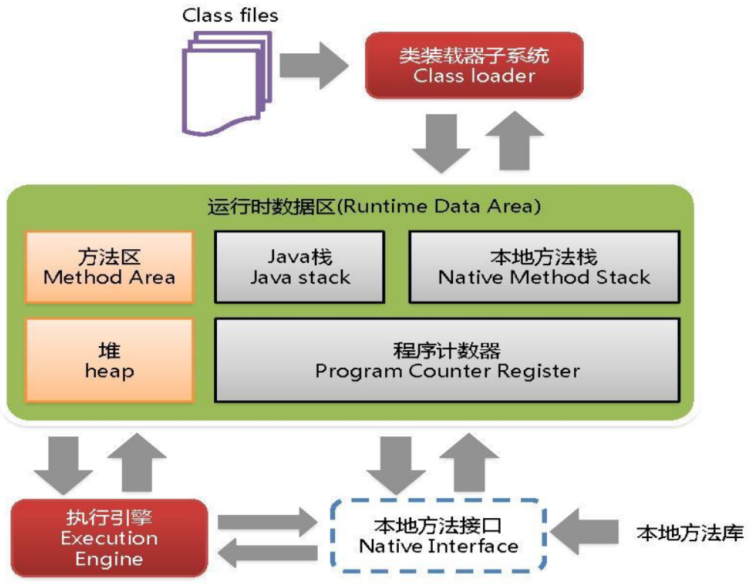
本地方法接口

在讲Java虚拟机运行时数据区中本地方法栈之前，我们先来说说运行时数据区之外的一个叫本地方法接口的东西简称JNI（Java Native Interface）

JVM体系结构概览



简单来讲，一个Native Method就是一个java调用非java代码的接口，一个Native Method 是这样一个java方法：该方法的底层实现由非Java语言实现，比如C。这个特征并非java特有，很多其他的编程语言都有这一机制，比如在C++ 中，你可以用extern “C” 告知C++ 编译器去调用一个C的函数。

在定义一个native method时，并不提供实现体（有些像定义一个Java interface），因为其实现体是由非java语言在外面实现的。

本地接口的作用是融合不同的编程语言为java所用，它的初衷是融合C/C++程序。

标识符native可以与其他所有的java标识符连用，但是abstract除外。

```java
/**
 * 本地方法
 */
public  class IHaveNatives {

    //abstract 没有方法体
    public abstract void abstractMethod(int x);

    //native 和 abstract不能共存，native是有方法体的，由C语言来实现
    public native void Native1(int x);

    native static public long Native2();

    native synchronized private float Native3(Object o);

    native void Native4(int[] array) throws Exception;

}
```

为什么要使用Native Method

java使用起来非常方便，然而有些层次的任务用java实现起来不容易，或者我们对程序的效率很在意时，问题就来了。

- 与java环境外交互：
  有时java应用需要与java外面的环境交互，这是本地方法存在的主要原因。 你可以想想java需要与一些底层系统，如擦偶偶系统或某些硬件交换信息时的情况。本地方法正式这样的一种交流机制：它为我们提供了一个非常简洁的接口，而且我们无需去了解java应用之外的繁琐细节。
- 与操作系统交互（比如线程最后要回归于操作系统线程）
  JVM支持着java语言本身和运行库，它是java程序赖以生存的平台，它由一个解释器（解释字节码）和一些连接到本地代码的库组成。然而不管怎样，它毕竟不是一个完整的系统，它经常依赖于一些底层系统的支持。这些底层系统常常是强大的操作系统。通过使用本地方法，我们得以用java实现了jre的与底层系统的交互，甚至jvm的一些部分就是用C写的。还有，如果我们要使用一些java语言本身没有提供封装的操作系统特性

时，我们也需要使用本地方法。

- Sun’s Java
  Sun的解释器是用C实现的，这使得它能像一些普通的C一样与外部交互。jre大部分是用java实现的，它也通过一些本地方法与外界交互。例如：类java.lang.Thread的setPriority()方法是用Java实现的，但是它实现调用的事该类里的本地方法setPriority0（）。这个本地方法是用C实现的，并被植入JVM内部，在Windows 95的平台上，这个本地方法最终将调用Win32 setPriority()API。这是一个本地方法的具体实现由JVM直接提供，更多的情况是本地方法由外部的动态链接库（external dynamic link library）提供，然后被JVM调用。

现 状

目前该方法的是用越来越少了，除非是与硬件有关的应用，比如通过java程序驱动打印机或者java系统管理生产设备，在企业级应用已经比较少见。因为现在的异构领域间的通信很发达，比如可以使用Socket通信，也可以是用Web Service等等，不多做介绍。