Java对象的内存布局

创建java对象的方式

1. New语句
2. 反射机制
3. Object.clone()方法
4. 反序列化
5. Unsafe.allocateInstance

Object.clone()和反序列化通过直接复制已有的数据，来初始化新建的对象实例字段。

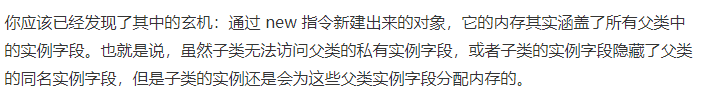
Unsafe.allocateInstance没有初始化实例字段。

New和反射机制，通过调用构造器来初始化实例方法。

构造器调用分为：

1. 显式
   1. 直接使用“super”关键字调用父类构造器。
   2. 使用“this”关键字调用同一个类中的其他构造器。
2. 隐式
   1. 父类存在无参数构造器，java构造器会自动添加对父类构造器的调用。

总而言之，当调用一个构造器的时候，它将优先调用父类的构造器，直至Object类。这些构造器的调用者皆为同一个对象，也就是通过new指令新建而来的对象。



接下来，分析下这些字段在内存中的分布

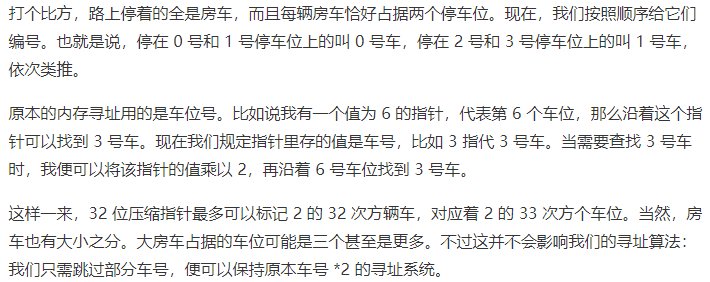
## 压缩指针

在64虚机中：

对象头标记字段占64位，类型指针占64位。也就是说每个java对象在内存中的额外开销就是16个字节。

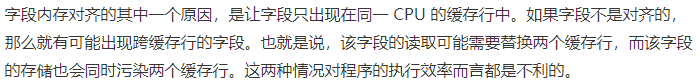
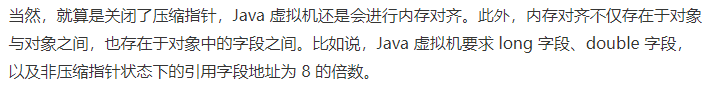
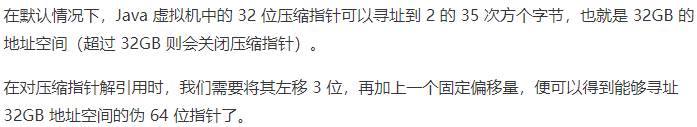
对于Integer，仅有一个int类型的私有字段占用4字节。因此一个Integer对象的额外开销就是至少400%，这就是java要引入基本类型的原因。





每辆车从偶数位停起，这个概念我们称之为内存对齐（对应虚机选项-XX:ObjectAlignmentInBytes，默认值为8）

默认情况下，java虚机堆中对象的起始地址需要对齐至8的倍数。如果一个对象用不到8N个字节，那么空白的那部分就浪费掉了。这些浪费的空间称为对象间的填充（padding）



## 字段重排

Java虚机重新分配字段你的先后顺序，以达到内存对其的目的。

三种排列方法，对应虚机中选项（-XX:FieldsAllocationStyle，默认值为1），遵循下面原则：

1. 如果一个字段占C字节，那么改字段偏移量需要对齐至NC。这里偏移量指的是字段地址与对象的起始地址的偏差。
2. 子类所继承的字段偏移量，要和父类的偏移量保持一致。

