1. 高级语言的运行机制

我们编程都是用的高级语言(写汇编和机器语言的大牛们除外)，计算机不能直接理解高级语言，只能理解和运行机器语言，所以必须要把高级语言翻译成机器语言，计算机才能运行高级语言所编写的程序。

翻译，其实翻译的方式有两种，一个是编译，一个是解释。两种方式只是翻译的时间不同。

(1). 编译型语言 （如C、C++）

使用专门的编译器，针对特定平台（操作系统）将某种高级语言源代码一次性”翻译“成可被该平台硬件执行的机器码（包括机器指令和操作数），并包装成该平台所能识别的可执行性程序的格式。

优点: 运行效率较高
缺点: 编译生成的程序无法移植

(2). 解释型语言 （如Ruby、Python）

使用专门的解释器对源程序逐行解释成特定平台的机器码并立即执行的语言。解释型语言通常不会整体性的编译和链接处理，解释型语言相当于把编译型语言中的编译和解释过程混合到一起同时完成。

优点: 跨平台较容易
缺点: 效率较低，不能脱离解释器独立运行

2. Java语言的运行机制

(1). Java是编译型和解释型语言的结合体。

- 首先采用通用的java编译器将java源程序编译成为与平台无关的字节码文件（class文件）
- 然后由java虚拟机（JVM）对字节码文件解释执行。

注意：java字节码具有平台无关性、可以在各种不同系统平台中运行，但是需要有不同版本的java虚拟机，不同系统平台的java运行环境其java虚拟机是不一样的。

(2). JVM(Java Virtual Machine Java虚拟机)

- JVM是Java字节码执行的引擎，为java程序的执行提供必要的支持，还能优化java字节码，使之转换成效率更高的机器指令。程序员编写的程序最终都要在JVM上执行，JVM中类的装载是由类加载器（ClassLoader）和它的子类来实现的。
- ClassLoader是java运行时一个重要的系统组件，负责在运行时查找和装入类文件的类。
- JVM屏蔽了与具体操作系统平台相关的信息，从而实现了java程序只需生成在JVM上运行的字节码文件（class文件），就可以在多种平台上不加修改地运行。不同平台对应着不同的JVM，在执行字节码时，JVM负责将每一条要执行的字节码送给解释器，解释器再将其翻译成特定平台环境的机器指令并执行。java语言最重要的特点就是跨平台运行，使用JVM就是为了支持与操作系统无关，实现跨平台运行。

(3). ClassLoader

- 是JVM实现的一部分，包括bootstrapclassloader（启动类加载器）
- ClassLoader在JVM运行的时候加载java核心的API，通过java程序实现两个ClassLoader:
  - ExtClassLoader，它的作用是用来加载java的扩展API，也就是lib\ext类；
  - AppClassLoader，用来加载用户机器上CLASSPATH设置目录中的Class.
- ClassLoader加载流程：当运行一个程序的时候，JVM启动，运行bootstrapclassloader，该ClassLoader加载java核心API，然后调用ExtClassLoader加载扩展API，最后AppClassLoader加载CLASSPATH目录下定义的Class.

(4). JRE

JRE是JavaRuntimeEnvironment，java运行时环境，它是java程序运行所必须的环境集合，主要由java虚拟机、java平台核心类和若干支持文件组成。其不包含开发工具、编译器、调试器以及其他工具。

在执行java程序的时候，操作系统会按照下面顺序寻找JRE环境。

- 先查找当前目录下有没有JRE
- 再查找父目录下有没有JRE
- 接着在环境变量PATH制定的路径中查找JRE
- 注册表查看CurrentVersion的键值指向哪个JRE

JRE自带的基础类库主要在JRE\LIB\rt.jar文件中。在程序运行时，JRE由ClassLoader(类加载器)负责查找和加载程序引用到的基类库和其他类库。基础类库，Classloader会自动到rt.jar的位置；其他的类库，ClassLoader在环境变量CLASSPATH制定的路径中搜索。

(5). JDK

- JDK是Java Development Kit，简称java开发工具包。
- JDK是java的核心。它包括java运行环境、一堆java工具盒java基础的类库（rt.jar）。
- JDK包含JRE的全部内容外，还包含开发者用以编译、调试和运行java程序的工具。
- JDK、JRE、JVM之间的关系：JDK、JRE、JVM之间是包含关系。范围由大到小依次为JDK、JRE、JVM。