## Java 程序是如何执行的

了解任何一门语言的精髓都是先俯览其全貌，从宏观的视角把握全局，然后再深入每个知识点逐个击破，这样就可以深入而快速的掌握一项技能。同样学习 Java 也是如此，本节就让我们先从整体来看一下 Java 中的精髓。

### Java 介绍

Java 诞生于 1991 年，Java 的前身叫做 Oak（橡树），但在注册商标的时候，发现这个名字已经被人注册了，后来团队的人就在咖啡馆讨论这件事该怎么办，有人灵机一动说叫 Java 如何，因为当时他们正在喝着一款叫做 Java 的咖啡。就这样，这个后来家喻户晓的名字，竟以这种“随意”的方式诞生了，并一直沿用至今。

Java 发展历程：

* 1990，Sun 成立了“Green Team”项目小组
* 1991，Java 语言前身 Oak（橡树）诞生
* 1995，Oak 语言更名为 Java
* 1996，Java 1.0 发布
* 1997，Java 1.1 发布
* 1998，Java 1.2 发布
* 2000，Java 1.3 发布
* 2000，Java 1.4 发布
* 2004，Java 5 发布
* 2006，Java 6 发布
* 2011，Java 7 发布
* 2014，Java 8 发布
* 2017，Java 9（非长期支持版）发布
* 2018.03，Java 10（非长期支持版） 发布
* 2018.09，Java 11（长期支持版）发布
* 2019.03, Java 12（非长期支持版） 发布

注：长期支持版指的是官方发布版本后的一段时间内，通常以“年”为计数单位，会对此版本进行持续维护和升级。

**版本发布时间**

Java 10 之后，官方表示每半年推出一个大版本，长期支持版本（LTS）每三年发布一次。

### Java 和 JDK 的关系

JDK（Java Development Kit）Java 开发工具包，它包括：编译器、Java 运行环境（JRE，Java Runtime Environment）、JVM（Java 虚拟机）监控和诊断工具等，而 Java 则表示一种开发语言。

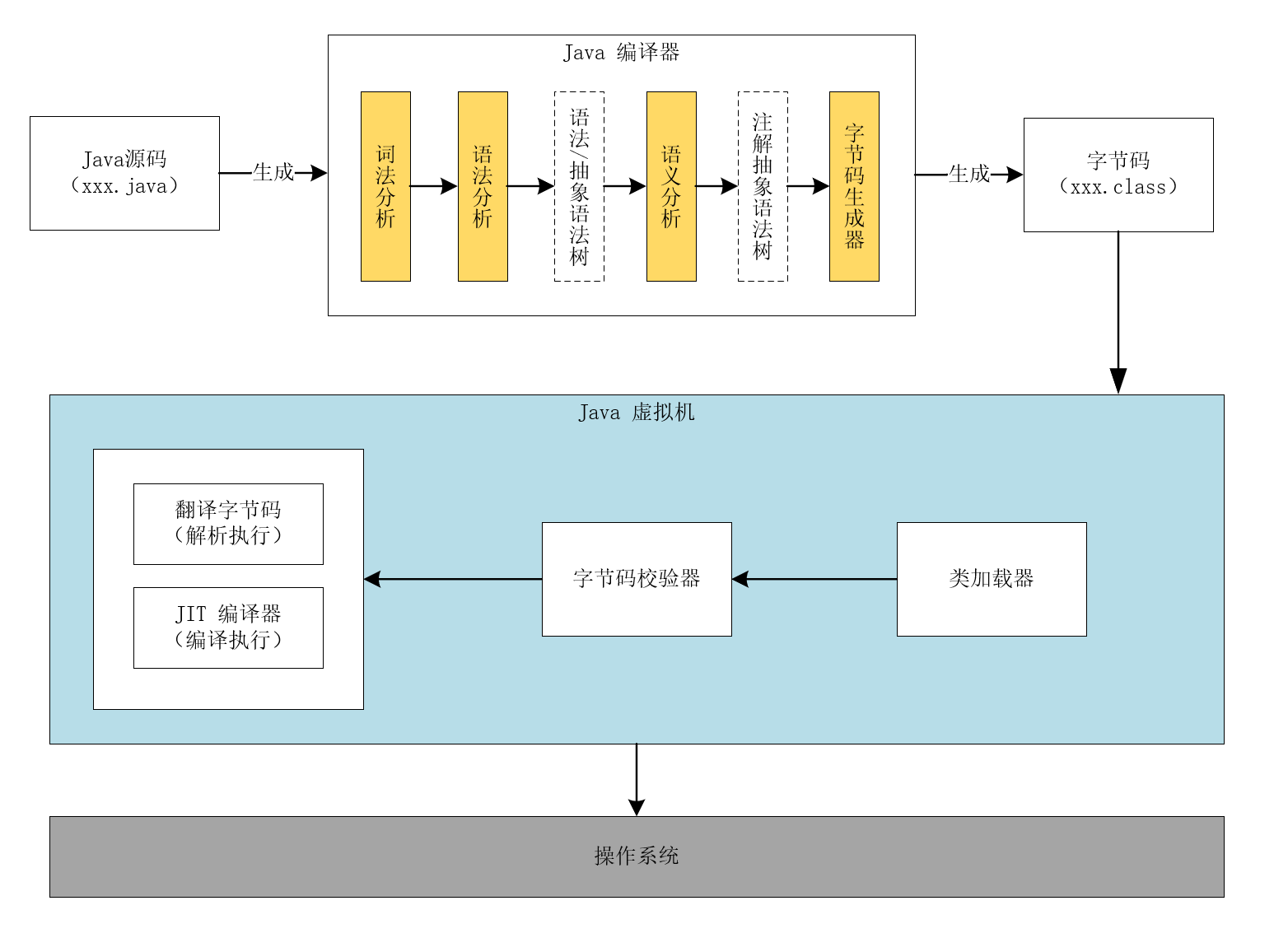
### Java 程序是怎么执行的？

我们日常的工作中都使用开发工具（IntelliJ IDEA 或 Eclipse 等）可以很方便的调试程序，或者是通过打包工具把项目打包成 jar 包或者 war 包，放入 Tomcat 等 Web 容器中就可以正常运行了，但你有没有想过 Java 程序内部是如何执行的？

其实不论是在开发工具中运行还是在 Tomcat 中运行，Java 程序的执行流程基本都是相同的，它的执行流程如下：

1. 先把 Java 代码编译成字节码，也就是把 .java 类型的文件编译成 .class 类型的文件。这个过程的大致执行流程：Java 源代码 -> 词法分析器 -> 语法分析器 -> 语义分析器 -> 字节码生成器 -> 最终生成字节码，其中任何一个节点执行失败就会造成编译失败；
2. 把 class 文件放置到 Java 虚拟机，这个虚拟机通常指的是 Oracle 官方自带的 Hotspot JVM；
3. Java 虚拟机使用类加载器（Class Loader）装载 class 文件；
4. 类加载完成之后，会进行字节码校验，字节码校验通过之后 JVM 解释器会把字节码翻译成机器码交由操作系统执行。但不是所有代码都是解释执行的，JVM 对此做了优化，比如，以 Hotspot 虚拟机来说，它本身提供了 JIT（Just In Time）也就是我们通常所说的动态编译器，它能够在运行时将热点代码编译为机器码，这个时候字节码就变成了编译执行。

Java 程序执行流程图如下：



avatar

### Java 虚拟机是如何判定热点代码的？

Java 虚拟机判定热点代码的方式有两种：

* 基于采样的热点判定

主要是虚拟机会周期性的检查各个线程的栈顶，若某个或某些方法经常出现在栈顶，那这个方法就是“热点方法”。这种判定方式的优点是实现简单；缺点是很难精确一个方法的热度，容易受到线程阻塞或外界因素的影响。

* 基于计数器的热点判定

主要就是虚拟机给每一个方法甚至代码块建立了一个计数器，统计方法的执行次数，超过一定的阀值则标记为此方法为热点方法。

Hotspot 虚拟机使用的基于计数器的热点探测方法。它使用了两类计数器：方法调用计数器和回边计数器，当到达一定的阀值是就会触发 JIT 编译。

方法调用计数器：在 client 模式下的阀值是 1500 次，Server 是 10000 次，可以通过虚拟机参数： -XX:CompileThreshold=N 对其进行设置。但是JVM还存在热度衰减，时间段内调用方法的次数较少，计数器就减小。

回边计数器：主要统计的是方法中循环体代码执行的次数。

由上面的知识我们可以看出， **要想做到对 Java 了如指掌，必须要好好学习 Java 虚拟机** ，那除了 Java 虚拟机外，还有哪些知识是面试必考，也是 Java 工程师必须掌握的知识呢？

#### 1. Java 基础中的核心内容

字符串和字符串常量池的深入理解、Array 的操作和排序算法、深克隆和浅克隆、各种 IO 操作、反射和动态代理（JDK 自身动态代理和 CGLIB）等。

#### 2. 集合

集合和 String 是编程中最常用的数据类型，关于集合的知识也是面试备考的内容，它包含：链表（LinkedList）、TreeSet、栈（Stack）、队列（双端、阻塞、非阻塞队列、延迟队列）、HashMap、TreeMap 等，它们的使用和底层存储数据结构都是热门的面试内容。

#### 3. 多线程

多线程使用和线程安全的知识也是必考的面试题目，它包括：死锁、6 种线程池的使用与差异、ThreadLocal、synchronized、Lock、JUC（java.util.concurrent包）、CAS（Compare and Swap）、ABA 问题等。

#### 4. 热门框架

Spring、Spring MVC、MyBatis、SpringBoot

#### 5. 分布式编程

消息队列（RabbitMQ、Kafka）、Dubbo、Zookeeper、SpringCloud 等。

#### 6. 数据库

MySQL 常用引擎的掌握、MySQL 前缀索引、回表查询、数据存储结构、最左匹配原则、MySQL 的问题分析和排除方案、MySQL 读写分离的实现原理以及 MySQL 的常见优化方案等。 Redis 的使用场景、缓存雪崩和缓存穿透的解决方案、Redis 过期淘汰策略和主从复制的实现方案等。

#### 7. Java 虚拟机

虚拟机的组成、垃圾回收算法、各种垃圾回收器的区别、Java 虚拟机分析工具的掌握、垃圾回收器的常用调优参数等。

#### 8. 其他

常用算法的掌握、设计模式的理解、网络知识和常见 Linux 命令的掌握等。

值得庆幸的是以上所有内容都包含在本专栏中，接下来就让我们一起学习，一起构建 Java 的认知体系吧!

### 相关面试题

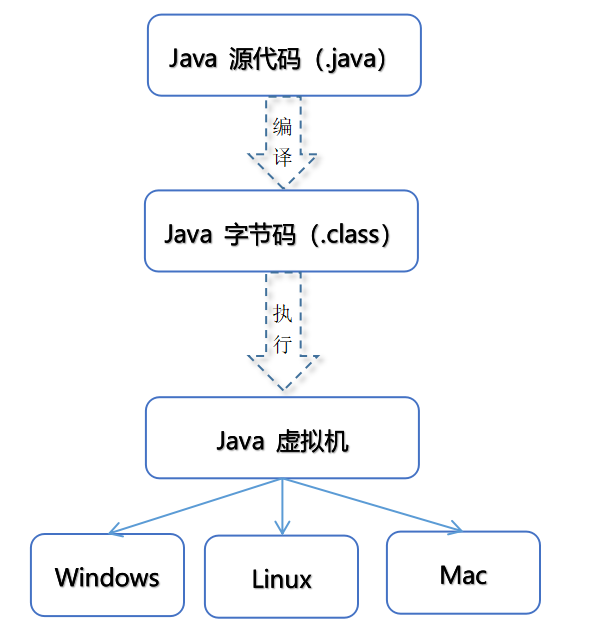
#### 1. Java 语言都有哪些特点？

答：Java 语言包含以下特点。

* 面向对象，程序容易理解、开发简单、方便；
* 跨平台，可运行在不同服务器类型上，比如：Linux、Windows、Mac 等；
* 执行性能好，运行效率高；
* 提供大量的 API 扩展，语言强大；
* 有多线程支持，增加了响应和实时交互的能力；
* 安全性好，自带验证机制，确保程序的可靠性和安全性。

#### 2. Java 跨平台实现的原理是什么？

答：要了解 Java 跨平台实现原理之前，必须先要了解 Java 的执行过程，Java 的执行过程如下：



执行过程

Java 执行流程：Java 源代码（.java）-> 编译 -> Java 字节码（.class） ->通过 JVM（Java 虚拟机）运行 Java 程序。每种类型的服务器都会运行一个 JVM，Java 程序只需要生成 JVM 可以执行的代码即可，JVM 底层屏蔽了不同服务器类型之间的差异，从而可以在不同类型的服务器上运行一套 Java 程序。

#### 3. JDK、JRE、JVM 有哪些区别？

答：了解了 JDK、JRE、JVM 的定义也就明白了它们之间的区别，如下所述。

* JDK：Java Development Kit（Java 开发工具包）的简称，提供了 Java 的开发环境和运行环境；
* JRE：Java Runtime Environment（Java 运行环境）的简称，为 Java 的运行提供了所需环境；
* JVM：Java Virtual Machine（Java虚拟机）的简称，是一种用于计算设备的规范，它是一个虚构出来的计算机，是通过在实际的计算机上仿真模拟各种计算机功能来实现的，简单来说就是所有的 Java 程序都是运行在 JVM（Java 虚拟机）上的。

总体来说，JDK 提供了一整套的 Java 运行和开发环境，通常使用对象为 Java 的开发者，当然 JDK 也包含了 JRE；而 JRE 为 Java 运行的最小运行单元，一般安装在 Java 服务器上，所以 JDK 和 JRE 可以从用途上进行理解和区分。JVM 不同于 JDK 和 JRE，JVM 是 Java 程序运行的载体，Java 程序只有通过 JVM 才能正常的运行。

#### 4. Java 中如何获取明天此刻的时间？

答：JDK 8 之前使用 Calendar.add() 方法获取，代码如下：

Calendar calendar = Calendar.getInstance();  
calendar.add(Calendar.DATE, 1);  
System.out.println(calendar.getTime());

JDK 8 有两种获取明天时间的方法。

方法一，使用 LocalDateTime.plusDays() 方法获取，代码如下：

LocalDateTime today = LocalDateTime.now();  
LocalDateTime tomorrow = today.plusDays(1);  
System.out.println(tomorrow);

方法二，使用 LocalDateTime.minusDays() 方法获取，代码如下：

LocalDateTime today = LocalDateTime.now();  
LocalDateTime tomorrow = today.minusDays(-1);  
System.out.println(tomorrow);

minusDays() 方法为当前时间减去 n 天，传负值就相当于当前时间加 n 天。

#### 5. Java 中如何跳出多重嵌套循环？

答：Java 中跳出多重嵌套循环的两种方式。

* 方法一：定义一个标号，使用 break 加标号的方式
* 方法二：使用全局变量终止循环

方法一，示例代码：

myfor:for (int i = 0; i < 100; i++) {  
 for (int j = 0; j < 100; j++) {  
 System.out.println("J:" + j);  
 if (j == 10) {  
 // 跳出多重循环  
 break myfor;  
 }  
 }  
}

方法二，示例代码：

boolean flag = true;  
for (int i = 0; i < 100 && flag; i++) {  
 for (int j = 0; j < 100; j++) {  
 System.out.println("J:" + j);  
 if (j == 10) {  
 // 跳出多重循环  
 flag = false;  
 break;  
 }  
 }  
}

#### 6. char 变量能不能存贮一个中文汉字？为什么？

答：char 变量可以存贮一个汉字，因为 Java 中使用的默认编码是 Unicode ，一个 char 类型占 2 个字节（16 bit），所以放一个中文是没问题的。

#### 7. Java 中会存在内存泄漏吗？请简单描述一下。

答：一个不再被程序使用的对象或变量一直被占据在内存中就造成了内存泄漏。

Java 中的内存泄漏的常见情景如下：

* 长生命周期对象持有短生命的引用，比如，缓存系统，我们加载了一个对象放在缓存中，然后一直不使用这个缓存，由于缓存的对象一直被缓存引用得不到释放，就造成了内存泄漏；
* 各种连接未调用关闭方法，比如，数据库 Connection 连接，未显性地关闭，就会造成内存泄漏；
* 内部类持有外部类，如果一个外部类的实例对象的方法返回了一个内部类的实例对象，这个内部类对象被长期引用了，即使那个外部类实例对象不再被使用，但由于内部类持有外部类的实例对象，这个外部类对象将不会被垃圾回收，这也会造成内存泄露；
* 改变哈希值，当一个对象被存储进 HashSet 集合中以后，就不能修改这个对象中的那些参与计算哈希值的字段了，否则对象修改后的哈希值与最初存储进 HashSet 集合中时的哈希值就不同了，在这种情况下，即使在 contains 方法使用该对象的当前引用作为的参数去 HashSet 集合中检索对象，也将返回找不到对象的结果，这也会导致无法从 HashSet 集合中单独删除当前对象，造成内存泄露。

更多java学习资料如下：

[Java高级架构师课程(总共122门课程，1460GB)](https://www.consultdog.com/index.php/front/44/)

[最强java面试视频课程(21门课程，126GB)](https://www.consultdog.com/index.php/front/78/)