JVM体系结构概览

网友笔记，优秀：
https://blog.csdn.net/q961250375/article/details/107499173

1. JVM简介

JVM是Java Virtual Machine（Java虚拟机）的缩写。JVM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。

注意：

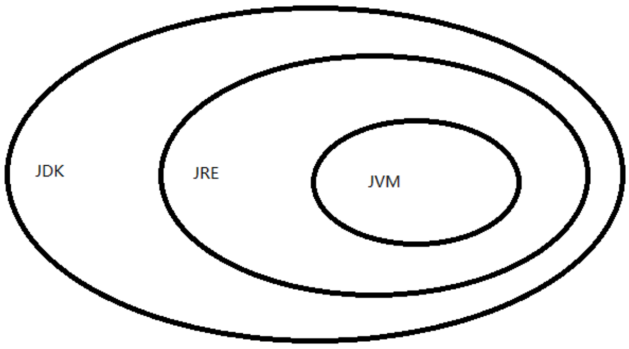
JRE、JVM、JDK三者的关系是什么？

JDK：Java开发工具包，目的就是用来编译和调试Java程序。

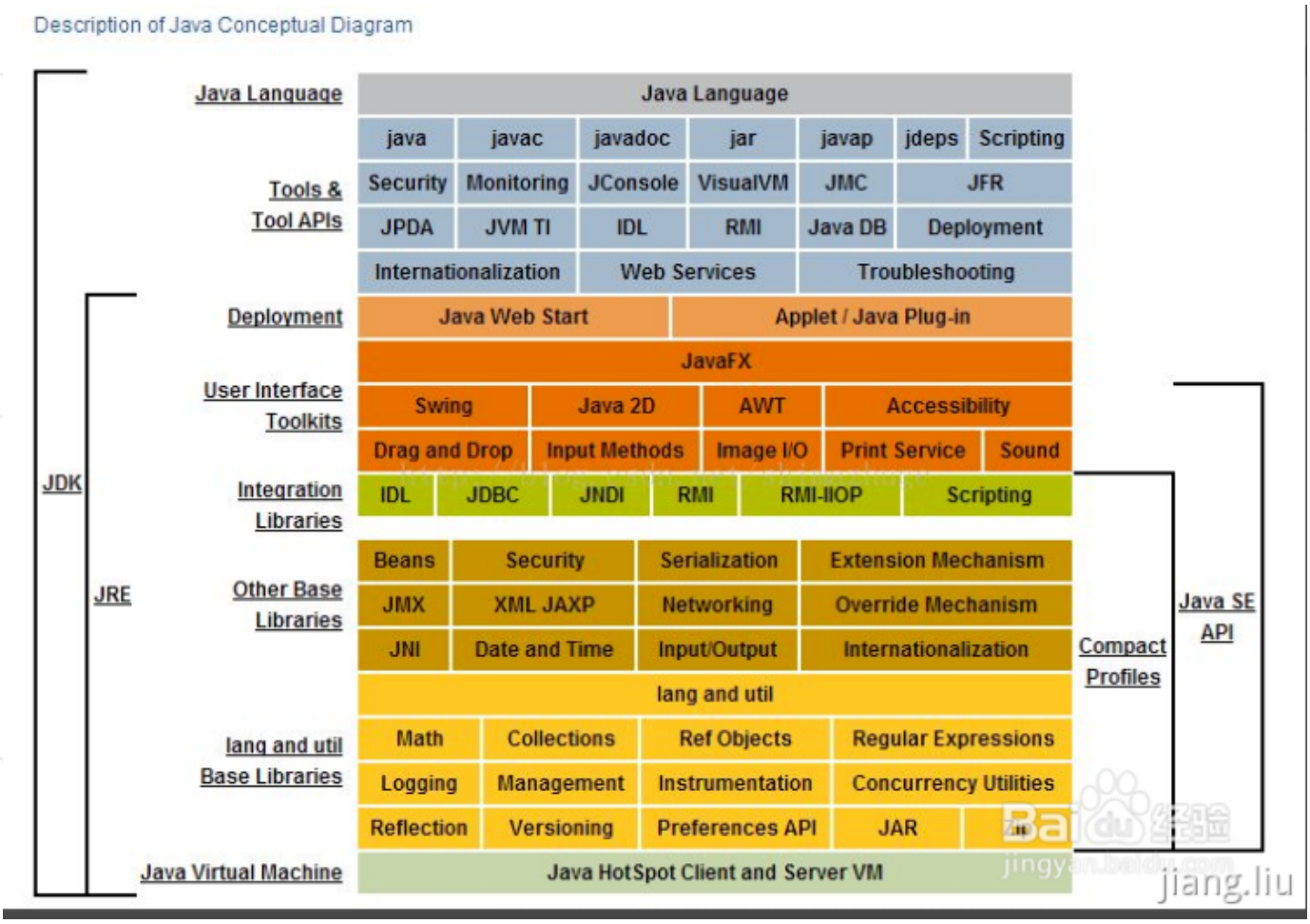
JRE：Java运行环境，也就是我们的写好的程序必须在JRE才能够运行

JVM：Java虚拟机，是指负责将字节码解释成为特定的机器码进行运行，值得注意的是在运行过程中，Java源程序需要通过编译器编译为.class文件，否则JVM不认识。

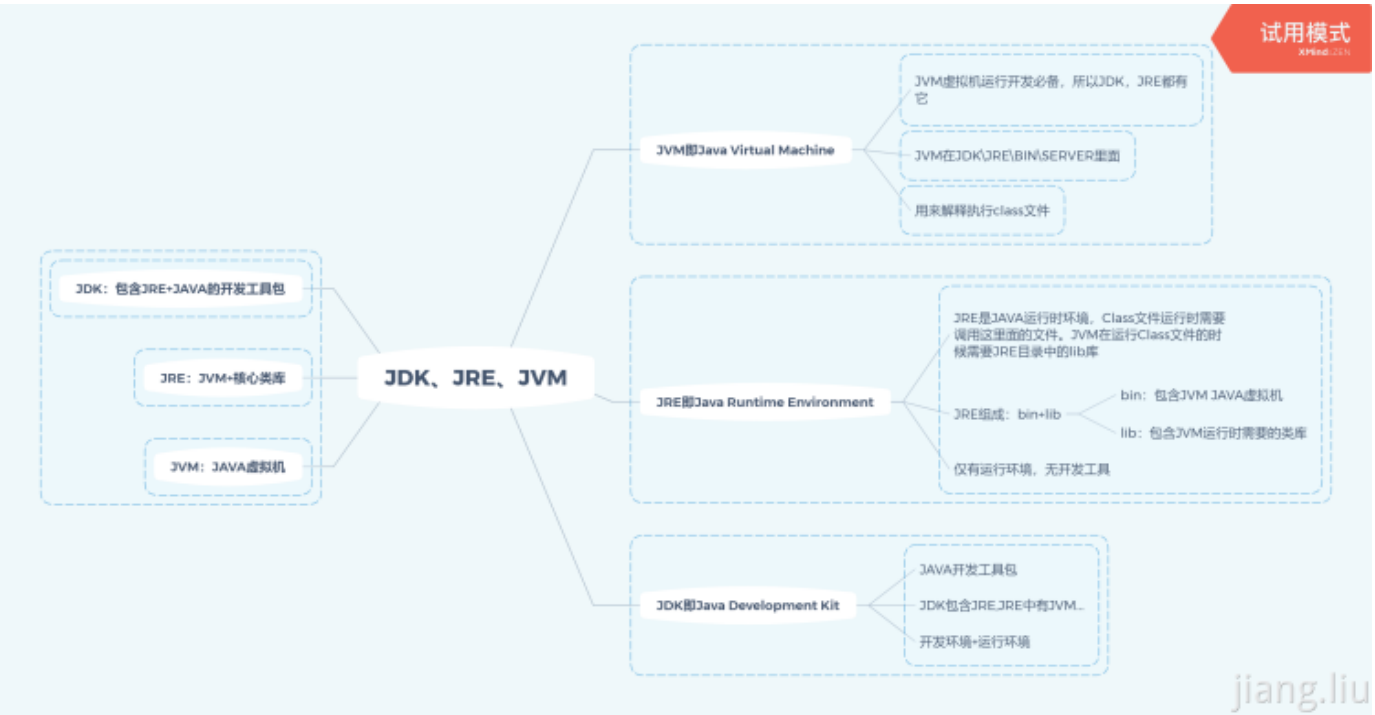
JDK包含了JRE和JVM，JRE包含了JVM，其中JRE中没有javac



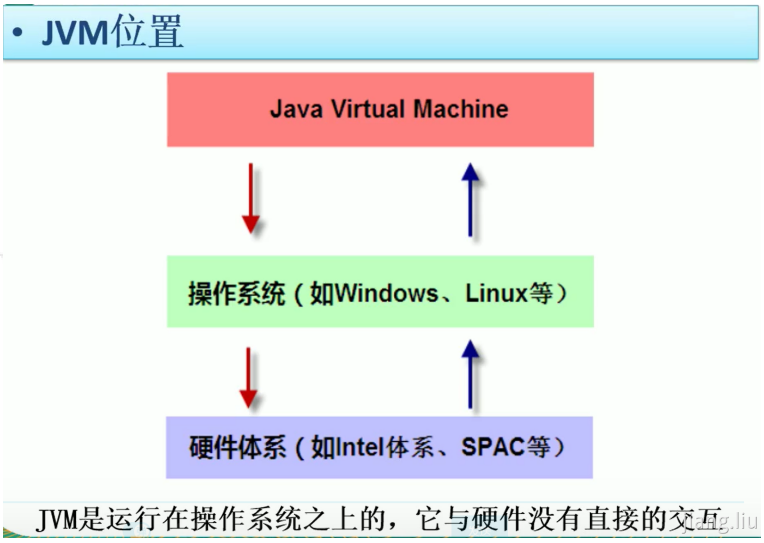
官方文档给出的图：



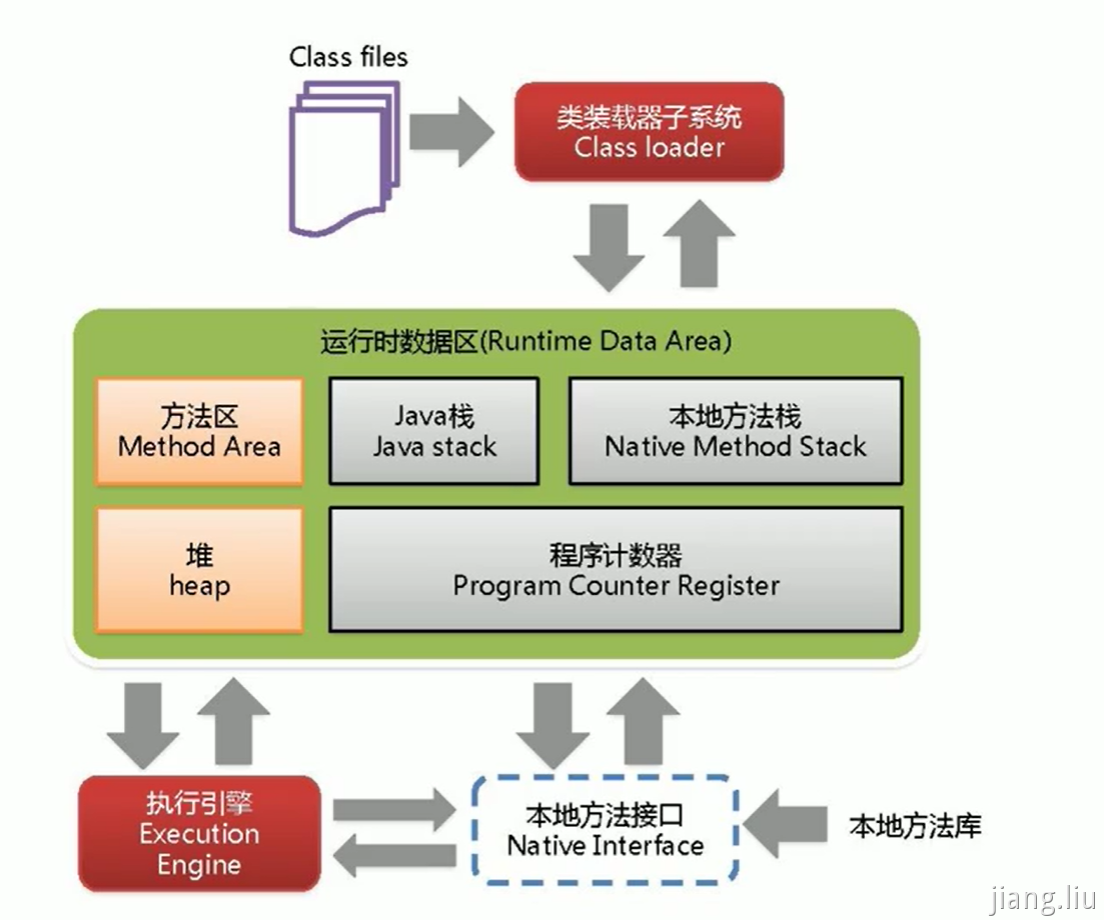
借网友图：



2. JVM位置

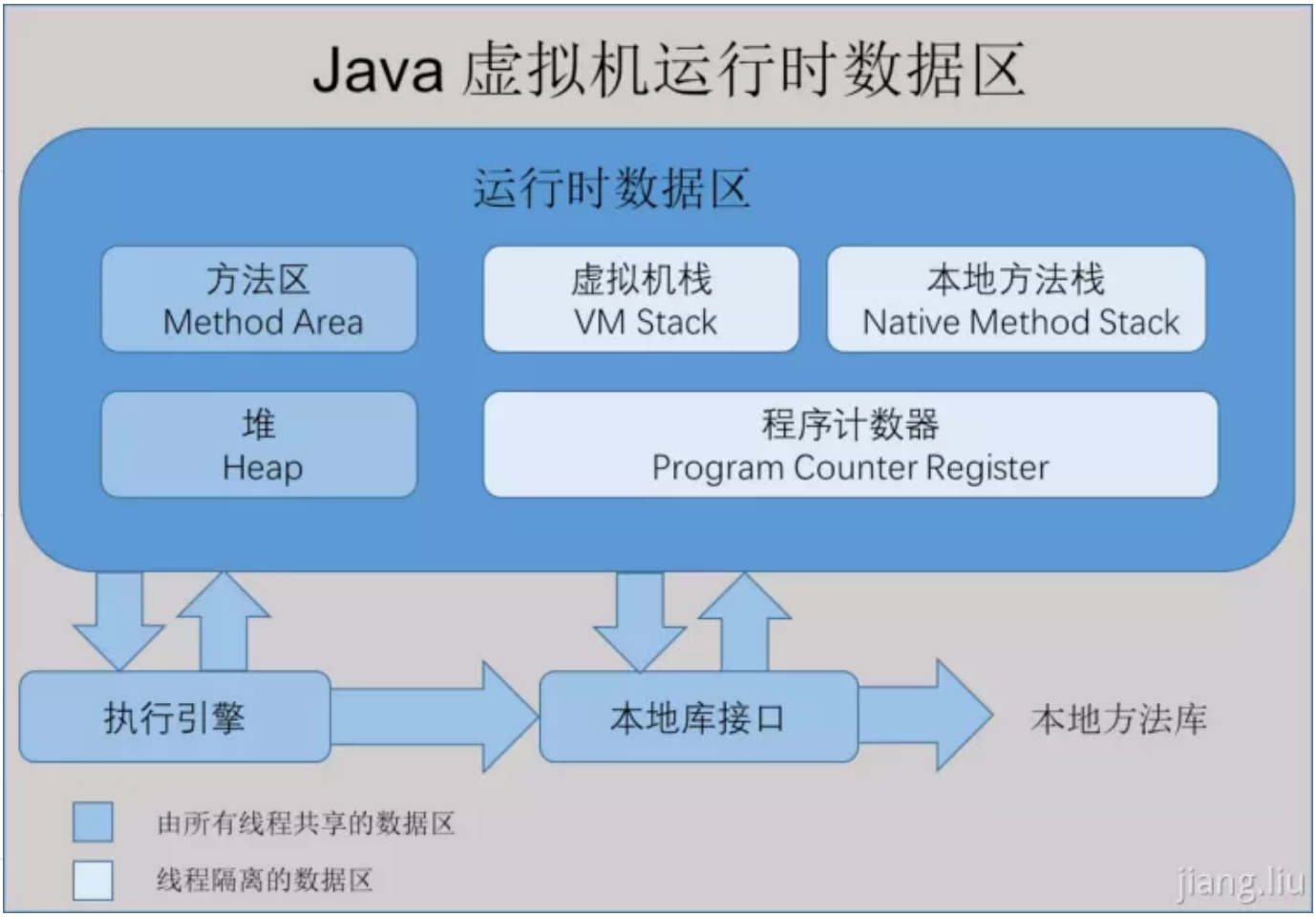


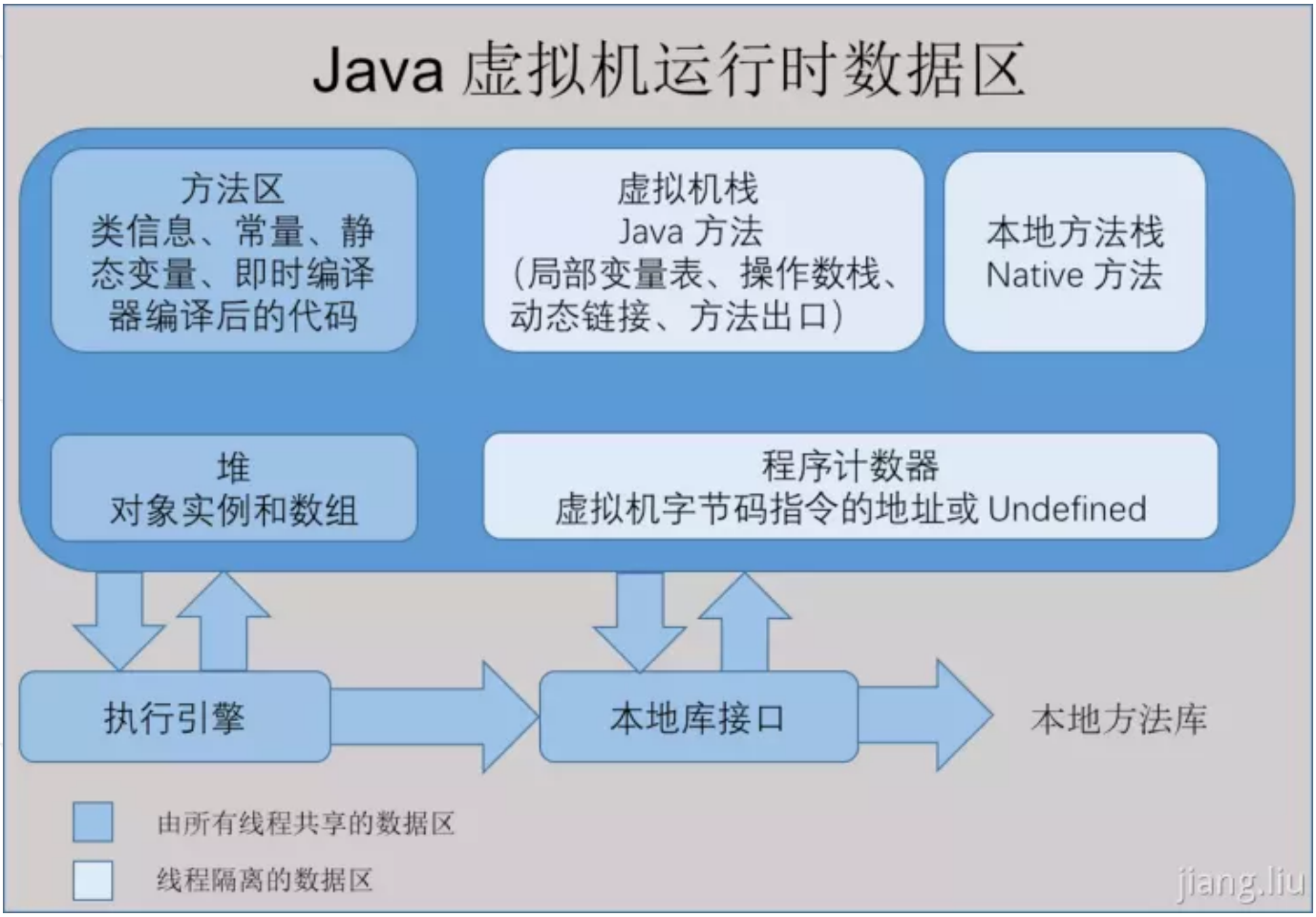
3.JVM运行时数据区域



灰色：代表线程私有 内存占用较少
橙色：线程共享 存在垃圾回收

下图出自《Java 虚拟机规范(Java SE 7 版)》


Java 虚拟机运行时数据区
运行时数据区
方法区
Method Area
虚拟机栈
VM Stack
本地方法栈
Native Method Stack
堆
Heap
程序计数器
Program Counter Register
执行引擎
本地库接口
本地方法库
由所有线程共享的数据区
线程隔离的数据区
jiang.liu

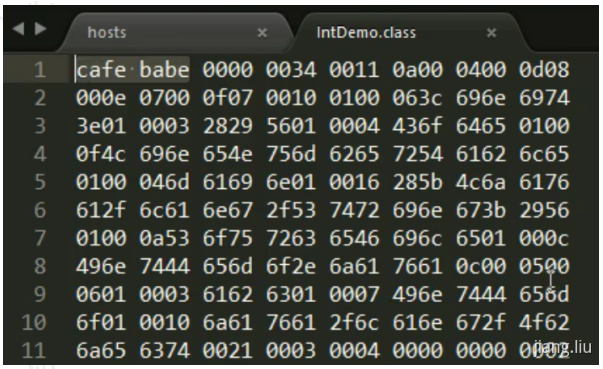


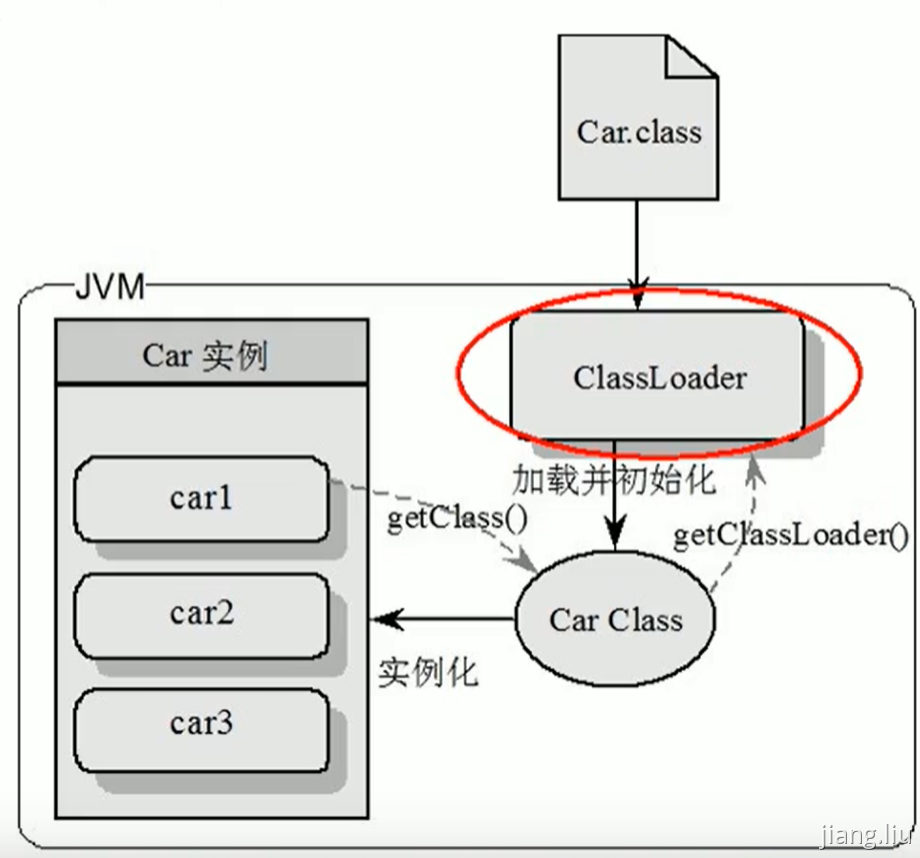
4. 类加载器 ClassLoader

类加载器介绍

负责加载class文件，class文件在**文件开头有特定的文件标示**，将class文件字节码内容加载到内存中，并将这些内容转换成方法区中的运行时数据结构，ClassLoader只负责class文件的加载，至于它是否可以运行，则由Execution Engine（执行引擎）决定。

class文件示例：开头有cafe babe标识





题外话：C语言是面向过程的语言，Linux是用C编写的，Redis也是用C编的，由于它是面向过程，太底层了，所有往上升了一层，多了一门语言：C++，C++就是C语言加面向对象的思想。C++很强大但是太复杂，所以在C++的基础之上再次简化，所以java诞生了，java对C++做了一些复杂功能的屏蔽和删减，比如C++支持多继承，java只支持单继承。

类加载器的种类

虚拟机自带的类加载器（3种）：

- 启动类加载器（Bootstrap），又称为“根加载器”；由C++语言编写。负责加载rt.jar里的所有class，不是ClassLoader的子类。
- 扩展类加载器（Extension），java编写。负责加载JRE的扩展目录，lib/ext或者由java.ext.dirs系统属性指定的目录中的JAR包的类。由Java语言实现，父类加载器为null（也即bootstrap根类加载器，因为它是c 写的就用null表示，但是null不是代表空）。
- 应用程序类加载器（AppClassLoader），也叫系统类加载器，加载当前应用的classpath的所有类。他的父类加载器为扩展类加载器。

启动类加载器加载rt.jar等第一代jdk自带的类(一出娘胎就自带)，扩展类加载器加载的jdk这20多年迭代的扩展类，诸如javax.*, 应用程序类加载器加载程序中的class文件，包括用户自己导入的外部jar。程序可以通过ClassLoader的静态方法getSystemClassLoader()来获取系统类加载器。如果没有特别指定，

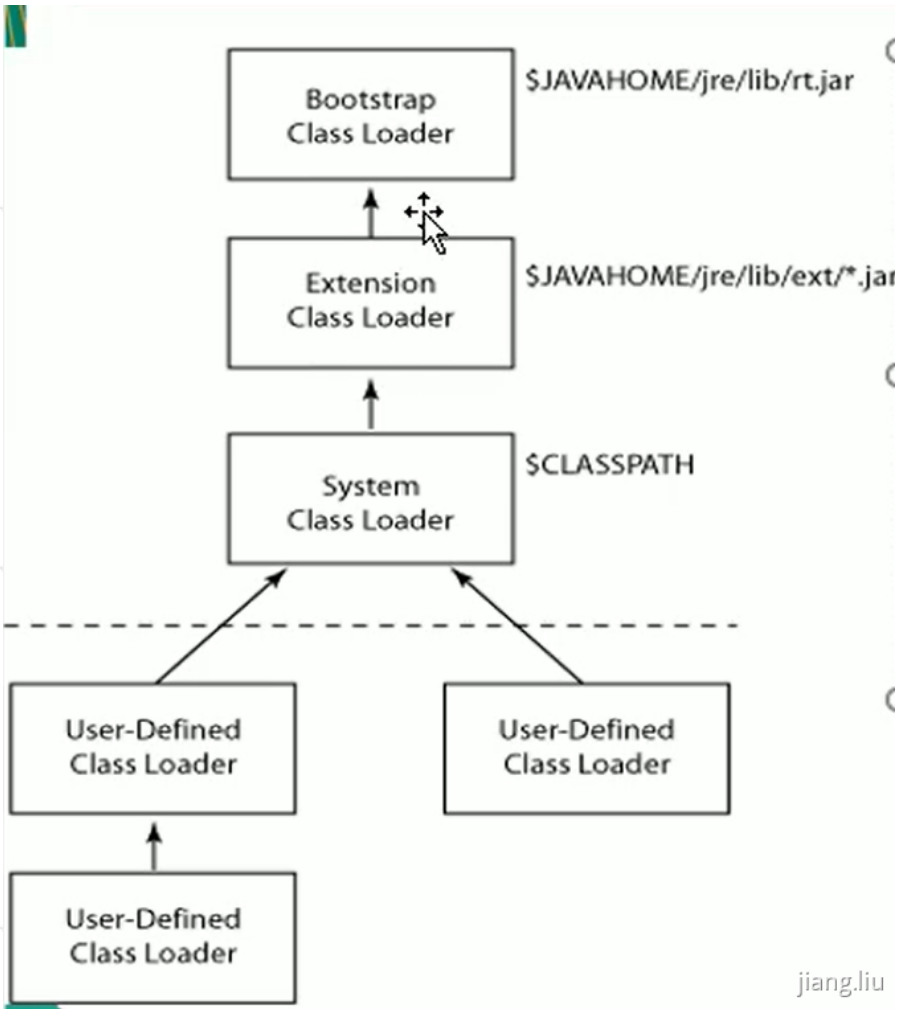
则用户自定义的类加载器都以此类加载器作为父加载器。由Java语言实现，父类加载器为ExtClassLoader。

用户自定义类加载器：

Java.lang.ClassLoader的子类，用户可以定制类的加载方式。 开发者可以通过继承ClassLoader基类来创建自己的类加载器。

Java提供了抽象类 java.lang.ClassLoader，所有用户自定义的类加载器都应该继承ClassLoader类。如果没有指定它的父加载器(ClassLoader构造方法无参数)，那么系统类加载器就将成为该自定义类加载器的父加载器。

类加载器的层级结构



```java
public class Demo1 {

    public static void main(String[] args) {

        Object object = new Object();
        System.out.println(object.getClass().getClassLoader());

        Hello hello = new Hello();
        System.out.println(hello.getClass().getClassLoader());
        System.out.println(hello.getClass().getClassLoader().getParent());
    }
}

输出:
null
sun.misc.Launcher$AppClassLoader@18b4aac2
sun.misc.Launcher$ExtClassLoader@45ee12a7

说明: sun.misc.Launcher是java虚拟机的入口引用类。
```

类加载器加载Class步骤

类加载器加载Class大致要经过如下8个步骤:

1. 检测此Class是否载入过，即在缓冲区中是否有此Class，如果有直接进入第8步，否则进入第2步。
2. 如果没有父类加载器，则要么Parent是根类加载器，要么本身就是根类加载器，则跳到第4步，如果父类加载器存在，则进入第3步。
3. 请求使用父类加载器去载入目标类，如果载入成功则跳至第8步，否则接着执行第5步。
4. 请求使用根类加载器去载入目标类，如果载入成功则跳至第8步，否则跳至第7步。
5. 当前类加载器尝试寻找Class文件，如果找到则执行第6步，如果找不到则执行第7步。
6. 从文件中载入Class，成功后跳至第8步。
7. 抛出ClassNotFountException异常。
8. 返回对应的java.lang.Class对象。

资料:

https://blog.csdn.net/qq_41701956/article/details/81664921