一文让你“一夜封神“深入理解JVM一一简介和体系结构，java常用的框架和技术

原创

普通网友 于 2021-12-03 23:04:11 发布 150 收藏

分类专栏： 程序员 文章标签： 面试 java 后端

本文链接：https://blog.csdn.net/m0_64384302/article/details/121708890

程序员 专栏收录该内容

137 篇文章 1 订阅

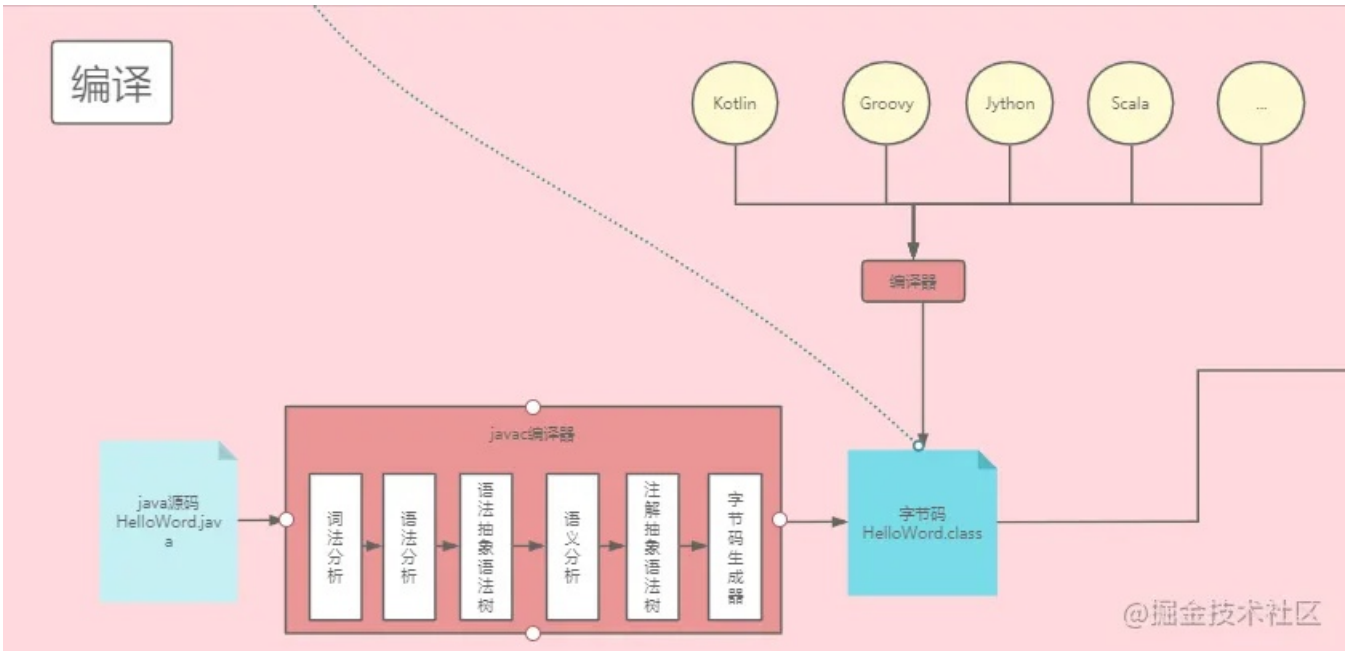
订阅专栏

注：本系列知识基于HotSpot JVM

HotSpot指的就是它的热点代码探测技术。

➢通过计数器找到最具编译价值代码，触发即时编译或栈上替换

➢通过编译器与解释疑协同工作，在最优化的程序响应时间与最佳执行性能中取得平衡



虚拟机和Java虚拟机区别

==========

虚拟机

===

就是一台虚拟的计算机。它是一款软件，用来执行一系列虚拟计算机指令。

虚拟机可以分为系统虚拟机和程序虚拟机。

> 大名鼎鼎的Visual Box，Mware就属于系统虚拟机，它们完全是对物理计算机的仿真，提供了一个可运行完整操作系统的软件平台。
>
> 程序虚拟机的典型代表就是Java虚拟机，它专门为执行单个计算机程序而设计，在Java虚拟机中执行的指令我们称为 Java 字节码指令。

无论是系统虚拟机还是程序虚拟机，在上面运行的软件都被限制于虚拟机提供的资源中。

Java虚拟机

=======

Java 虚拟机是一台执行Java字节码的虚拟计算机，它拥有独立的运行机制，其运行的Java字节码也未必由Java语言编译而成。

JVM平台的各种语言可以共享Java 虚拟机带来的跨平台性、优秀的垃圾回器，以及可靠的即时编译器。

Java技术的核心就是Java虚拟机（JVM，Java Virtual Machine），因为所有的Java程序都运行在Java虚拟机内部。

Java 虚拟机就是二进制字节码的运行环境，负责装载字节码到其内部，解释/编译为对应平台上的机器指令执行。每一

条Java指令，Java虚拟机规范中都有详细定义，如怎么取操作数，怎么处理操作数，处理结果放在哪里。

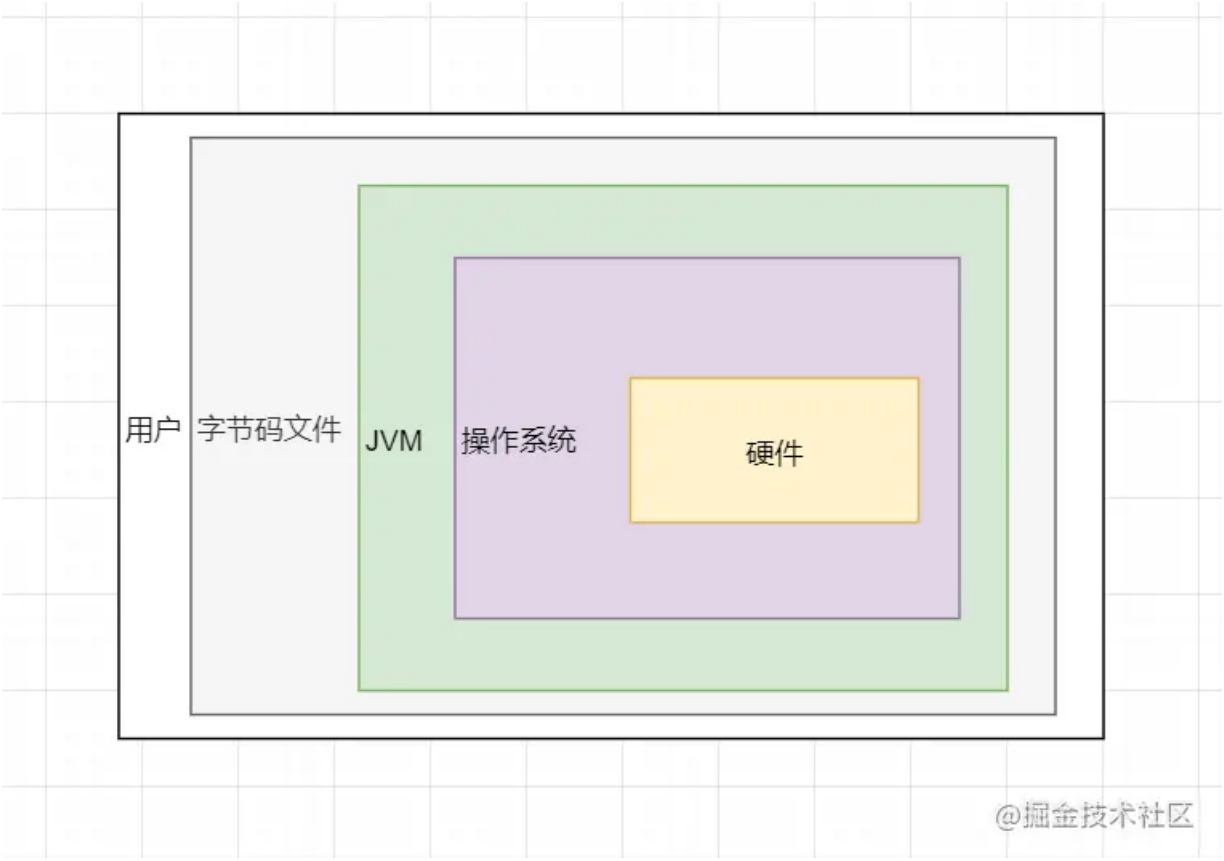
优点：

> 一次编译，到处运行
>
> 自动内存管理
>
> 自动垃圾回收功能

JVM位置

=====

计算机硬件上的第一层软件就是操作系统了，而jvm是基于操作系统之上的，它并没有与硬件有直接的交互。 其次，用户可通过Java编译器编译生成字节码文件，让字节码文件在jvm上执行。



Java体系结构

========

- **JDK**

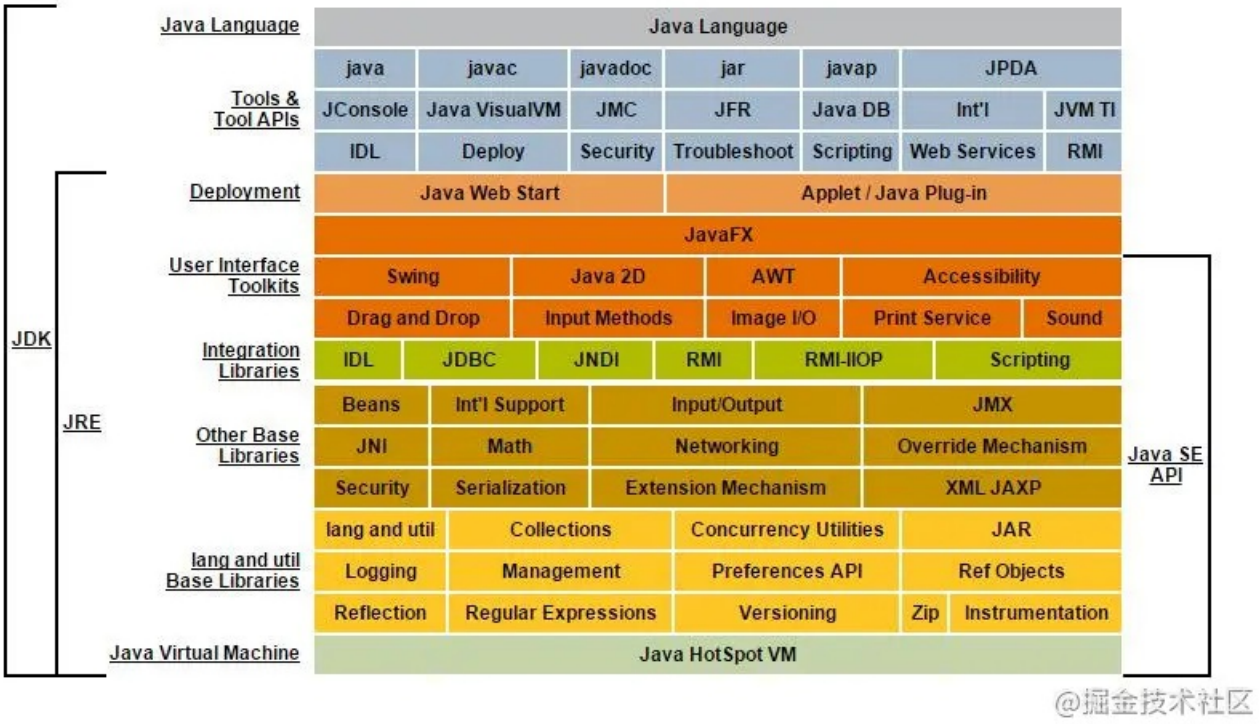
JDK(Java SE Development Kit)，Java标准开发包：它提供了编译、运行Java程序所需的各种工具和资源，包括Java编译器、Java运行时环境，以及常用的Java类库等。

- **JRE**

JRE(Java Runtime Environment) 、Java运行环境：用于解释执行Java的字节码文件。普通用户而只需要安装 JRE（Java Runtime Environment）来运行 Java 程序。而程序开发者必须安装JDK来编译、调试程序。

- **JVM**

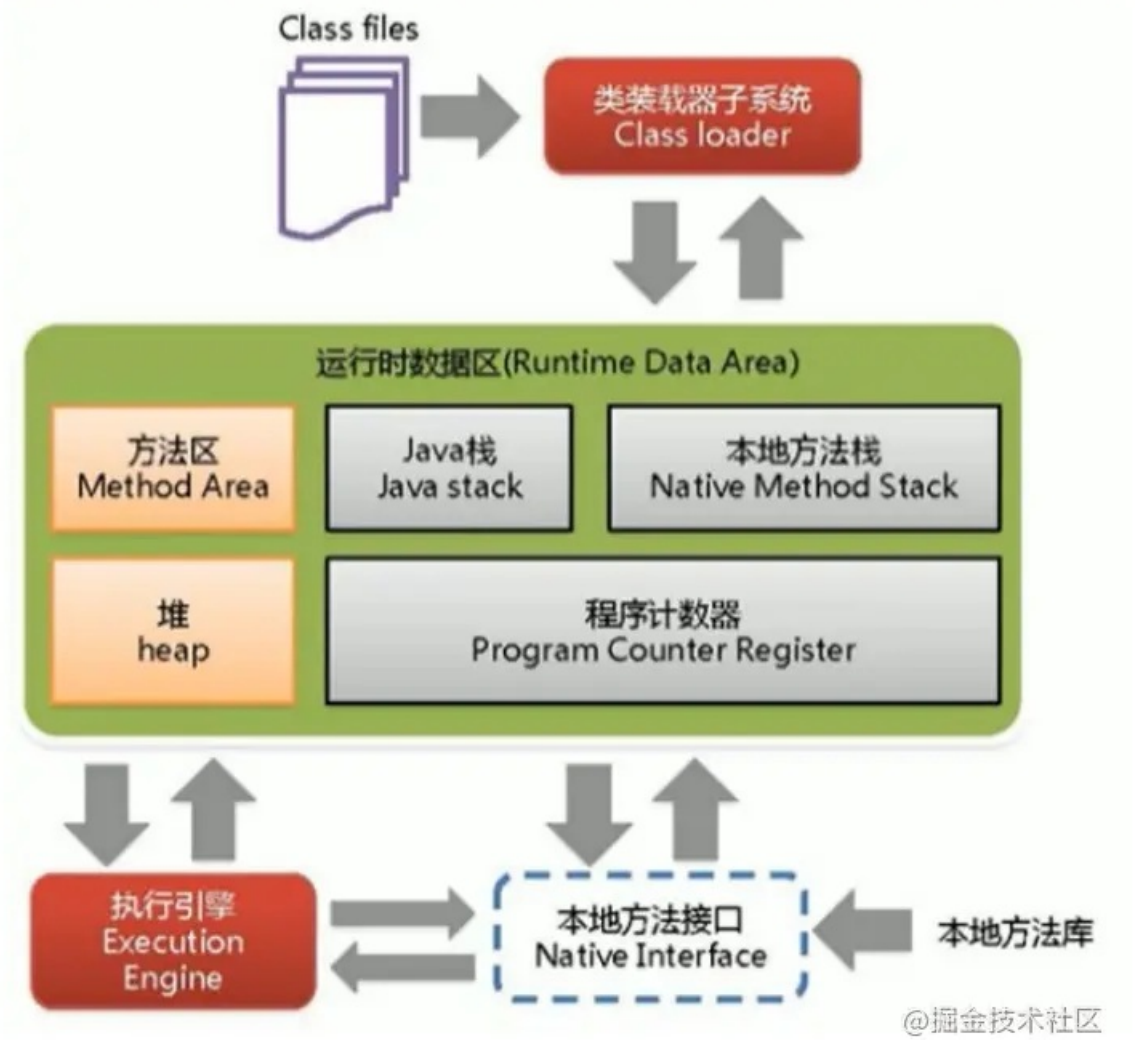
JVM(Java Virtual Mechinal)，Java虚拟机：是JRE的一部分。它是整个java实现跨平台的最核心的部分，负责解释执行字节码文件，是可运行java字节码文件的虚拟计算机。所有平台的上的JVM向编译器提供相同的接口，而编译器只需要面向虚拟机，生成虚拟机能识别的代码，然后由虚拟机来解释执行。



JVM 整体结构

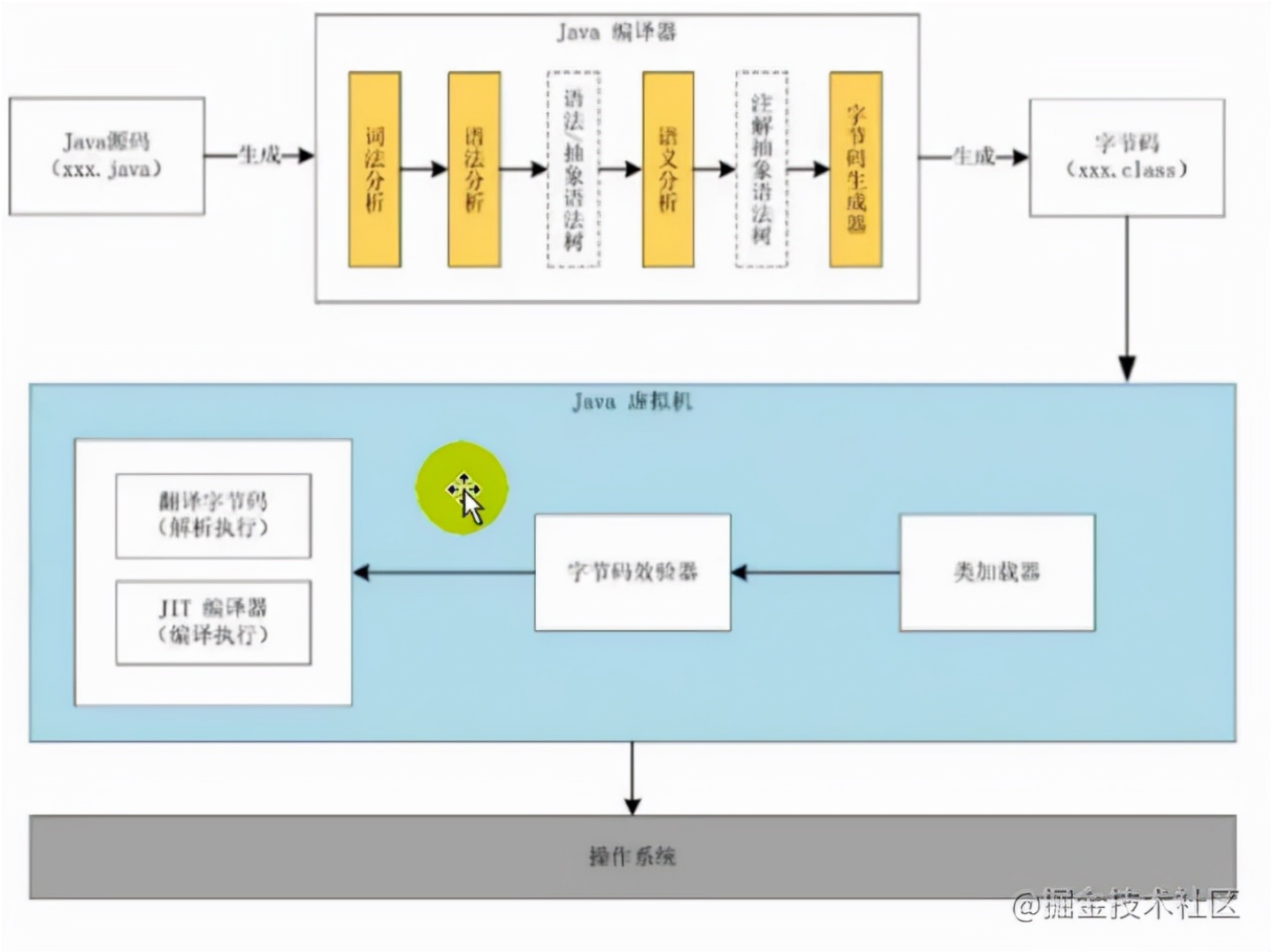
======

==


Class files
类装载器子系统
Class loader
运行时数据区(Runtime Data Area)
方法区
Method Area
Java栈
Java stack
本地方法栈
Native Method Stack
堆
heap
程序计数器
Program Counter Register
执行引擎
Execution
Engine
本地方法接口
Native Interface
本地方法库
@掘金技术社区

Java代码执行流程

==========



JVM 的架构模型

=========

Java编译器输入的指令流基本上是一种基于栈的指令集架构，另外一种指令集架构则是基于寄存器的指令集架构。具体来说:这两种架构之间的区别:

1. 基于栈式架构的特点

   设计和实现更简单，适用于资源受限的系统

   避开了寄存器的分配难题:使用零地址指令方式分配。

   指令流中的指令大部分是零地址指令，其执行过程依赖于操作栈。指令集更小，编译器容易实现。

   不需要硬件支持，可移植性更好，更好实现跨平台基于寄存器架构的特点

1. 基于寄存器架构的特点

- 典型的应用是x86的二进制指令集:比如传统的PC以及Android 的 Davlik虚拟机。