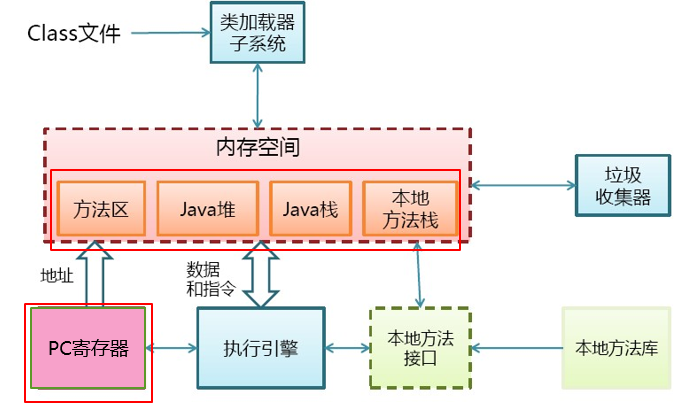
1. **JVM分区，基本结构**



* **堆**

1. 堆中存储的都是对象信息，每一个对象都有一个class字节码文件与之对应
2. 堆中的数据是线程之间共享的，他只保存对象实例本身，而不保存引用类型，比如A a = new A()，引用类型的a是保存在栈中的，而A实例本身是保存在对中的。

* **方法区**

方法区是**线程共享**的，保存了类的源信息，举几个例子：

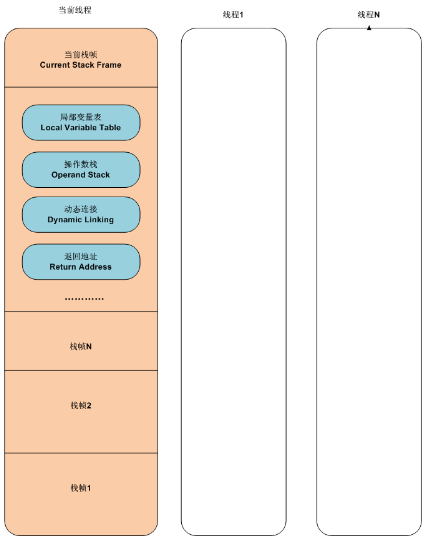
1. 类的完整类名
2. 类的父类的完整类名
3. 类的类型修饰符(public,final,abstract….)
4. 实现的接口interface的有序列表
5. 域(Field)的信息，比如说成员变量之类的
6. 方法(Method)信息，就是我们在类中创建的一个个方法

保存方法的那些信息呢？比如方法名，方法的修饰符，方法的返回类型，有序的方法的参数列表(知道重载是怎么判断的吗)

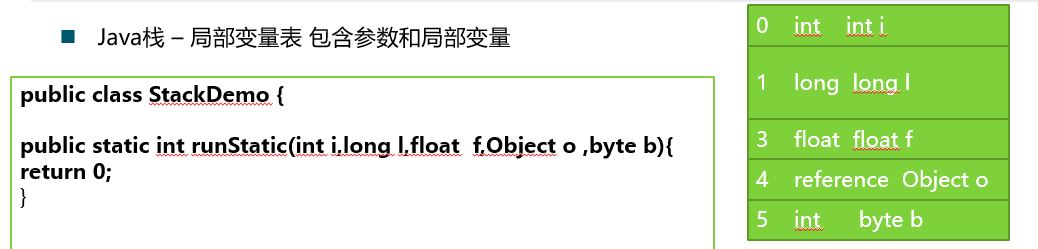
1. 静态变量信息
2. 类型常量池信息，不过String常量池在JDK7的时候就从方法区变为堆了

* **栈**

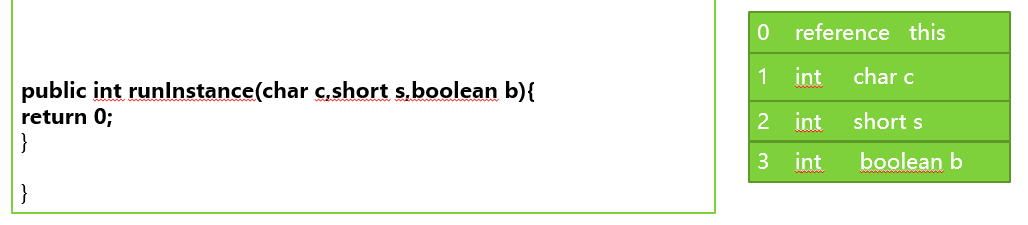
1. 栈是**线程私有**的，是运行时创建的，每启动一个线程，就创建一个栈。栈存储的都是当前线程运行时相关的信息，和堆区别一点就是对是线程共享的，是用来存储数据的。栈相当于处理逻辑，而堆相当于数据。
2. 通俗易懂的说，栈中存放着基本数据类型和引用类型
3. 栈是由一个一个**帧**组成的，因此JAVA中的栈也被称为**帧栈**
4. 帧保存一个方法里的**局部变量表，操作数栈，常量池指针**



1. 每一次方法**调用**，就**创建**一个帧，然后向**栈内压入**，这个帧包含了调用的方法的相关信息
2. **局部变量表**：存储了方法相关的局部变量，这个参数是一个非常宽泛的概念，不仅仅是方法体内部的局部变量，还有调用方法需要传入的参数。我们可以通过**索引**的方式访问到局部变量表中存储的参数。

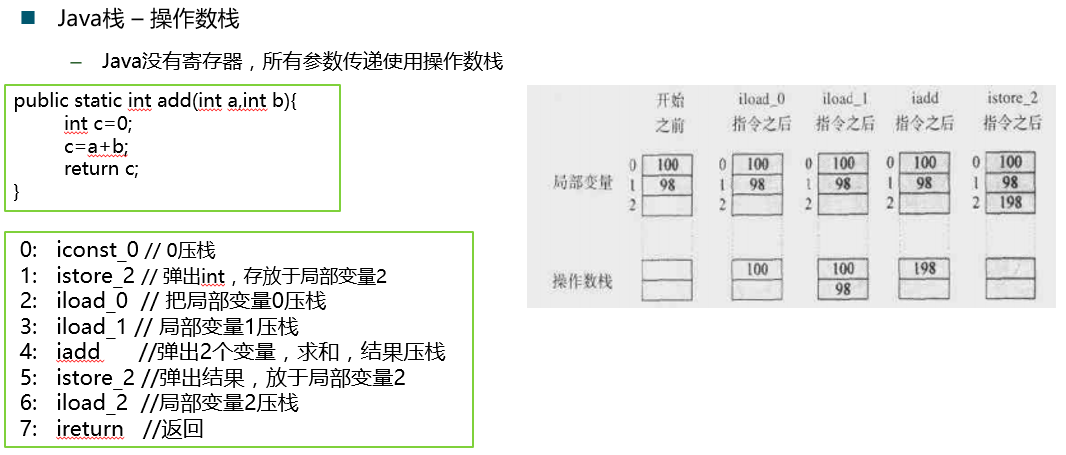


上图为一个**静态方法**的局部变量表，从0开始利用**索引**来存储参数，每个**slot**(槽位)可以占据32位的数据类型。可以看到long类型占的表空间比较大，因为long的字长比较长为64位，因此就占用了两个**slot**。Object是一个对象，是引用类型的，一个指针，他也是32位的。



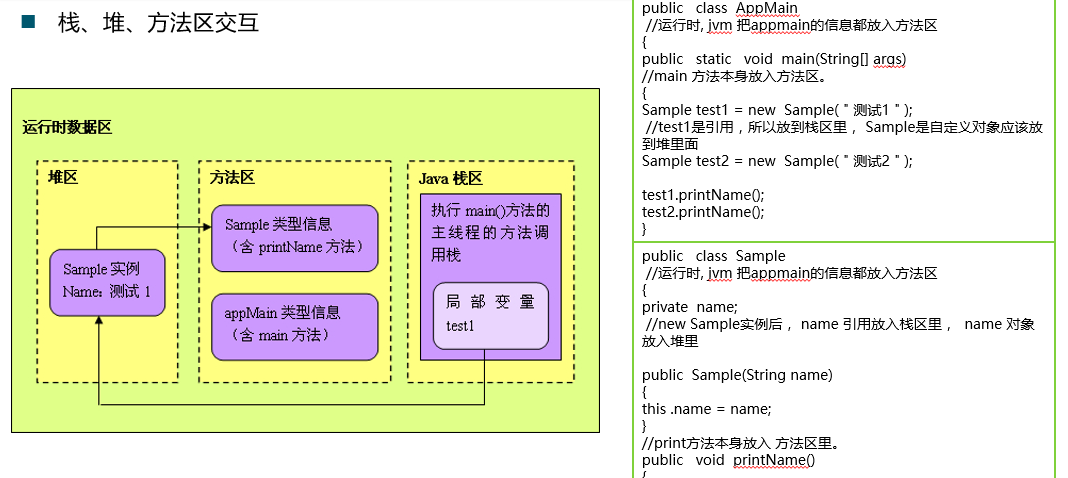
上图是一个**非静态方法**的局部变量表，第一个引用**this**是调用该方法的当前对象的引用。

1. **操作数栈**：和局部变量表不同，操作数栈不是依靠索引来访问的，他是通过入栈和出栈来操作的。也就可以理解为，操作数栈只是一个数据进行存储计算时的临时存储数据的区域。



平时我们说的堆是用来存放对象的，不过小对象(几十个bytes的大小)是可以直接存放在栈上的，因为栈的空间是非常小的。这样小对象可以自动回收，缓解GC的压力。

* **一幅图来理解堆，栈，方法区的区别和联系**



类本身的信息(域信息，方法信息等等)保存在方法区中，方法中新创建的实例存在于堆中，而引用test存在于栈中。重复那句话，栈代表了程序运行时线程之中的业务逻辑，而堆代表了其中需要存储的数据。

如果更加上升到一个面向对象的角度来说的话，一个对象，是由属性(数据)和行为来组成的，对象的属性数据存放在JVM堆中，而这个对象的行为，也就是方法，运行逻辑，是保存在JVM栈之中的。我们在编写对象的时候，其实就是编写了这个对象数据结构保存在堆之中，同时也编写了处理这个对象之中数据的逻辑。