


- 一个线程中方法的调用链可能会很长，很多方法都同时处于执行状态。对于JVM执行引擎来说，在活动线程中，只有位于JVM虚拟机栈**栈顶**的元素才是有效的，即称为**当前栈帧**，与这个栈帧相关连的方法称为**当前方法**，定义这个方法的类叫做**当前类**。
- 执行引擎运行的所有字节码指令都只针对当前栈帧进行操作。如果当前方法调用了其他方法，或者当前方法执行结束，那这个方法的栈帧就不再是当前栈帧了。
- 调用新的方法时，新的栈帧也会随之创建。并且随着程序控制权转移到新方法，新的栈帧成为了当前栈帧。方法返回之际，原栈帧会返回方法的执行结果给之前的栈帧(返回给方法调用者)，随后虚拟机将会丢弃此栈帧。

## 对象和引用

### 对象

在HotSpot虚拟机中，对象在内存中的布局分为3部分：对象头、实例数据、对齐填充。

1. **对象头**：分为两部分。①第一部分存储对象的运行时数据，如哈希码、GC分代年龄、状态标志。②类型指针，即对象指向他自己的类元数据的指针，虚拟机通过这个指针来确定对象是哪个类的实例。
2. **实例数据**：存储对象的真正有效信息，也就是程序代码中所定义的字段内容。
3. **对齐填充**：HotSpot虚拟机中规定，对象头的起始地址必须是8的倍数，当实例数据长度不齐的时候需要对齐填充来补全。

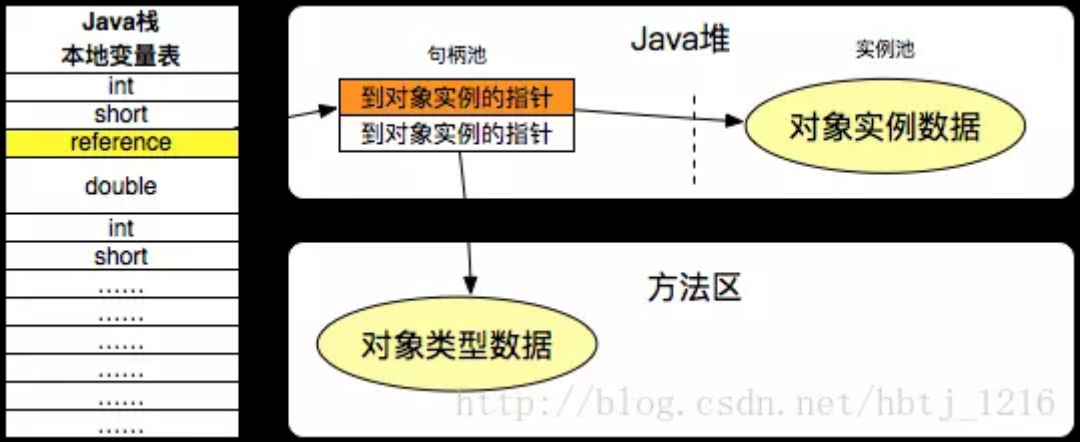
### 引用

创建引用的时候可以指定关联的引用队列，当GC释放对象内存的时候，会将引用加入到引用队列中。引用队列中的对象在内存被回收之前会采取一些机制，类似于与后置通知。

- **强引用**。new出来的，只要引用在，其对象就不会被回收。
- **软引用**。内存不够时才进行回收。比如浏览器中的后退功能，打开新页面时，把旧页面的引用置为软引用。
- **弱引用**。每次GC时都会被回收。
- **虚引用**。get方法总是返回null。GC时不会被立马清除，会被放入引用队列，做一些后置工作。且必须和引用队列一起使用。

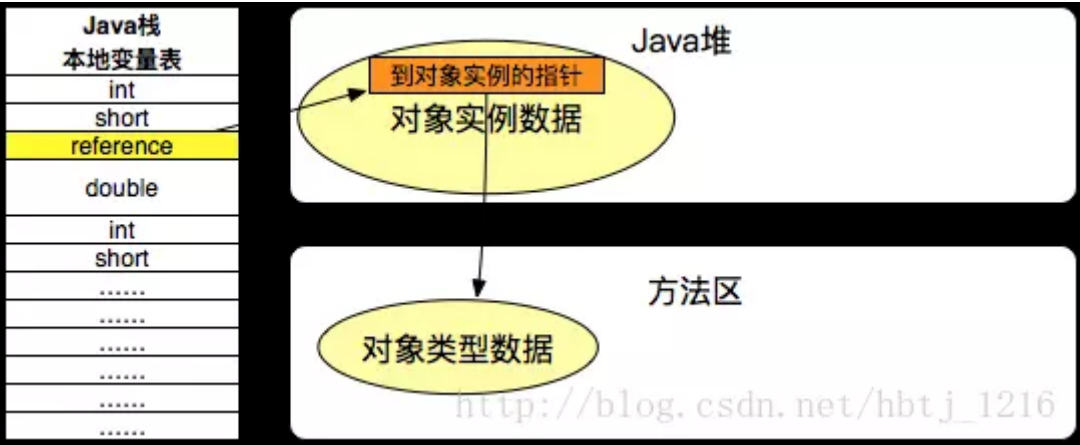
# 访问对象的方式

## 通过句柄访问对象



- 优点：reference存储的是稳定的句柄地址，在对象被移动（垃圾收集时移动对象是非常普遍的行为）时只会改变句柄中的实例数据指针，而reference本身不需要改变。
- 缺点：增加了一次指针定位的时间开销。

## 通过直接指针访问对象（HotSpot使用的方式）



- 优点：节省了一次指针定位的开销。
- 缺点：在对象被移动时reference本身需要被修改。

# 方法区

方法区又被称为静态区，是线程共享的区域。用于存储虚拟机加载的类信息、常量、静态变量。运行时常量池是方法区的一部分。

对于HotSpot来说，Class对象比较特殊，它虽然是对象，但是存储在方法区内。

