Java 作为一门高级程序语言，它的语法非常复杂，抽象程度也很高。因此直接在硬件上运行这种复杂的程序并不现实。

Java 虚拟机具体是怎样运行 Java 字节码的?

执行 Java 代码首先需要将它编译而成的 class 字节码文件加载到 Java 虚拟机中。加载后的 Java 类会被存放于方法区。

Java 虚拟机同样也在内存中划分出堆和栈来存储运行时数据。Java 虚拟机会将栈细分为面向 Java 方法的 Java 方法栈，面向本地方法（用 C++ 写的 native 方法）的本地方法栈，以及存放各个线程执行位置的 PC 寄存器。

在运行过程中，每当调用进入一个 Java 方法，Java 虚拟机会在当前线程的 Java 方法栈中生成一个栈帧，用以存放局部变量以及字节码的操作数。这个栈帧的大小是提前计算好的，而且 Java 虚拟机不要求栈帧在内存空间里连续分布。

当退出当前执行的方法时，不管是正常返回还是异常返回，Java 虚拟机均会弹出当前线程的当前栈帧，并将之舍弃。

Java 字节码无法直接执行，因此 JVM 需要将字节码翻译成机器码。在 hotspot 里，上述翻译过程有两种形式：

- 第一种是解释执行，即逐条将字节码翻译成机器码并执行；
- 第二种是即时编译 *(Just-In-Time compilation, JIT)*，即将一个方法中包含的所有字节码编译成机器码后再执行。

JVM 会先解释执行成字节码，而后将其中反复执行的热点代码，以方法为单位进行即时编译。

hotspot 内置了多个即时编译器：C1、C2 和 Graal。引入多个即时编译器的原因是为了在编译时间和生成代码的执行效率之间进行取舍。C1 是 Client 编译器，采用的优化手段相对简单，因此编译时间较短；C2 是 Server 编译器，采用的优化手段相对复杂，因此编译时间较长。

从 JDK7 开始，hotspot 默认采用分层编译的方式：热点方法首先会被 C1 编译，而后热点方法中的热点会进一步被 C2 编译。为了不干扰应用的正常运行，hotspot 的即时编译是放在额外的编译线程中进行的，hotspot 会根据 CPU 的数量设置编译线程的数目，并且按 1:2 的比例配置位 C1 及 C2 编译器。

1