## 死锁与活锁的区别，死锁与饥饿的区别？

死锁：是指两个或两个以上的进程（或线程）在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。

产生死锁的必要条件：

互斥条件：所谓互斥就是进程在某一时间内独占资源。请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放。不剥夺条件:进程已获得资源，在末使用完之前，不能强行剥夺。循环等待条件:若干进程之间形成一种头尾相接的循环等待资源关系。活锁：任务或者执行者没有被阻塞，由于某些条件没有满足，导致一直重复尝试，失败，尝试，失败。

活锁和死锁的区别在于，处于活锁的实体是在不断的改变状态，所谓的“活”， 而处于死锁的实体表现为等待；活锁有可能自行解开，死锁则不能。s

饥饿：一个或者多个线程因为种种原因无法获得所需要的资源，导致一直无法执行的状态。

Java中导致饥饿的原因：

高优先级线程吞噬所有的低优先级线程的CPU时间。线程被永久堵塞在一个等待进入同步块的状态，因为其他线程总是能在它之前持续地对该同步块进行访问。线程在等待一个本身也处于永久等待完成的对象(比如调用这个对象的wait方法)，因为其他线程总是被持续地获得唤醒。

## Java中用到的线程调度算法是什么？

采用时间片轮转的方式。可以设置线程的优先级，会映射到下层的系统上面的优先级上，如非特别需要，尽量不要用，防止线程饥饿。

## 什么是线程组，为什么在Java中不推荐使用？

ThreadGroup类，可以把线程归属到某一个线程组中，线程组中可以有线程对象，也可以有线程组，组中还可以有线程，这样的组织结构有点类似于树的形式。

为什么不推荐使用？因为使用有很多的安全隐患吧，没有具体追究，如果需要使用，推荐使用线程池。

## 为什么使用Executor框架？

每次执行任务创建线程 new Thread()比较消耗性能，创建一个线程是比较耗时、耗资源的。调用 new Thread()创建的线程缺乏管理，被称为野线程，而且可以无限制的创建，线程之间的相互竞争会导致过多占用系统资源而导致系统瘫痪，还有线程之间的频繁交替也会消耗很多系统资源。接使用new Thread() 启动的线程不利于扩展，比如定时执行、定期执行、定时定期执行、线程中断等都不便实现。

#乐观锁与悲观锁原理及实现

一、乐观锁

 总是认为不会产生并发问题，每次去取数据的时候总认为不会有其他线程对数据进行修改，因此不会上锁，但是在更新时会判断其他线程在这之前有没有对数据进行修改，一般会使用版本号机制或CAS操作实现。

 version方式：一般是在数据表中加上一个数据版本号version字段，表示数据被修改的次数，当数据被修改时，version值会加一。当线程A要更新数据值时，在读取数据的同时也会读取version值，在提交更新时，若刚才读取到的version值为当前数据库中的version值相等时才更新，否则重试更新操作，直到更新成功。

核心SQL代码：

update table set x=x+1, version=version+1 where id=#{id} and version=#{version};

 CAS操作方式：即compare and swap 或者 compare and set，涉及到三个操作数，数据所在的内存值，预期值，新值。当需要更新时，判断当前内存值与之前取到的值是否相等，若相等，则用新值更新，若失败则重试，一般情况下是一个自旋操作，即不断的重试。

一、悲观锁

 总是假设最坏的情况，每次取数据时都认为其他线程会修改，所以都会加锁（读锁、写锁、行锁等），当其他线程想要访问数据时，都需要阻塞挂起。可以依靠数据库实现，如行锁、读锁和写锁等，都是在操作之前加锁，在Java中，synchronized的思想也是悲观锁。

---------------------

原文：https://blog.csdn.net/L\_BestCoder/article/details/79298417