
HotSpot虚拟机对象
对象创建
虚拟机遇到创建对象指令，如new时
第一步：检查这个指令的参数是否能在常量池中定位到一个类的符号引用，并且检查这个符号引用代表的类是否被加载、解析和初始化过，如果没有，则先执行类的加载过程
第二步：虚拟机为新生对象分配内存
内存分配方式
指针碰撞（Bump the Pointer）
使用Serial，ParNew等带Compact过程的GC器时
并发时，如何保证？
CAS：Compare and Swap
CAS有3个操作数，内存值V，
旧的预期值A，要修改的新值B。
当且仅当预期值A和内存值V相同时，
将内存值V修改为B，否则什么都不做
CAS配上失败重试保证更新的原子性
本地线程分配缓冲（Thread Local Allocation Buffer，、TLAB），通过参数：-XX:+/-UseTLAB 控制
空闲列表（Free List）
使用CMS这种基于Mark-Sweep算法的GC器时
第三步：虚拟机对对象进行必要的设置，如：对象是哪个类的实例，对象的GC分代年龄等
第四步：进行真实初始化，按程序员的意愿进行初始化
至此，对象才真正创建完成，可以使用了
对象内存布局
对象头（Header）
第一部分：存储对象自身的运行时数据，如：哈希码、GC分代年龄、锁状态标志、线程持有锁等，
这部分数据在32位与64位虚拟机中分别为32bit和64bit，官方称之为：Mark Word
第二部分：类型指针，即对象指向它的类元数据指针，虚拟机通过这个指针来确定对象是哪个类的实例。
如果对象是数据，则还必须有一块用于记录数组长度的数据
实例数据（Instance Data）
对象真正的有效信息，也是在程序代码中所定义的各种类型的字段内容
HotSpot-VM要求对象起始地址必须是8字节的整数倍，
即对象大小必须是8字节的整数倍
对齐填充（Padding）
非必须，起点位符的作用，在不够位数的情况下才有，
如：对象大小是63位，此时就需要额外的1个占位符，凑足64，即8的整数倍
对象访问定位
使用句柄
需要Java-Heap划分出一块内存来作为句柄池，栈中的reference中存储句柄地址，
句柄中包含了对象实例数据与类型数据各自的具体地址
直接指针
reference中存放对象地址
Sun HotSpot默认的对象方法策略