Java高并发

## Java多线程

1. 什么是进程，什么是线程

进程 -> exe文件，程序，线程 -> 进程中的小单位，目标 -> 充分利用资源，提高吞吐量

1. 多线程的实现方式
   1. 继承Thread类
   2. 实现Runnable接口，重现run()方法
   3. 实现Callable接口，call()方法有返回值，可以抛出异常
   4. 线程池，ThreadPoolExecutor，newFixedThreadPool，newCachedThreadPool
2. 线程方法
   1. start() – 新建新线程
   2. run() – 执行原线程
   3. join() – 阻塞线程，插队
   4. interrupt() – 截断线程
   5. sleep() – 线程休眠，不让出资源
   6. yield() – 短暂让出空间

### 线程同步

1. 线程锁的分类
   1. 乐观锁（基于CAS – 保证原子性）

\*拓展，CAS(Compare and swap)，CAS（原值==期望值 : 新的值: ）

* 1. 悲观锁
     1. 排他锁（互斥锁） - 独占
     2. 共享锁 – 只读不写
  2. 自旋锁 – 循环等待，占CPU资源，适用于执行时间短，线程数少
  3. 公平锁(不抢占)与不公平锁(抢占)

1. Synchronized锁
   1. synchronized关键字
   2. synchronized同步块

* 区别，关键字作用于方法整体，同步块作用于代码块
* 锁细化，锁粗化
  1. 锁升级
     1. 无锁态（new）
     2. 偏向锁（BiasedLocking）
        1. 默认延迟4s
        2. 多个线程抢占，升级
     3. 轻量级锁（自旋锁，CAS，无内核状态的锁）
        1. 自适应自旋，JVM托管，升级
     4. 重量级锁（需向内核申请，不占CPU资源）



1. volatile关键字
   1. 用于防止指令重排，保持线程可见性（每次都在内存中重读取）
   2. DCL，double check lock中使用
2. 几种新类型同步锁（基于AQS，astractQueuedSynchronizer）
   1. ReentrantLock(try catch finally)
      1. lock()
      2. unlock()
      3. tryLock()
      4. lockInterruptibly()
   2. CountDownLatch(set count)
      1. 阻塞，减到0时，程序继续运行
   3. CyclicBarrier
      1. 拉闸阻塞，满人发车
   4. ReadWriterLock
      1. 读锁 –> 共享锁 -> 读的过程中不允许写
      2. 写锁 -> 排他锁 –> 写的时候独占
   5. Semaphore
      1. 限流，只能有若干个线程（参数）同时执行
   6. Exchanger
      1. 交换，两两线程之间阻塞交换
   7. phaser
      1. bulkRegister()，参数：总参与线程数
      2. 分阶段，需重现onAdvance方法， case when
      3. arriveAndAwaitAdvance() -> 阻塞 -> onAdvance
      4. arriveAndDeregister，解除注册