finalization机制

Object对象提供了finalize方法来添加对象的自定义销毁逻辑。当垃圾回收器发现没有引用指向一个对象时，会调用该对象的finalize方法。

由于finalize方法的存在，对象的状态可分为三种。可达状态、复活状态、不可达状态。

- **可达状态。根据引用类型的不同，可分为强引用可达、软引用可达、弱引用可达。**
- **复活状态。当jvm发现没有引用指向一个对象时，会调用该对象的finalize方法，在finalize方法中可能为当前对象添加新的引用，所以finalize方法执行完之后，jvm会重新检测对象的可达性。如果检测到该对象可达，那么会复活该对象。复活之后，如果重新变为没有引用指向该对象，那么该对象会直接变为不可达状态，而不会再一次执行finalize方法。也就是说，finalize方法只会执行一次。**
- **不可达状态。这种状态下jvm会释放对象所占的空间。**

虚拟机栈

通过栈指针可以访问处理器。栈是一种快速有效的分配存储方法，访问速度仅次于寄存器。通过栈指针的上下移动来动态调整空间。可以认为，这一方式约束了栈的灵活性。

总的来说，栈的优势是访问速度比堆要快，且栈数据是可以被共享的。缺点是栈里面的数据大小与生存期必须是确定的，从这一点来看，栈明显缺乏灵活性。

一个栈帧需要分配多少内存，不会受到程序运行期变量数据的影响，而仅仅取决于具体的虚拟机实现。

逃逸分析

根据逃逸分析原理对JVM进行优化。首先需要找到未逃逸的变量，将该变量的类实例直接在栈上分配。线程结束时，栈被回收，局部变量对象也被回收。