# 在java中守护线程和本地线程区别？

java中的线程分为两种：守护线程（Daemon）和用户线程（User）。

任何线程都可以设置为守护线程和用户线程，通过方法Thread.setDaemon(bool on)；true则把该线程设置为守护线程，反之则为用户线程。Thread.setDaemon()必须在Thread.start()之前调用，否则运行时会抛出异常。

两者的关系：

唯一的区别是判断虚拟机(JVM)何时离开，Daemon是为其他线程提供服务，如果全部的User Thread已经撤离，Daemon 没有可服务的线程，JVM撤离。也可以理解为守护线程是JVM自动创建的线程（但不一定），用户线程是程序创建的线程；比如JVM的垃圾回收线程是一个守护线程，当所有线程已经撤离，不再产生垃圾，守护线程自然就没事可干了，当垃圾回收线程是Java虚拟机上仅剩的线程时，Java虚拟机会自动离开。

# 线程与进程的区别？

进程是操作系统**分配资源**的最小单元，线程是操作系统**调度的最小单元**。

一个程序至少有一个进程,一个进程至少有一个线程。

# 什么是多线程中的上下文切换？

多线程会共同使用一组计算机上的CPU，而线程数大于给程序分配的CPU数量时，为了让各个线程都有执行的机会，就需要轮转使用CPU。不同的线程切换使用CPU发生的切换数据等就是上下文切换。

# 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：

1. 互斥条件：所谓互斥就是进程在某一时间内独占资源。
2. 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。
3. 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。
4. 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，处于活锁的线程是在不断的改变状态、尝试，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。
* 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：
  1. 高优先级线程吞噬所有的低优先级线程的CPU时间。
  2. 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。
  3. 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。

# 为什么使用Executor框架？

1. 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。
2. 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。
3. 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。

# 在Java中Executor和Executors的区别？

* Executors 工具类的不同方法按照我们的需求创建了不同的线程池，来满足业务的需求。例如：固定大小线程池、单线程线程池、带缓存的线程池、调度线程池
* Executor 接口对象能执行我们的线程任务。
* ExecutorService接口继承了Executor接口并进行了扩展，提供了更多的方法我们能获得任务执行的状态并且可以获取任务的返回值。
* ThreadPoolExecutor 可以创建自定义线程池。真正实现执行方法的是AbstractExecutorService
* Future 接口，表示异步计算的结果，他提供了检查计算是否完成的方法，以等待计算的完成，并可以使用get()方法获取计算的结果。真正实现获取结果、查看状态，是FutureTask

# 什么是原子操作？在Java Concurrency API中有哪些原子类(atomic classes)？

* 原子操作：线程要修改共享变量的值，通常需要先把变量值从主存读到线程的私有内存中（读），然后将值进行修改（改），然后将值写回主存中（写）。所谓的原子操作，就是指读、改、写这3步操作是原子的，不可分割的。
* 原子类：AtomicBoolean，AtomicInteger，AtomicLong，AtomicReference
* 原子数组：AtomicIntegerArray，AtomicLongArray，AtomicReferenceArray
* 原子类的原理就是通过CAS算法。Compare & Swap