如何编写一个线程类并启动？

**通过继承Thread类或实现Runnable接口，重写run方法；通过实现Callable接口，重写call方法。通过new线程并调用其start方法启动线程或通过线程池的execute、commit方法启动线程。**

直接调用线程的run方法会怎样？

**与普通的方法调用一样，并不会启动线程。**

线程池有几种类型？

**SingleThreadPool:单线程线程池**

**FixedThreadPoold:数量固定线程池**

**CachedThreadPool:数量不固定线程池**

**ScheduledThreadPool:定时线程池**

如何创建一个线程池？

**ExecutorService threadPool = Executors.newSingleThreadPool();**

**ExecutorService threadPool = new ThreadPoolExecutor(int corePoolSize, int maximumPoolSize, long keepAliveTime, TimeUnit unit, BlockingQueue<Runnable> workQueue， ThreadFactory factory，RejectEcecutionHandler handler);**

线程池参数？

**1.核心线程数量**

**2.最大线程数量**

**3.非核心线程存活时间**

**4. 非核心线程存活时间单位**

**5. 工作队列**

**6. 线程工厂**

**7. 任务拒绝方式**

新任务来了，线程池线程创建方式？

1. **未达到核心线程限制数量，创建核心线程。**
2. **达到核心线程数量，将新任务存储至工作队列。**
3. **工作队列满，未达到非核心线程限制，创建非核心线程。**
4. **达到非核心线程限制，使用任务拒绝方式。**

线程池工作队列类型？

1. **直接提交队列（**SynchronousQueue**），无容量，直接给非核心线程执行。**
2. **有界队列（**ArrayBlockingQueue**），可设置大小，满了会创建非核心线程。**
3. **无界队列（**LinkedBlockingQueue**），不会创建非核心线程。**
4. **优先任务队列（**PriorityBlockingQueue**），无界，不会创建非核心线程；线程可以实现Comparable的compareTo方法，进而设置优先级。**

线程池的拒绝策略？

**1、AbortPolicy策略：该策略会直接抛出异常，阻止系统正常工作；**

**2、CallerRunsPolicy策略：如果线程池的线程数量达到上限，该策略会把任务队列中的任务放在调用者线程当中运行；**

**3、DiscardOledestPolicy策略：该策略会丢弃任务队列中最老的一个任务，也就是当前任务队列中最先被添加进去的，马上要被执行的那个任务，并尝试再次提交；**

**4、DiscardPolicy策略：该策略会默默丢弃无法处理的任务，不予任何处理。当然使用此策略，业务场景中需允许任务的丢失；**

ThreadLocal的作用?

**ThreadLocal为每一个线程创建一个副本存入线程的ThreadLocalMap中，以ThreadLocal作为key，ThradLocal设置的值为value，实现数据隔离。**

Volatile修饰符的作用？

**Volatile修饰的变量被修改时会马上更新到主存并使其他线程立即可见。它只能保证可见性，有序性，不能保证原子性。**

sleep和wait的区别？

**sleep是Thread的静态方法，wait是Object的方法；sleep不释放锁使线程阻塞一定时间，wait释放锁不阻塞线程；wait只能在同步块或同步方法中使用。**

悲观锁和乐观锁的区别？

**悲观锁假设最坏的情况，每次拿数据的时候都认为别人会修改，所以每次拿数据时都将数据上锁，别人拿同样的数据时就会阻塞直到他也拿到锁。sychronized修饰词就是悲观锁的一种实现方式。**

**乐观锁每次拿数据都认为别人不会修改，所以不会上锁，只是在更新的时候判断此间别人是否修改了数据，若没有则提交更新。CAS是乐观锁的实现方式，java.util.concurrent.atomic包下的原子变量类就是通过CAS实现的。**

CAS是什么？

**Compare and set，cpu指令，当且仅当预期值A和内存值V相同时，将内存值V修改为B，否则什么都不做。**

什么是守护线程？

**守护线程指在后台运行为用户线程提供服务的线程，如垃圾回收线程。当所有非守护线程都退出时，JVM会杀死所有守护线程，程序退出。在守护线程中创建的线程也是守护线程。通过调用线程的setDaemon(true)将线程转化为守护线程。**

**锁的升级？**

**无锁**

**偏向锁：只有一个线程进入同步块（无同步操作）**

**轻量锁：多个线程错开进入同步块，未出现锁竞争（CAS）**

**重量锁：多个线程进入同步块，出现锁竞争（互斥同步）**

**死锁产生的条件？**

**互斥、请求和保持、不可剥夺、循环等待。**

Lock与synchronize?

**Synchronize为java关键字,可以修饰代码快、方法，不需要手动进行解锁，进入互斥代码后获取锁或被阻塞，离开互斥代码后自动释放锁。**

**Lock为JAVA提供的接口，实现类包括ReentrantLock（可重入锁）, ReentrantReadWriteLock(可重入读写锁)，需要手动解锁，但提供了更丰富的方法，入tryLock可以尝试获取锁，获取失败可以执行后续的操作，也可以主动释放锁。**

**Lock基于双向链表，通过CAS实现加锁，synchronize是jvm实现的。**

Lock公平锁和非公平锁的区别？

**Lock采用AQS(队列同步器)的方式构建锁，通过维护一个state锁状态及一个FIFO（先进先出）阻塞队列来实现锁的控制。**

**Lock默认使用非公平锁，线程通过CAS获取锁失败时，都有一次机会再尝试获取锁的机会，获取失败后才进入阻塞队列，因此可能造成线程插队的情况，但同时运行效率也比较高。**

**非公平锁情况下线程通过CAS获取锁失败后，就会直接进入阻塞队列，按照队列先后顺序获取锁，不会出现插队情况，但同时运行效率也较差。**

Future和FutureTask？

**Future可以获取线程状态、中断线程、阻塞获取callable返回信息。**

**Future除了实现Future接口，还是先了Runnable接口，具备Future功能之外还能作为线程运行。**