# java

1. hashCode与equals
   1. hashCode获取哈希码（散列码），用来确定对象在哈希表中的索引位置
   2. 为什么要有hashCode
      1. 拿HashSet举例
      2. 对象加入HashSet时，它会先计算这个对象的HashCode值来判断对象加入的位置，查看这个位置是否有值，如果没有，就直接添加进去
      3. 如果有，会调用equals方法来检查两个对象是否相等，如果相等，就不存这个对象，不同就散列到其他位置，这样可以大大减少equals的次数啊，提高执行速度
   3. hashcode和equals为什么要一起重写
      1. hashcode是对堆上的对象产生一个值，如果没有重写的话，那么这个类的两个对象是不会相等的
2. final修饰基本类型和引用类型变量的区别
   1. 修饰基本类型变量，不能对其重新复制
   2. 引用类型变量，它保存的只是一个引用，fianl只保证这个引用所引用的地址不会发生改变，也就是说一个引用着同一个对象，但是这个对象是可以改变的
3. HashMap和HashTable有什么区别？其底层实现是什么？
   1. HashTable是线程安全的
   2. HashMap允许key和value为null
   3. 底层实现：链表+数组
   4. jdk8之后链表高度为8，数组长度超过64，链表就会变为红黑树
   5. map.put(k,v)实现原理
      1. 1.先将k，v封装到Node对象中
      2. 2.底层调用hashcode方法得到哈希值
      3. 3.通过哈希函数/算法，将哈希值转化为数组的下标
      4. 如果下标位置上没有元素，就将这个节点添加到该位置。
      5. 如果存在链表，就会拿着k和链表上的每一个节点中的k进行equals
      6. 如果返回的是false，那么这个新节点会被添加到链表的末尾
      7. 如果其中一个返回了true，那么这个节点的值会被覆盖
   6. map.get（k）实现原理
      1. 1.先调用hashCode方法得到哈希值，通过哈希算法得到数组下标
      2. 2.快速定位到这个位置，如果啥也没有，返回null，如果有单向链表，则拿参数k与单向链表的每个节点k进行equals比较
      3. 3.如果返回false，则get方法返回null，只要其中一个节点返回true，则这个节点就是我们需要的value，就将其返回
4. 类加载器与双亲委派机制
   1. BootstrapClassLoader（启动类加载器）爷
      1. lib/rt.jar
   2. ExtClassLoader（扩展类加载器）父
      1. lib/etc/\*
   3. AppClassLoader（应用类加载器）自定义
      1. classpath中的类
   4. 在代码执行之前会将所需要的类全加到JVM中，它就会去找String.class
   5. 首先会去启动类加载器去加载，如果没有，那么再去扩展类加载器加载，如果还是没有，那就是到应用类加载器了，直到加载到为止
   6. 好处
      1. 为了安全性，避免用户自己编写的类替换Java的核心类
5. GC如何判断对象可以被回收
   1. 引用计数法
      1. 每个对象有一个引用计数属性，新增一个引用计数就+1，释放就-1.为0时即可回收
   2. 可达性分析法