## 2020/08/02===============================

1. 哈希表怎么实现，冲突怎么解决
   1. 哈希表的底层数据结构是数组，很多地方也叫Bucket。首先通过将key的值传给hash函数，求出对应的索引，找到相应的下标进行存储，时间复杂度是O（1）。
   2. 解决方法：
      1. 开放定址法
      2. 再hash法
      3. 链地址法（HashMap）

## 2020/08/02===============================

1. 快排优化策略
   1. 基准值选择
      1. 固定基准
      2. 随机基准
      3. 三值取中
   2. 优化策略
      1. 插入排序
      2. 尾递归优化(递归最后，返回无需求)
      3. 聚集元素
      4. 多线程
2. 排序的稳定性
   1. 稳定：冒泡、插排、归并
   2. 不稳定：选择、快排、希尔、堆
3. priorityQueue
   1. add()和offer()
   2. element()和peek()
   3. remove()和poll()

以上均为，前者抛出异常，后者返回false/null

## 2020/07/31===============================

1. i++是线程安全的吗？

不是。每个线程有自己的工作线程，线程将共享变量从主内存中load到自己的工作内存中，完成操作再save到主内存中。还没有更新到主内存，但是另一个线程读到了主内存，这时的数据是脏数据，会覆盖其它线程计算的值。

volatile只能保证可见性，不能保证原子性。

解决方法：使用synchronized或者AtomicInteger系列的类，同时也会增加性能开销。

## 2020/07/29===============================

1. 三种注入方式
   1. 构造器注入
   2. setter注入
   3. 注解注入

## 2020/07/28===============================

1. 什么是AOP？

AOP是面向切面编程，面向对象开发中代码逻辑是自上而下的，在这些自上而下的过程中会产生一些横切性问题，如日志记录、权限验证、事务管理，这些横切性问题和主业务逻辑关系不大，并且散落在代码的各个地方难以维护。

AOP的思想就是把业务逻辑和横切性问题进行分离，达到解耦的目的，提高代码重用性，提高开发效率。

1. 什么是IOC？

将本来由程序来实例化对象的工作，交给spring容器，并将实例化的对象注入到引用中。

\*\*\*\*\*\*\*\*待解决问题

1. 介绍一下常用linux命令
2. 介绍一下常用设计模式
3. http1.1和http2.0；http和https
4. 事务，范式，Hytrix。
5. Sleep和wait区别
6. Final，finally，finalize
7. Springboot

## 2020/07/23===============================

1、HashMap链表使用红黑树而不是AVL树

在Jdk1.8版本后，Java对HashMap做了改进，在链表长度大于8的时候，将后面的数据存在红黑树中，以加快检索速度。

两者查找速度都是logN。红黑树适用于插入删除，平衡数适用于查找。因为红黑树旋转次数少，平衡数深度小。

2、为什么不一直用树

内存占用和查找复杂性之间的权衡。大多数哈希函数将产生非常少的冲突。

## 2020/07/23===============================

1、 进程和线程的区别

进程是程序运行和资源分配的基本单位；线程是cpu调度和分派的基本单位，是能够独立运行的基本单元；同一进程的多个线程并发执行。

2、 Forward和Redirect的区别

Forward是服务器内部重定向，服务器直接访问目标地址的url，把资源响应给浏览器，浏览器地址栏不会变化，浏览器只发送一次请求。

Redirect是服务器返回一个状态码给浏览器，浏览器重新请求新的地址，地址栏，2次请求。

3、 简述TCP和UDP区别

1. TCP是面向连接的，UDP是无连接的。
2. TCP提供可靠的服务，UDP则是尽最大努力交付。TCP有丢包的重发控制和乱序的顺序控制。UDP有较好的实时性，效率比TCP高。
3. TCP是点到点的，UDP可以一对多，多对一，一对一，多对多。
4. TCP对资源要求多，TCP首部20字节，UDP8个字节。