多线程与并发

线程池

- 好处
  - 降低资源消耗
  - 提高响应速度
  - 提高线程的可管理性
- 线程池
  - Executors
    - newFixedThreadPool — 创建一个固定的线程池
    - newCachedThreadPool — 创建一个最多缓存60s的线程池
    - 子主题 3
  - 参数不同而已
  - ThreadPoolExecutor
    - corePoolSize — 核心线程数
    - maximumPoolSize — 最大的线程数
    - workQueue — 工作队列的大小
    - keepAliveTime — 超过核心线程数线程的最大存活时间
    - threadFactory — 线程工厂
    - handler（饱和策略）
      - AbortPolicy(抛异常)
      - CallerRunsPolicy(增加队列容量)
      - DiscardPolicy(直接丢弃)
      - DiscardOldestPolicy（丢弃最早没有处理的）
  - ForkJoinPool
    - 分治算法
    - fork分解任务
    - join合并结果

锁

- 死锁条件
  - 互斥条件
  - 请求与保持
  - 不剥夺条件
  - 循环等待条件
- Synchronized关键字
  - 使用
    - 实例方法
    - 代码块
    - 静态方法
  - 锁升级
    - 1. 偏向锁
    - 2. 自旋锁
    - 3. 重量级锁
  - 原理
    - monitorenter
    - monitorexit
    - 修饰方法：ACC_SYNCHRONIZED
- 锁的分类
  - AQS(工具类)
    - CountDownLatch
      - 递减，主线程集合
      - 多个子线程执行任务后执行countDown，主线等待所有线程执行完之后，在主线程获取结果
    - Semaphore — 同时允许的并发数
    - CyclicBarrier — 递减，不阻塞主线程
    - ReentrantLock
      - 公平锁和非公平锁
      - 也可配合Condition使用
  - LockSupport
  - ReadWriteLock — 读写锁分离

集合类

- concurrentHashMap
  - 数组+链表/红黑树
  - jdk7 分段锁，segment[] 降低锁粒度
  - jdk8 CAS+Node锁
- CopyOnWriteArrayList
  - 数组
  - 只添加写锁，最终一致性
  - 迭代器使用的是快照
- CopyOnWriteArraySet
- ConcurrentLinkedDeque
- ConcurrentLinkedQueue
- ConcurrentSkipListMap
- ConcurrentSkipListSet

原子类

- AtomicInteger
- AtomicBoolean
- AtomicLong
- AtomicIntegerArray
- AtomicLongArray
- AtomicMarkableReference
- AtomicReference
- AtomicReferenceArray
- AtomicStampedReference
- DoubleAccumulator
- DoubleAdder
- LongAccumulator
- LongAdder
- Striped64
- AtomicLongFieldUpdater

线程

- 线程的状态
  - NEW（初始化）
  - RUNNABLE（运行） — Thead.start()
  - WAIT（等待）
    - 等待触发
      - LockSupport.park()
      - Thread.join()
      - Object.wait()
    - 解除等待
      - Object.notify()
      - Object.notifyAll()
      - LockSupport.unpark(Thread)
  - TIME_WAIT（超时等待）
    - 等待触发
      - Thread.sleep(long)
      - Object.wait(long)
      - Thread.join(long)
      - LockSupport.parkNanos()
      - LockSupport.parkUntil()
    - 相同
    - 解除等待
  - BLOCKED（阻塞）
    - 等待进入synchronized方法
    - 等待进入synchronized块
  - TERMINATED（结束）
- Volatile关键之
  - 保证变量可见性（内存屏障）
  - 防止指令重排序
- 启动main方法，默认线程数是5
- 上下文切换
- 守护线程 — 指的是主线程结束，守护线程也必须跟着结束
- ThreadLocal — 线程本地变量