1. JVM内存结构

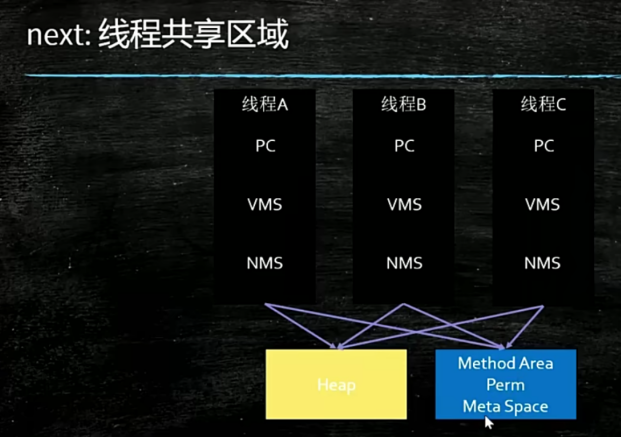
PC：程序计数器

VMS：JVM方法栈

NMS：本地方法栈

Heap：堆区

Method Area：方法区，实现方式（1.8之前，使用Perm永久代实现；1.8之后，使用metaSpace元空间实现）







1. 栈帧Frame

一个方法一个栈帧



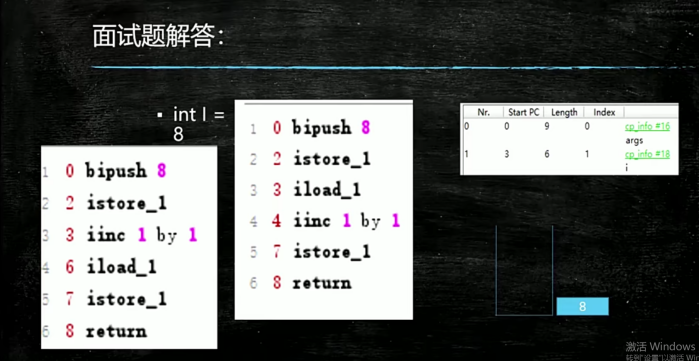
1. Local Variable Table局部变量表
2. Operand Stack操作数栈
3. Dynamic Linking：动态链接是一个将符号引用解析为直接引用的过程。java虚拟机执行字节码时，遇到一个操作码，操作码第一次使用一个指向另一类的符号引用，则虚拟机就必须解析这个符号引用。解析时需要执行三个基本的任务：

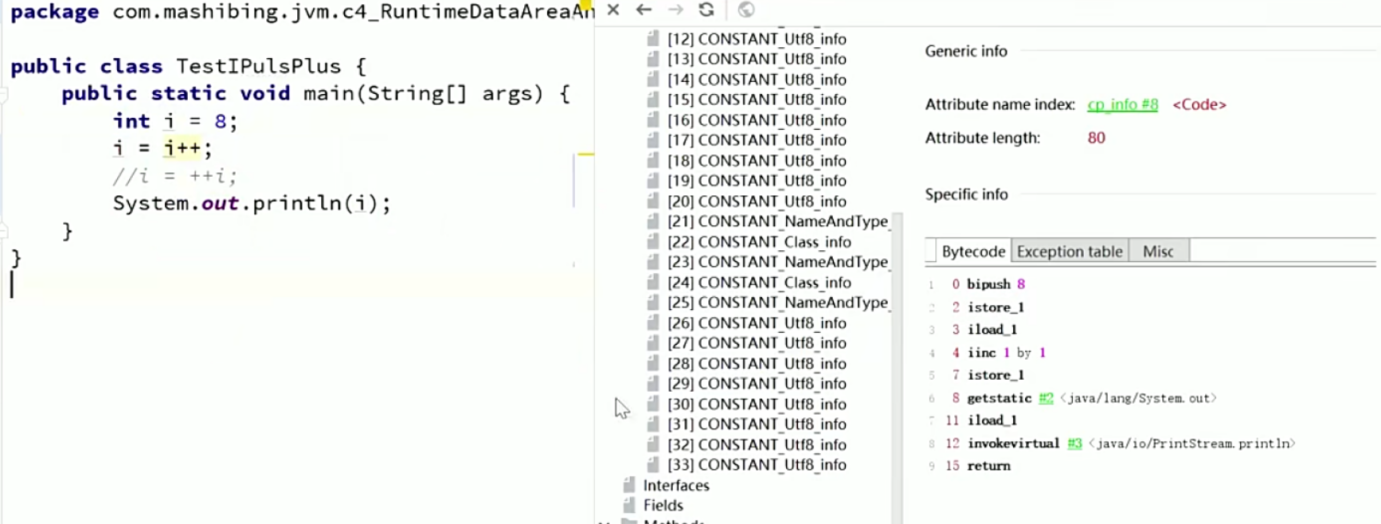
1.查找被引用的类(有必要的话就装载它，一般采用延时装载)。

2.将符号引用替换为直接引用，这样当再次遇到相同的引用时，可以使用这个直接引用，省去再次解析的步骤。

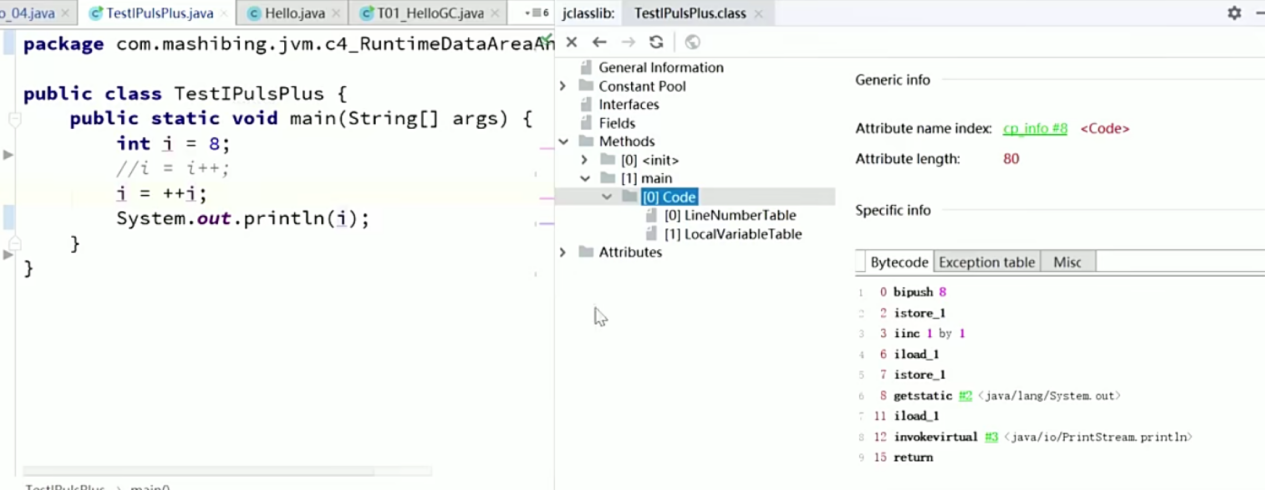
3.当java虚拟机解析一个符号引用时，class文件检查器的第四趟扫描确保了这个引用时合法的。

1. Return Address：方法返回值得地址

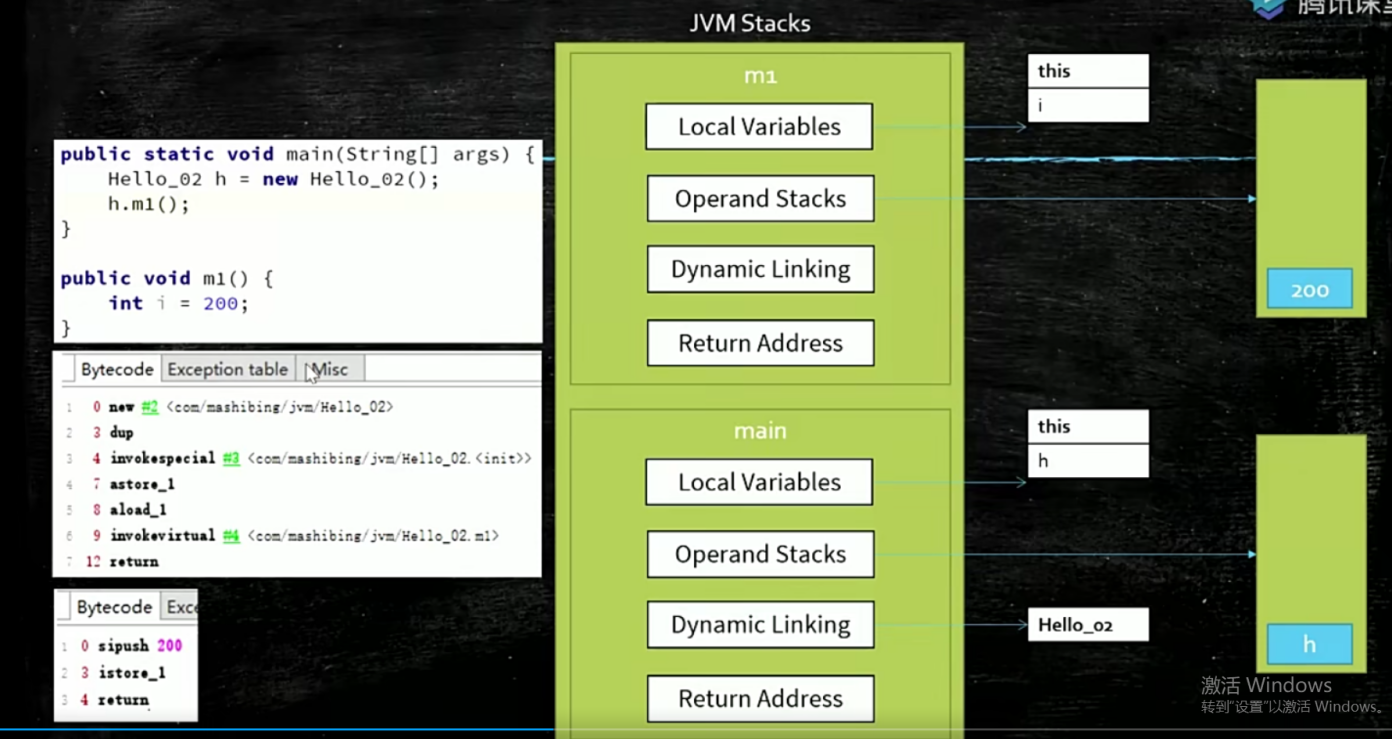




1. bipush 8: 将8压入操作数栈
2. istore\_1：将8出栈，并赋值给局部变量表中下标为1（下标为0的是 this 变量）的变量a，完成赋值语句。
3. iload\_1：因为要对a进行赋值操作，把8压入操作数栈。
4. iinc 1 by 1：但是在上一步最终完成赋值之前，要对a进行++操作，所以直接将局部变量表中下标为1即变量a进行加1，此时a = 9。
5. istore\_1：将操作数的的数弹出并赋值给a，a 又变成了8。所以最后输出8



1. bipush 8: 将8压入操作数栈
2. istore\_1：将8出栈，并赋值给局部变量表中下标为1的变量a，完成赋值语句。
3. iinc 1 by 1：因为这里是先自加再赋值，所以将a + 1变成9。
4. iload\_1：现在执行赋值操作，所以先让9入操作数栈。
5. istore\_1：让11出栈，给a赋值。最终a = 9。



常用指令

