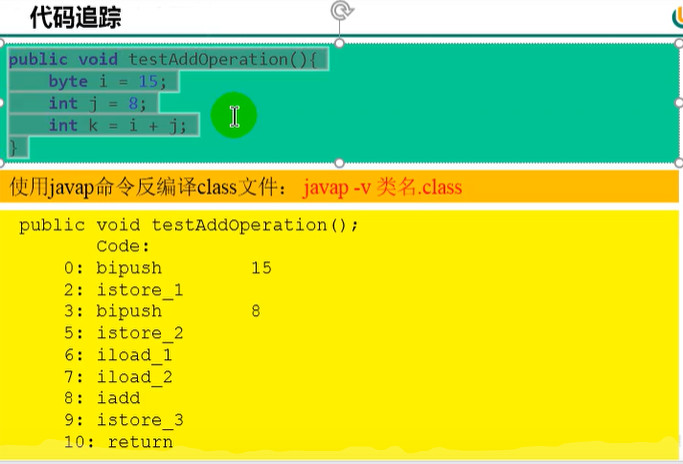
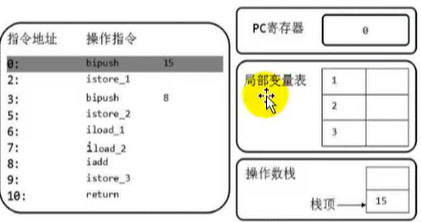
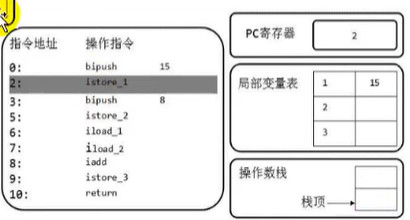
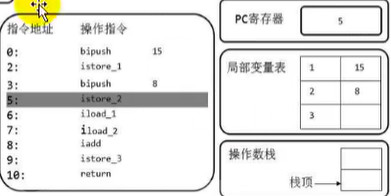
操作数栈实例代码演示：

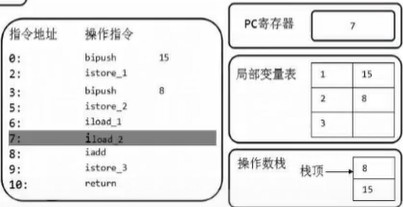


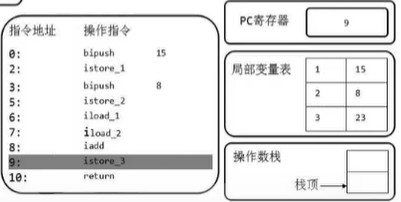
指令逐步执行，虚拟机栈以及PC寄存器的内部情况变化：



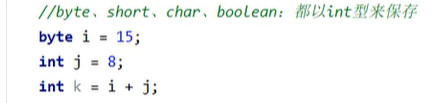




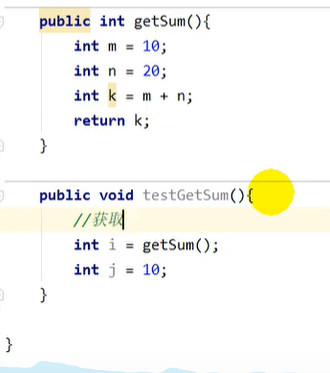


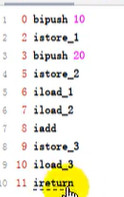


（想想为什么总是先将数据放在操作数栈中然后再放入局部变量表中？局部变量表是一个静态角度的数值暂存的空间，而操作数栈是当前正在操作、计算的数值，反应的是当前指令执行所用的数值的情况，然后再考虑是否存入局部变量表中。想想其实局部变量的声明赋值也是一个操作数值的过程，所以应该先放在操作数栈中因为需要后续使用所以需要存入局部变量表中）



带返回值的情况：（具体的执行过程：先调用testGetSum（）故会先创建栈帧，将其入Java栈，开始执行指令，第一步就是定义一个i变量并且赋予getSum()的值，所以先要将后者的执行结果放入到操作数栈，从字节码指令先从局部变量表中取出this，再取到常量池中的的getSum方法的符号引用，准备执行该方法，于是将该方法入栈，修改PC寄存器值，开始执行该方法，执行完毕后，将值进行返回到testGetSum（）的操作数栈中，再将这个值放入其局部变量表中）（这些调用是在编译期就确定好了的，用顺序执行的思路想想就能想通了）







**1、aload**

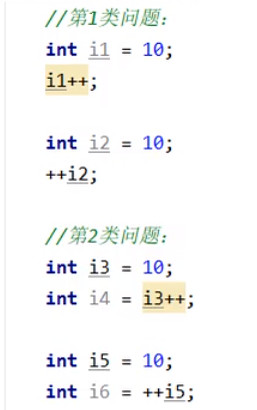
从局部变量表的相应位置装载一个对象引用到操作数栈的栈顶

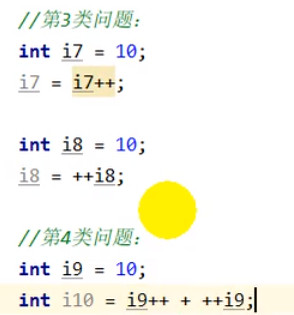
aload\_0把this装载到了操作数栈中aload\_0是一组格式为aload\_的操作码中的一个，这一组操作码把对象的引用装载到操作数栈中标志了待处理的局部变量表中的位置，但取值仅可为0、1、2或者3。

**2、iload\_,lload\_,fload\_,dload\_**

还有一些其他相似的操作码用来装载非对象引用，包括iload\_、lload\_、fload\_和dload\_，这里的i代表int型，l代表long型，f代表float型以及d代表double型。在局部变量表中的索引位置大于3的变量的装载可以使用iload、lload、fload,、dload和aload，这些操作码都需要一个操作数的参数，用于确认需要装载的局部变量的位置。

字节码角度解释面试问题：





（看看字节码指令就行）