1. 线程简介
2. 概述

线程作为操作系统调度的最小单元，多个线程能够同时执行。一个进程中可以创建多个线程，这些线程都拥有自己的计数器，堆栈和局部变量等属性，并且能够访问共享的内存变量，CPU在这些线程上高速切换，使用者误以为 是同时执行的。

Java中内置了对多线程的支持，这是它的优势，main()方法就是一个main线程，由它来掉用其他的线程。

1. 线程优先级

操作系统采用时分的形式调度运行的线程，分出一个个时间片，线程会分配到若干内的时间片，当线程的时间片用完了就会发生线程调度，等待下次分配。线程分配的时间片决定了线程使用CPU资源的多少，线程优先级能够决定线程分配的时间片多少。

通过priorty变量来控制优先级，范围1-10，使用setPriority(int)方法来修改优先级，默认是5。对于频繁阻塞的线程设置较高优先级，对于使用CPU时间过多的线程设置较低优先级，防止独占CPU。但是，线程优先级并不一定能够决定CPU资源的分配，有的系统甚至会忽略线程优先级。

1. 线程状态

线程在运行声明周期中有6种不同的状态，我们可以首先通过jps工具查看当前运行的java进程，然后通过jstack工具来查看对应的进程中各个线程的状态。

1. Daemon线程

守护线程，主要用作程序中后台调度以及支持性工作，当一个JVM中不存在非Daemon线程时，JVM会退出。可以通过调用Thread.setDaemon(true)将线程设置为Daemon线程。在JVM退出时，Daemon线程中的finally块并不一定会执行，因此在守护线程中不能依靠finally来确保关闭清理资源。

1. 启动和终止线程
2. 构造线程

在运行线程之前首先要构造一个线程对象，需要提供线程需要的属性，如所属的线程组，线程优先级 ，是否是守护线程等。新线程由其父线程进行空间分配，子线程继承了父线程是否是守护线程，优先级和加载资源的contextClassLoader以及可继承的ThreadLocal，还会分配一个唯一ID标识这个子线程。

1. 启动线程

start()方法的意思是父线程告知JVM，只要线程规划器空闲，立即启动调用start()方法的线程。启动之前，最好为线程设置一个名称，这样在使用jstack分析时能够快速找到。

1. 中断

中断是线程的一个标识位属性，标识一个运行中的线程是否被其他线程进行了中断操作，其他线程通过调用该线程的interrupt()方法对其进行中断操作。线程通过检查自身是否被中断进行响应，通过方法isInterrupted()来判断是否被中断，也可以调用Thread.interrupted()对当前线程的中断标识进行复位。

有许多抛出InterruptedException异常的方法，会将该线程的中断标识位清除。

1. 过期的suspend(),resume(),stop()

暂停，恢复，停止操作，过期的原因是不能保证线程资源的正确释放。如suspend()，调用会，线程不会释放已经占有的资源，而是占着资源进入睡眠 状态，容易引发死锁问题。stop()终结线程时不会保证线程资源的正确释放。

1. 线程间通信
2. volatile和synchronized

Java支持多个线程同时访问一个对象或对象的成员变量，每个线程可以拥有这个对象的拷贝，因此每个线程看到的变量不一定是最新的。volatile可以用来修饰变量，任何对该变量的访问需要从共享内存中获取，对它的改变需要刷新回共享内存中，保证所以线程对变量的可见性。

synchronized可以修饰方法或同步块，确保只有一个线程处于方法或同步块中，保证线程对变量访问的可见性和原子性。通过javap查看类文件，对于同步块的实现，通过monitorenter和monitorexit指令，同步方法则是通过修饰符上的ACC\_SYNCHRONIZED实现的。无论哪种方式，本质都是对一个对象的监控器进行获取，只有一个线程能够获取到监控器。

任何对象都有自己的监控器，若有线程没有获取到某对象的监控器会被阻塞，进入同步队列等待，当获得该对象的线程释放了锁，则该释放操作唤醒阻塞在同步队列中的线程，使其重新尝试获取监视器。

1. 等待/通知机制

线程A调用了对象o的wait()方法进入等待状态，另一个线程B调用了对象o的notify()方法或notifyAll()方法，线程A收到通知后，从对象o的wait()方法返回。上述两个线程通过对象o完成交互 。

使用wait(), notify(), notifyAll()需要先对调用对象加锁

notify()或notifyAll()调用后，等待线程不会从wait()返回，需要该调用notify()的线程释放锁之后，等待线程才有机会从wait()返回；

notify()将等待队列中的一个线程移到同步队列中，而notifyAll()是将所有线程移到同步队列中，并设置状态从WAITING到BLOCKING;

从wait()方法返回的前提是获得了调用对象的锁；

等待/通知机制依托于同步机制，目的是确保等待线程从wait()方法返回时能够感知到通知线程对变量做出的修改。

1. 等待/通知经典范式

等待方（消费者）：

1. 获取对象锁；
2. 如果条件不满足，调用对象的wait()方法；
3. 条件满足继续执行；

通知方（生产者）：  
1.获取对象锁；

2. 改变条件；

3. 通知所有等待在该对象上的线程；

1. Thread.join()

如果一个线程A执行了thread.join()方法，意思是当前线程A需要等待thread线程终止之后才会从thread.join()返回。

1. ThreadLocal

线程变量，是一个以ThreadLocal对象为键，任意对象为值的存储结构，这个结构被附带在线程上，一个线程可以根据一个ThreadLocal对象查询到绑定在这个线程上的一个值。可以通过set()来设置，get()来获取。