做了很大的功课。它的CLR,CLI,和核心框架，都非常的不同于Java，所以我们不能说JVM = CLR这个等式。你不可能使用一个简单的文件格式转换工具把你编译好的Java class转换成能在.NET runtime上运行的代码。

要证据吗？你只需看一看IKVM就知道了。这是一个非常有趣的项目，它能够使Java和.NET跨平台兼容，于是，你的Java代码可以在不做修改的情况下在CLR（或者是等效的.NET runtime，比如Mono)上运行... 但IKVM**并不是**一个简单的、类*dx*的 文件格式转换器。对Java class的转化、对Java核心API的适配，都是十分的复杂，即使对一个简单的HelloWorld程序也是这样。各个平台的内部机制，如反射，安 全，并行，异常处理，字节码验证，I/O，以及其它核心API，特征上大致相同，但是在细节上完全不同，一些死胡同的情况会迫使IKVM不得不钻越一个又 一个的火圈来让Java代码运行到了.NET VM上。它需要依赖于一个巨大的额外的runtime层，来适配从OpenJDK源代码里来的完整JavaSE API。我大致的关注IKVM的开发已经有数年了 —— 我阅读这精彩的IKVM 博客 – 所以我完全清楚他们为了让Java程序和JavaSE应用适配到.NET上所做的巨大的努力。(这项工作仍然没有完工；而且很多部分都需要以丧失某些性能为代价。)

*(*老的*Visual J++ Visual J#* 也不是一个简单的 *Java-to-.NET* 转换器。我不想讨论它，但我们完全可以说*Visual J#* 对*Java*的兼容并不比最早期的*IKVM*强多少。*)*

我把P = NP引进来了讨论；有些人把图灵等效(Turing-equivalence)理论引进来，说任何图灵完备的平台/语言/VM都是相互等价的。这也没错， 但与本论题无关。图灵模型这种方式太泛化了；使用这种表面价值来考量会把更个软件专利系统摧毁(尽管这不是个坏事！)。我们需要在地上为JVM等效画条 线，一条更接近实用需求而远离图灵等效的线。按我的观点，这微不足道的二进制格式转换，穷尽的高层源代码和runtime的兼容，使Android明显的处于Java等效的这条线内。

APIs 和 Runtime 相当　Android使用了一个相当大的JavaSE APIs子集。这些APIs (来自于Harmony项目)都是全新

的实现，但它们是以JavaSE为模子。如果不是因为TCK许可证问题，Harmony完全可以取得JavaSE认 证。但这并没有改变这样的一个事实：Harmony 和 JavaSE APIs是 完全的等效的 —— 这是特意的，不是偶然的。就像Charles Nutter——有名的JRuby人物——最近写道:

> *Android*支持一个不完整的（但相当大的）*Java 1.5* 类库子集。这个子集大到**一个复杂的*JRuby*项目几乎不经任何修改就能在*Android*上运行，**很少有限制情况。

看起来Dalvik对JVM是如此的接近，它不得不完全兼容大部分的JVM规范，包括完全详细的JMM (就像Android支持Java风格的线程和并发，已经深入到了高级的*java.util.concurrent*包里了)。可为什么有如此多的"Dalvik是个新VM"或"Dalvik不能运行Java类"的说法呢(90%的讨论这场诉讼的论坛和博客都持这种观点)。

最后的思考　这篇博客并不是关于Oracle和Google诉讼官司的法律依据的。我将会忽略（我会删除掉）那些跑题的评论（跟Android = Java不相关的话）。我只是讨厌那些"*Android*跟*Java*完全没关系"的胡说八道;Google和 Android的拥护者必须要找一个更有意义的论据。

*(*我将拭目以待这场官司的进展，带着我所有的预见，直到所有细节和最终结果都出来。除非你有内部消息（我没有），**不要太天真。** 保持冷静。 我们并不知道*Oracle*的 —— 或 *Google*的 —— 真正的全部动机和计划。我们并不知道这荧幕背后的故事，自从*2007*年*Google*首次宣告*Android*的诞生*(*这导致了*JavaME*生态环境的崩 溃*), Sun*就痛恨不已，但最后还是不得不夹着尾巴行事。我不相信任何一个有*10*亿美金的股东控股公司会有利他主义的动机：*Google*不会，*Oracle*不 会，即使我喜爱的老的*Sun*公司也不会。我们等着看吧。*)*

我不相信Google没有能力创造出一种既不背离Java太远，又以Java风格为基础的平台（就像.NET做的那样）。Dalvik，以及Android框架，它们可能是在权衡了与大量的现有的Java程序，类库，Java天才，和Java工具链高度兼容的愿望的最后结 果。微软在一咬牙一跺脚后放弃了现成移植Java带来的好处，创造了全新的.NET。Google没有这样做。

这个Android = Java等式显然并不是包括所有的东西（不是一一对应的）。每种平台都有自己一些独特的API，当然，Android是一个完整的操作系统，包括一个 Linux-based的内核，图形系统和电信堆栈，等等。很显然，我只是谈论其中最常用的部分：Java为中心的用户使用区/依赖于Java源代码、 Java classes（切不管什么格式）、Java APIs（包括成千上万的常用JavaSE APIs)和出色的类Java的虚拟机的应用框架。对于Android和其它的Java平台之间的关系有个准确的说法，就是使用版本的概念。我曾记得有个 博客说过这样的话"*Android*里没有*J*"。那么，我现在说也不晚：我建议把Android改名为Java GE（Java Google Edition）。这样一来就再也不会导致混淆了