1 Java技术体系

组成部分：java程序设计语言、java virtual machine(JVM)、class文件的格式、java api 类库、第三方类库

其中java语言、JVM、JAVA API类库称为JDK，JDK是支持java程序的最小开发环境。JAVA API类库中的JAVA SE API子集和JVM统称为JRE（java runtime environment)，JRE是支持java程序运行的标准环境

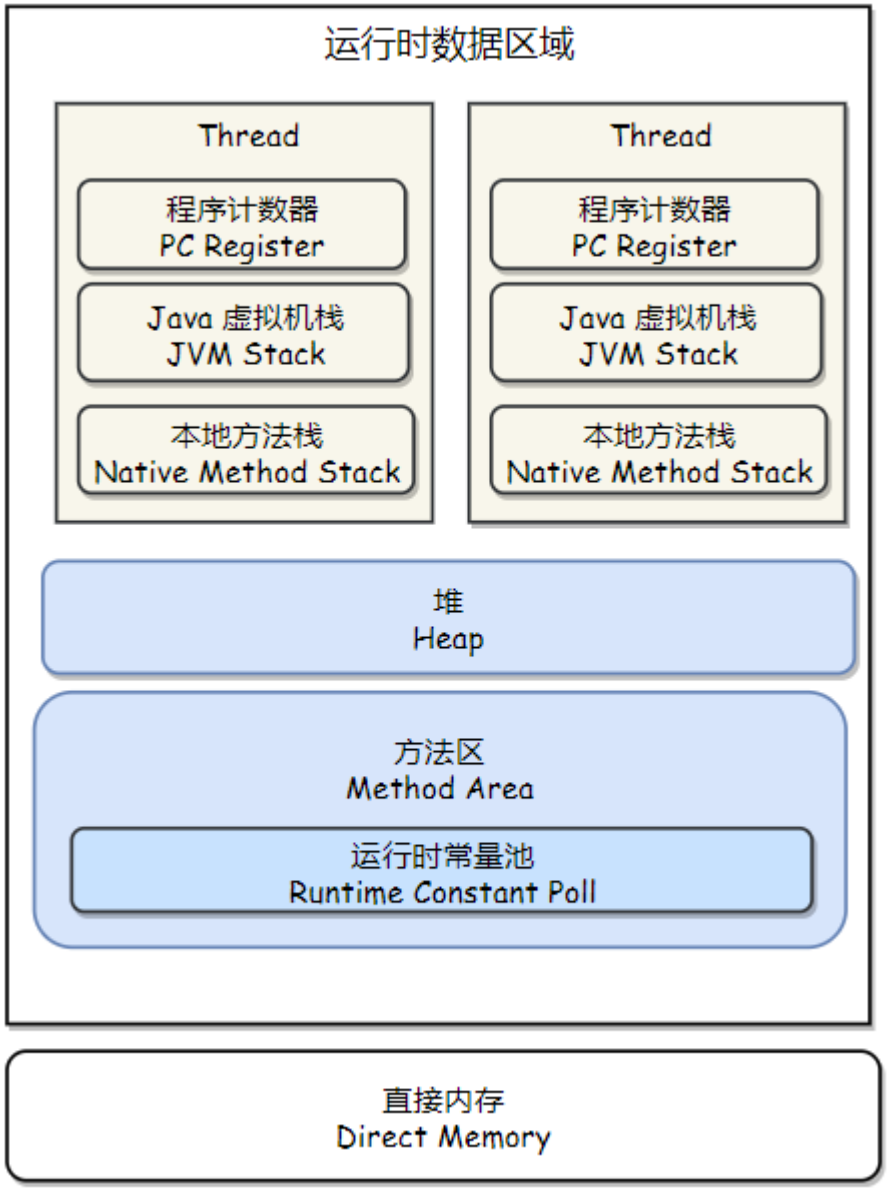
JAVA ME: 移动库
JAVA SE: 标准库
JAVA EE：企业库

JVM

jvm版本众多，使用范围较广的有SUN JDK(Oracle JDK)和 OpenJDK。也有其他的如，Apache Harmony VM、Microsoft JVM

2 java 自动内存管理机制

java 虚拟机运行时的数据分区如下：



线程私有：

- 虚拟机栈(jvm stack)：

jvm stack 的生命周期和线程相同，虚拟机栈描述的是java方法执行的内存模型：每个方法执行的同时都会创建一个栈帧,用于存储 **局部变量表，操作数栈，动态链接， 方法出口** 等信息，每一个方法从调用到完成的过程就对应着一个栈帧在虚拟机栈中入栈到出栈的过程。

通常所说的栈内存指的是虚拟机栈中的**局部变量表**部分，存放了编译期间可知的各种基本数据类型、对象引用。其中long和double类型的数据会占用2个局部变量空间，其他只占1个。

局部变量表的空间大小在编译期间就已经完成分配，运行期间不会改变局部变量表的大小。

在jvm规范中，对这个区域规定了两种异常情况：

1. 如果线程请求的栈深度大于虚拟机所允许的深度，则抛出StackOverflowError异常
2. 如果jvm设置了动态扩展虚拟机栈的话，则会抛出OutOfMemoryError异常

例如在函数递归调用深度过深的话可能导致栈溢出异常。

- 本地方法栈(native method statck):

和虚拟机栈非常相似，区别在于虚拟机栈为虚拟机执行Java方法，而本地方法栈执行的是Java Native 方法服务。

- 程序计数器(program counter register):

当前线程所执行的字节码的行号指示器，记录正在执行的虚拟机字节码指令的地址（如果正在执行的是本地方法则为空）

线程共享：

- 堆(Heap)

Java 堆是被所有线程共享的一块内存区域，唯一目的是存放对象实例，是jvm中内存最大的一块。也是java GC的主要区域。

java堆也称为"GC"堆，现在的垃圾收集器大多采用分代收集算法。

该算法的思想是针对不同的对象采用不同的垃圾回收算法，java堆分为3块：

- 新生代
- 老生代
- 永生代

当一个对象被创建时，它首先进入新生代，之后有可能被转移到老年代中。新生代存放着大量的生命很短的对象，因此新生代在三个区域中垃圾回收的频率最高。

- 方法区(Method Area):

  方法区和java堆一样，是各个线程共享的内存区域，用来存储被虚拟机加载的类信息、常量、静态变量、即时编译后的代码数据，方法区也叫"非堆区"。在Hot-spot循迹上面，方法区也可以成为"永久代"。

  - 运行时常量池 运行时常量池属于方法区的一部分，用于存放编译期生成（即Class文件中）的各种字面量和符号引用。

    除了在编译期生成的常量，还允许动态生成，例如 String 类的 intern()。这部分常量也会被放入运行时常量池。

直接内存(Direct Memory)：

在 JDK 1.4 中新加入了 NIO 类，引入了一种基于通道 (channel) 和缓冲区 (Buffer) 的 I/O 方式它可以使用 Native 函数库直接分配堆外内存，然后通过一个存储在 Java 堆里的 DirectByteBuffer 对象作为这块内存的引用进行操作。这样能在一些场景中显著提高性能，因为避免了在 Java 堆和 Native 堆中来回复制数据。

直接内存不受java堆大小的限制，但是受物理机的总内存和swap的限制，因此扩展的时候可能出现OutOfMemoryError异常

3 jvm 垃圾收集器

程序计数器、虚拟机栈和本地方法栈这三个区域属于线程私有的，只存在于线程的生命周期内，线程结束之后也会消失，因此不需要对这三个区域进行垃圾回收。垃圾回收主要是针对 Java 堆和方法区进行。

判断对象是否可回收

1. 引用计数法

给对象添加一个引用计数器，没当一个地方引用它的时候，计数器加1，引用失效时，计数器减1，计数器为0的时候表示对象不可能被使用了。

但是该方法不能解决相互循环引用的问题。