1. **JDK、JRE、JVM是什么，作用又是什么？**

**JDK(Java Development Kit) ：**

Java Development Kit，即Java开发工具包，里面集成了整个Java开发的核心，JDK包含JRE（JVM+系统类库+支持文件）和Java工具。

**JRE(Java Runtime Environment) ：**

Java Runtime Environment， 即Java运行时环境，是Java程序运行的核心，JRE包含了Java虚拟机（JVM）、Java核心类库和支持文件。

**JVM (Java Virtual Machine) ：**

Java Virtual Machine，即Java虚拟机，是Java程序跨平台实现的核心（可看作简易操作系统），所有的Java程序会首先被编译成 .class 字节码文件，根据字节码指令进行识别并调用上层操作系统。

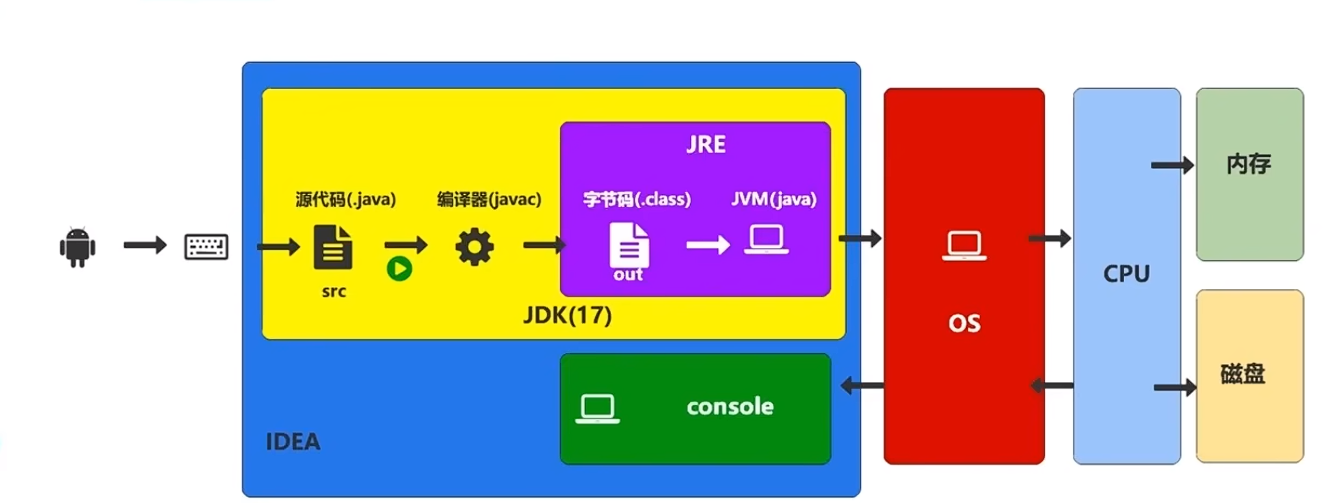


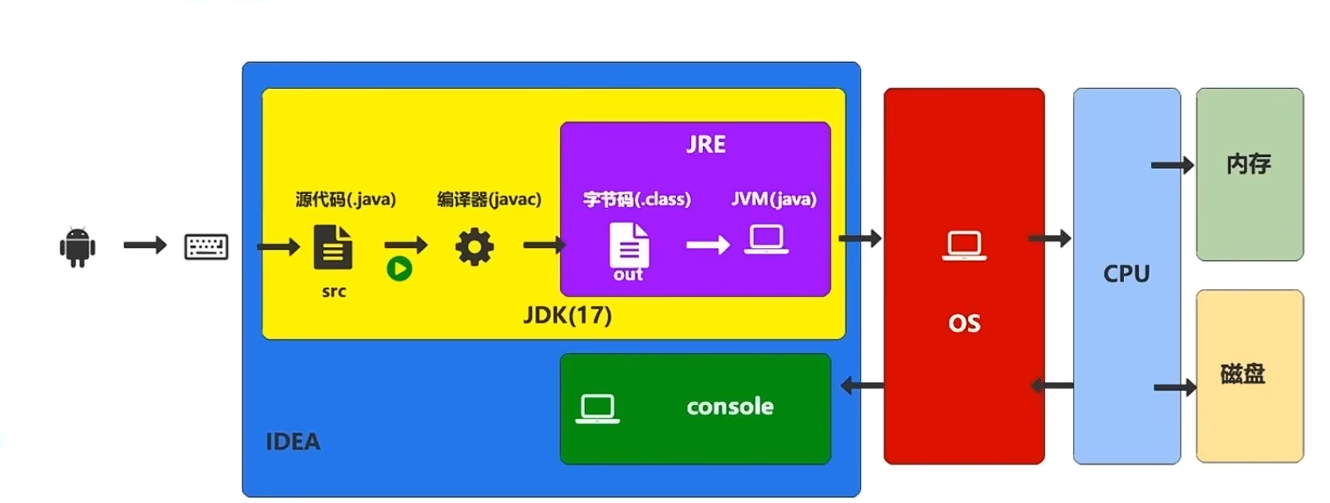
图1.java运行时流程

***总结 ：***

* ***JRE*** = ***JVM*** + 基本类库（系统类库） + 支持文件
* ***JDK*** = ***JRE*** + 开发工具包
* ***Java***核心优势：一次编译，到处运行，其跨平台运行的核心在于 ***JVM***

1. 请简述Java语言的特点
2. **简单性**：Java的设计目的是使语言易于学习和使用，避免复杂的特性如指针操作。
3. **面向对象**：Java完全支持面向对象的编程概念，如类、对象、继承、封装、多态等。
4. **健壮性**：Java通过严格的编译时检查和运行时异常处理机制来提高程序的健壮性。
5. **安全性**：Java提供了多种安全机制，比如沙箱模型可以防止恶意代码访问系统资源。
6. **平台无关性**：Java编写的程序可以在任何安装了Java虚拟机（JVM）的平台上运行，这得益于“一次编写，到处运行”（WORA）的理念。
7. **可移植性**：由于Java的平台无关性，一次编译，到处运行，使得它很容易从一个平台移植到另一个平台。
8. **解释型与编译型**：Java源代码首先被编译成字节码，然后由JVM解释执行，这种模式结合了解释型语言的灵活性和编译型语言的高效性。
9. **高性能**：虽然Java最初因为解释执行而被认为较慢，但随着JIT（Just-In-Time）编译器的发展，其性能得到了显著提升。
10. **多线程**：Java内置对多线程的支持，可以开发并发应用程序，有效利用多核处理器的能力。
11. **描述Java程序编写到运行的过程**

用户编译.java文件，通过javac编为字节码.class文件后进入jre，通过Java虚拟机JVM进行运行显示到终端窗口或调试控制台具体流程图如下:



1. **请简述环境变量CLASSPATH、PATH、JAVA\_HOME 的作用 JAVA\_HOME ：**JAVA\_HOME 是一个环境变量，用于指定Java开发工具包(JDK)的安装位置。这有助于系统知道在哪里可以找到JDK的bin目录和其他重要文件。许多Java相关的工具和程序都会依赖这个变量来确定JDK的位置。

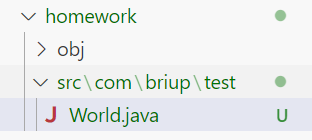
**PATH ：**PATH 是一个环境变量，它告诉操作系统在哪里可以找到可执行文件。当你在命令行中输入一个命令时，系统会在PATH变量列出的所有目录中查找该命令对应的可执行文件。对于Java开发者来说，在PATH中包含%JAVA\_HOME%/bin目录是非常重要的，这样就可以直接运行像java, javac等命令而不需要指定完整的路径。%JAVA\_HOME%/jre/bin这一路径的作用主要是为了让系统能够在命令行中直接运行Java运行时环境(JRE)中的可执行文件，如java.exe（在Windows系统上）或其他平台上的相应可执行文件。

**CLASSPATH ：**CLASSPATH 是一个环境变量，用于指定Java虚拟机(JVM)在加载类文件时应该搜索的路径。当你运行Java应用程序时，JVM会根据CLASSPATH变量的设置来查找需要的类文件。如果没有正确设置CLASSPATH，可能会导致JVM找不到所需的类文件，从而引发ClassNotFoundException异常。

1. **Linux中编写World.java文件，输出Hello Everyone 要求：**

* **java文件放在 ~/src 目录下**
* **类的全包名为 com.briup.test**
* **生成的class文件放到 ~/obj 目录下**

1. **创建目录及文件：**

****

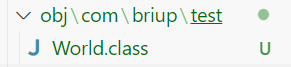
1. **文件内容：**
2. package com.briup.test;
3. class World {
4. public static void main(String[] args) {
5. System.out.println("Hello World!");
6. }
7. }
8. **创建目录并编译：**

****

**指令：**

javac -d ../../../../obj World.java

指令运行后文件夹变化



1. **运行：**

****

**指令：**

java -cp ../../../../obj com.briup.test.World

1. **简述Java跨平台的原理**

Java实现跨平台的主要原理是通过Java虚拟机（JVM）和字节码（Bytecode）技术。以下是具体步骤和原理：

1. 编写Java源代码：开发者使用Java语言编写程序。
2. 编译为字节码：Java源代码通过Java编译器（如javac）被编译成一种中间语言——字节码（.class文件），而不是直接编译成本地机器码。
3. 运行时环境：要运行Java程序，需要有一个Java运行时环境（JRE），其中包括Java虚拟机（JVM）。
4. JVM的作用：JVM负责读取字节码文件，并将其解释或即时编译（JIT）成本地机器码进行执行。·不同的平台上的JVM也各不相同，但它们都能理解相同的字节码。
5. 平台独立性：由于字节码是平台无关的，因此只要目标平台上安装了相应的JVM，就可以运行Java程序，实现了Sun Microsystems (Sun Microsystems是Java语言的原始创造者和早期的主要推动者)提出的口号“一次编写，到处运行”（WORA）的理念。
6. **请简述package和import的作用**

* **package：** 
  + **组织类**：通过将相关的类放在同一个包内，不同的包内放置不同功能和所操作对象的程序代码，可以帮助开发者更好地组织代码结构。
  + **访问控制**：包内的类默认可以互相访问（除非声明为private）。同时，可以通过public(公共)、protected(受保护的权限,子类可以访问)、default（即无修饰符, 默认权限 , 当不设定任何权限时 , JVM会默认提供权限 , 称之为包(路径)权限）和private(私有的 , 同一类中可以使用,可暂时理解为局部变量与全局变量的区别)等访问修饰符来控制类及其成员的可见性，和被访问的权限。
  + **避免命名冲突：**使用包名可以避免类名冲突，因为每个包内的类名必须是唯一的，但不同包之间可以有相同的名字**。**
* **import：**
  + **简化类名**：通过导入一个类，可以直接使用类名而不必加上完整的包名前缀。
  + **按需导入**：可以选择性地导入所需的类，而不是整个包，这样可以使代码更清晰，减少不必要的依赖，避免开发后文件过于臃肿，保证文件清洁。