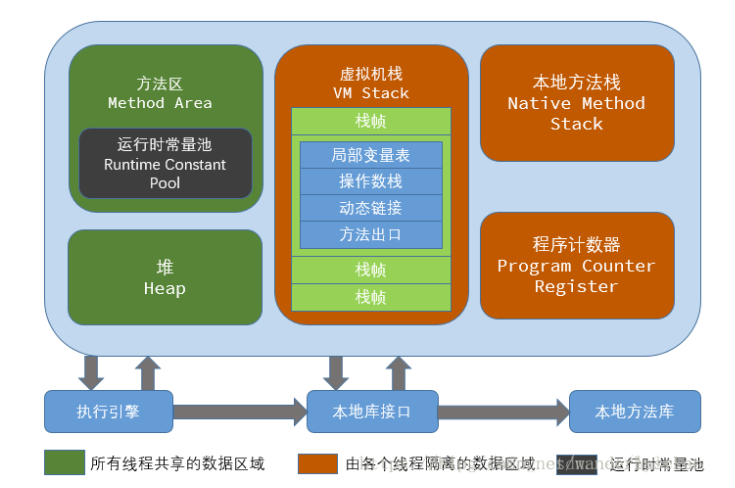
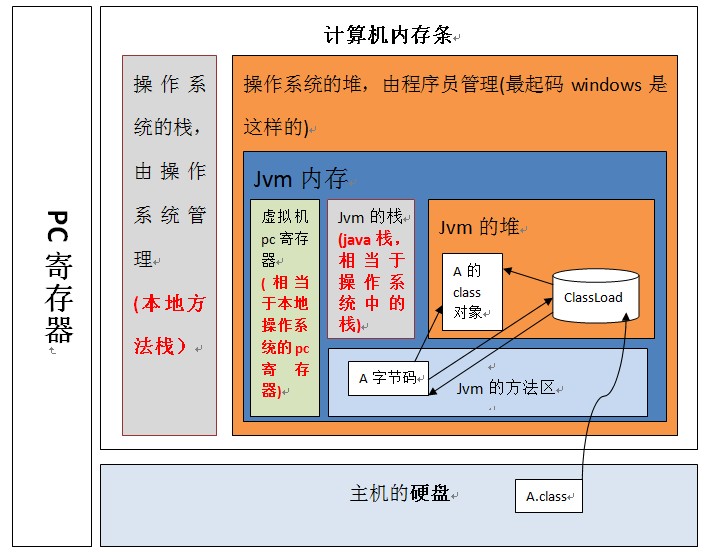
**Jvm内存模型：**



**Jvm的栈就相当于操作系统中的栈，都是自己管理的，计算机的堆是程序员管理的，jvm的堆是gc回收的，还有方法区，jvm的方法区就是计算机的硬盘，这部分的内容属于不断电的永久区。**

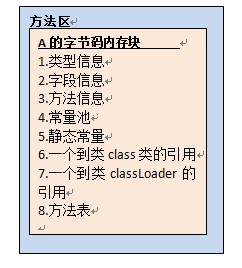
****

**Jvm里还有一个pc寄存器，保存的是下一条将要执行的指令的地址**

**ClassLoader生存在堆中，他会把A.class装载到jvm的方法区（这一步的操作相当于把class文件从操作系统复制到jvm的硬盘中）。这个class文件会被虚拟机解释执行，生成一个A的字节码文件，保存在方法区中，然后再堆内存中生成了一个A字节码的class对象。**

**这个存储在方法区中的A字节码文件中保存有2个引用，一个是指向A的class对象，一个指向加载自己的classLoader**

**这个字节码文件除了保存这2个，还保存了其他什么东西呢？他几乎把类的结构完整的保存到内存中了**

****

**package test;import java.io.Serializable;public final class ClassStruct extends Object implements Serializable {//1.类信息**

**//2.对象字段信息**

**private String name;**

**private int id;**

**//4.常量池**

**public final int CONST\_INT=0;**

**public final String CONST\_STR="CONST\_STR";**

**//5.类变量区**

**public static String static\_str="static\_str";**

**//3.方法信息**

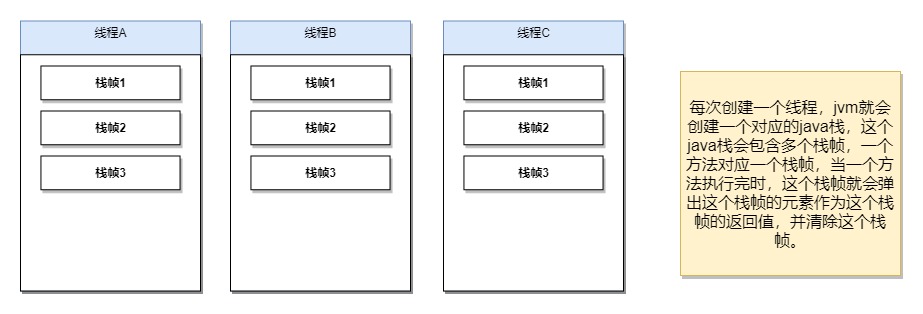
**public static final String getStatic\_str ()throws Exception{**

**return ClassStruct.static\_str;**

**}}**

**A字节码文件中保存了class对象的引用，所以想获得类的信息只能先获得这个对象，所以就得this,getClass().getName**

**每次创建一个线程，jvm就会创建一个对应的java栈，这个java栈会包含多个栈帧，一个方法对应一个栈帧，当一个方法执行完时，这个栈帧就会弹出这个栈帧的元素作为这个栈帧的返回值，并清除这个栈帧。**



**最先被调用的函数的栈帧会最先被压进栈，这个时候只有这个栈帧的本地变量可以被操作栈使用，当这个栈帧需要调用其他方法时，另一个方法的栈帧被压进栈，第二个方法的栈帧变成栈顶，只有他的本地变量能被使用，当第二个函数处理完并返回，他的返回值会变成第一个栈帧的操作栈中的一个操作数**

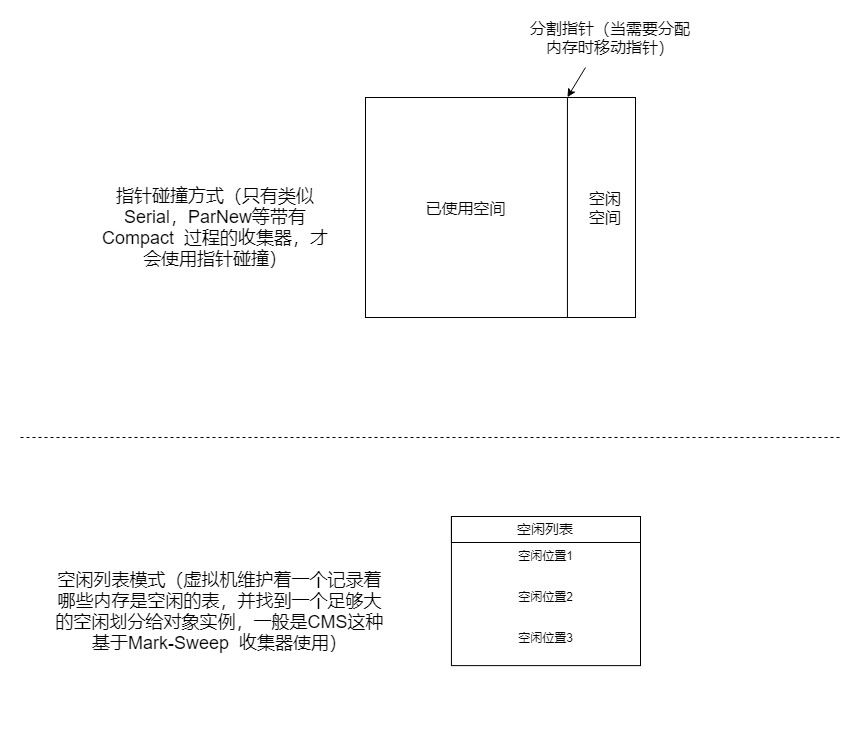
**新生代（Eden，From Survivor，To Survivor）使用Minor GC---> 年老区（Old）使用Full GC=8:1:1:20**

**方法区：储存类结构，属于堆的一部分**

**运行时常量池：是方法区的一部分。存储常量，运行时期常量池也是可以改变大小的，当string调用intern时，会现在常量池中找有没有，如果没有加入常量池并返回，这时可改变常量池大小，常量大小是不可改变的。**

**本地方法栈：为java中的那些native方法服务**

**创建对象时，需要从堆中先划分一份内存，一般有2种方式，一种是已使用内存存放在一边，空闲内存存放在另一边，划分内存时，分隔指针只需要移动一段所需要的距离即可，这种方式称为指针碰撞，另一种方式是，将所有空闲的空间记录在一个表内，划分时在表中寻找一份足够大的内存分配出来。这种方式称为空闲列表**



**分配对象空间不是线程安全的，因为在并发情况下有可能出现同一块内存被同时分配的情况，一般解决办法2种，1是采用CAS配上失败重试保证更新的原子性（要么全部执行，要么全部不执行），2是采用本地线程分配缓冲（TLAB），就是每个线程预先分配一块不重叠的空间，这样只有在预空间分配完了才需要同步锁定，是否使用TLAB由-XX:+/-UseTLAB设定**

**分配空间后会初始化为零值，这表明对象的实例字段可以不被赋予初始值就可以直接使用。**

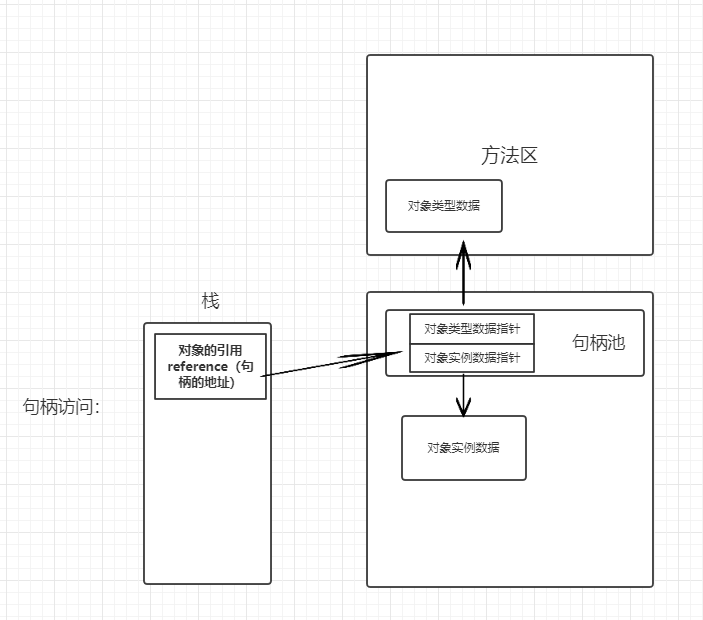
**对象的内存布局：**

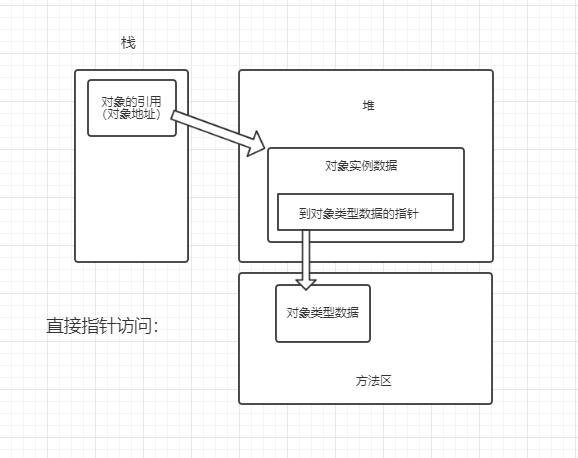
**对象头（Header）：哈希码，GC分代年龄，等等**

**实例数据（Instance Data）：类的字段内容，父类的字段内容，都会被记录，并且相同长度的数据类型会被放在一起，满足这个前提，父类的变量会出现在子类之前**

**对齐填充（Padding）：对象起始地址必须为8字节的整数倍，对象头一般正好是整数倍，当实例数据没有对齐时，使用对齐填充补全**

**对象被分配后，如何使用呢？**





**如果线程请求的栈深度大于虚拟机所允许的最大深度，将抛出StackOverflowError异常。**

**如果虚拟机在扩展栈时无法申请到足够的内存空间，则抛出OutOfMemoryError异常。**

**问题：栈是不是有一个最大值，在栈到达最大值之前会一直进行扩展，当实际内存已满，却还没有到达最大值时抛出OutOfMemoryError，当栈已经到达最大值时，再进行请求生成栈就会抛出StackOverflowError**

**一般来说，虚拟机提供了参数控制java堆和方法区2部分内存的最大值，所以**

**虚拟机栈和本地方法栈的大小=2GB（32bit）-Xmx-MaxPermSize-程序计数器（忽略不计）**

**当出现OutOfMemoryError时，可以通过减少栈容量或者减少最大堆来换取更多线程**

**在jdk1.7版本以上，字符串常量已经从永久代转移到堆中，并且jdk1.8不存在永久代的说法，只有元空间，他并不存在虚拟机中，而是存在本地内存中。**