1.Java中多线程同步是什么？

在一个多线程的程序中，同步就像一个队列，各个线程依次访问程序。一个公共资源不同步的话，就可能出现一个线程访问，另一个线程同时在增加或修改，这样容易出现程序异常，也就是我们平常说的数据不安全。

比较常用的同步方法是使用sychroized。

2.实现多线程的几种方法？

（1）继承Thread类，重写run()

（2）实现Runnable接口，实现run()

（3）写匿名内部类：

Thread t1 = new Thread(){

public void run() {

System.out.println(Thread.currentThread().getName());

}

};

3.什么是ThreadLocal类，怎么使用它？

ThreadLocal 不是用来解决共享对象的多线程访问问题的，一般情况下，通过ThreadLocal.set() 到线程中的方法对象只能当前线程使用（可以说是只能在当前对象中使用），其他线程是不能使用的。保护变量不受其他线程影响。

4.线程与进程的关系？

一个进程可以包括多个线程，不同的进程使用不同的内存空间，而不同的线程使用同一个内存空间。所以不同的进程之间互不影响，而线程会被影响。

5.Thread类中start()与run()的区别？

两个方法都能运行线程中的方法，但是run()仅仅是在当前线程中运行。start()方法是开启一个新的线程，在新的线程里面运行run()。

6.volatile变量是什么？

volatile变量是一个特殊的修饰符，只有成员变量能使用它。被它修饰的变量能够保证一个线程修改的时候，其他访问的线程能够立刻得到最新的变量。保证下一个读取操作会在写操作完成之后发生。但是实际中也不能确保线程安全（这个涉及到jvm内存模型，这里不细追究）。

7.什么是线程安全？

多个线程操作执行同一段程序，线程之间互不影响，获得的结果符合预期，这就是线程安全。

8.什么是竞态条件？

多线程访问公共资源的时候就会出现竞态条件。例如最开始执行的程序竞争失败，就会排到其他线程后面执行，那么整个程序就会出现不确定的bug，这种bug很难发现且会重复出现。例如：定时任务执行时间长，可能循环的时候，突然就不执行了，也不出现异常。

9.停止一个线程

Java提供了stop(),suspend(),resume()等方法，不过现在都已经过时了。我们自己一般的做法就是在run方法中控制，让run方法执行完，即可停止线程。

10.一个线程运行时发生异常会怎样？

如果没有捕获异常，程序线程就停止向下执行了。这种异常的线程会被Thread.UncaughtException捕获。用法如下。

public class ExceptionHandler implements UncaughtExceptionHandler {

@Override

public void uncaughtException(Thread t, Throwable e) {

System.out.println("出现异常了");

}

public static void main(String[] args) {

Thread.currentThread().setUncaughtExceptionHandler(new ExceptionHandler());

int a = 1 / 2;

System.out.println(a);

}

}

11.线程之间实现数据共享（线程间通信）

（1）使用volatile修饰的变量。

（2）使用阻塞队列这样的并发结构，让多个线程访问资源

（3）使用数据安全的数据结构，如ConcurrentHashMap，ConcurrentLinkedDeque，Vector等。

12.Java的notify和notifyAll的区别？

字面意思是将处于等待状态的线程唤醒一个和全部唤醒。两者都是java调用系统底层函数库的方法。与wait()方法一样，

都必须写在同步代码块内，否则会出现IllegalMoniterStateException。

13.Java中的同步集合与并发集合有什么区别

都是为了处理多线程和并发问题，但是并发集合扩展性要好一些。

同步集合是比较早出现的，如Vector，多线程访问的时候，争夺访问资源，降低系统性能。

并发集合，如ConcurrentHashMap，提供了线程安全、锁分离、内部分区等，提高扩展性和性能。

14.堆和栈有什么不同？

堆保存对象数据，栈保存对象的引用（变量名，方法参数等）。

而每个线程都有自己的栈内存，一个线程自己的栈内存中的变量对其他栈是不可见的。

虽然对象创建在堆中，但是线程会从堆中缓存对象到栈里，如果多线程访问这个对象，就可能出现问题。

而volatile只允许线程冲堆中读取数据。

15.线程池？怎么用？

创建线程需要消耗系统资源和时间，每次都创建会降低系统的性能。

所有在程序启动的时候就预先创建好若干线程，这就是线程池。

比如1.5的新特性：Executor

public static void main(String[] args) {

// 开启一个有6个线程的线程池

ExecutorService pool = Executors.newFixedThreadPool(6);

//创建一个线程对象

Thread thread = new Thread(){

@Override

public void run() {

System.out.println(Thread.currentThread().getName());

}

};

try {

while(true){

//依次将线程对象放入线程池中执行

pool.execute(thread);

Thread.sleep(1000);

}

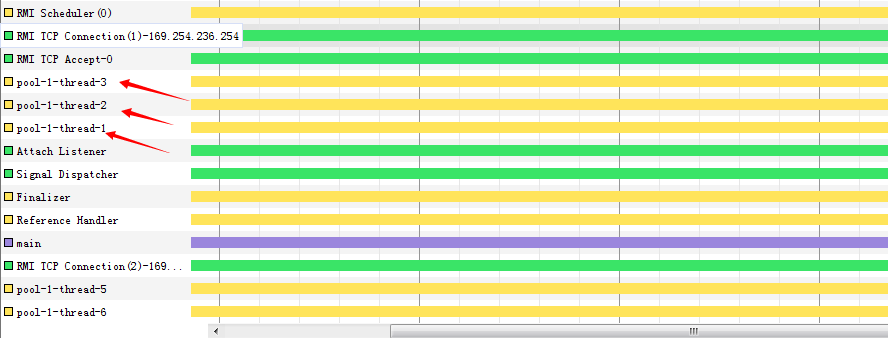
} catch (InterruptedException e) {

e.printStackTrace();

}

}

线程会依次创建6个，之后每次执行线程的方法的时候，出现的都是以下6个线程



16.什么是死锁？怎么造成死锁？怎么避免死锁？

当两个或以上的线程因访问多个资源出现的相互等待的现象，这会导致程序无法继续执行。

出现死锁必须满足以下四个因素：

(1)一个公共资源每次只能被一个线程访问；

(2)一个线程因为请求资源而阻塞，这个线程对已获得的资源保持不放；

(3)对已获得资源的线程，在资源没有使用完的时候，不能强行剥夺；

(4)多个线程之间形成一种头尾相接的循环等待的关系。

避免出现死锁的方法：

(1)使用ReentrantLock的tryLock方法设置超时时间，超时则退出防止死锁；

(2)使用java.util.concurrent下的并发类代替手动添加锁控制并发；

(3)不同多个方法使用同一个锁；

(4)减少使用同步代码块。

17.死锁、活锁、饥饿？

死锁是多个线程访问多个资源的时候，出现相互等待的现象。

活锁是线程不断的重复相同的操作，但每次都是失败的。虽然不会导致线程阻塞，但是程序无法进行下去。例如：一个线程用于从队列中读取数据，处理数据，但是队列为空，则一直在读取数据。

死锁和活锁的区别：死锁的程序是停止的，活锁的程序还在执行；活锁可能自行解开，死锁不能。

饥饿是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。

18.JVM中哪个参数是用来控制线程的栈堆栈小的

 -Xss参数用来控制线程的堆栈大小

19.有三个线程T1，T2，T3，怎么确保它们按顺序执行？

join()方法是将该线程加入到本线程中。也就是程序会依次进行。

T1.start();T1.join();T2.start();T2.join();T3.start();T3.join();

20.Thread类中的yield方法有什么作用？

使当前线程短暂放弃CPU的使用，让步给其他的线程。

Thread t2 = new Thread(){

@Override

public void run() {

Thread.yield();

}

};

21.设置优先级(并不是100%的有效，只是减低的本线程占用系统资源的机会)

Thread t = new Thread();//1.创建一个线程

t.setPriority(Thread.MAX\_PRIORITY);//最高优先级10

t.setPriority(Thread.MIN\_PRIORITY);//最低优先级1

t.setPriority(Thread.NORM\_PRIORITY);//普通优先级5

22.如果你提交任务时，线程池队列已满。会时发会生什么？

原先有人时候会异常，我看ThreadPoolExecutor类的execute()中，关于向线程池队列增加元素

采用的是offer方法，即不会阻塞线程，也不会抛出异常，而是返回添加结果false。不知道是不是

有的线程池队列使用的是add()、 remove()等方法，这样会抛出异常。

23.Java线程池中submit() 和 execute()方法有什么区别？

execute方法提交线程后没有返回值，且execute方法实现的是Executor接口。

submit方法踢脚线程后返回计算结果，且submit方法实现的是executorService接口。

24.怎样理解阻塞和非阻塞，同步和异步？参考<https://www.zhihu.com/question/19732473>

出场人物：老张，水壶两把（普通水壶，简称水壶；会响的水壶，简称响水壶）。

1 老张把水壶放到火上，立等水开。（同步阻塞）

老张觉得自己有点傻

2 老张把水壶放到火上，去客厅看电视，时不时去厨房看看水开没有。（同步非阻塞）

老张还是觉得自己有点傻，于是变高端了，买了把会响笛的那种水壶。水开之后，能大声发出嘀~~~~的噪音。

3 老张把响水壶放到火上，立等水开。（异步阻塞）

老张觉得这样傻等意义不大

4 老张把响水壶放到火上，去客厅看电视，水壶响之前不再去看它了，响了再去拿壶。（异步非阻塞）

老张觉得自己聪明了。

说明:阻塞就是干等着，非阻塞就是一边干别的一边等。

同步没有通知，异步通知。

实际开发:处理IO的时候，无论阻塞与否都是同步的。

使用银行之w类的API的时候会有异步的。

25.ReadWriteLock？

ReadWriteLock是代码级的锁技术。ReentrantReadWriteLock是它的一个实现类。

public static String get(String key){

lock.readLock().lock();// 1.1开启读锁

try{

return cache.get(key);

} finally{

lock.readLock().unlock();// 1.2关闭读锁

}

}

public static void put(String key, String value){

lock.writeLock().lock();// 1.1开启写锁

try{

cache.put(key, value);

} finally {

lock.writeLock().unlock();// 1.2关闭写锁

}

}

26.多线程的忙循环？

就是通过程序循环让线程等待，比如while循环，可能永远是在循环，这就导致了线程其实一直在运行，占用着CPU。这样做的好处是为了保留CPU的缓存，在多核的计算机中，一个线程唤醒后可能切换到其他的CPU上运行，这会导致CPU重建缓存。而忙循环避免重建缓存，免去了重建缓存等待的时间。

27.volatile 变量和 atomic 变量有什么不同？

volatile使修饰的对象，读写只操作在堆中，这样能保证读的操作会在写的操作之后执行。但是这样不能保证数据安全。

atomic 相关类的atomic方法，使操作具有原子性。

28. 如果同步块内的线程抛出异常会发生什么？

线程就会结束。如果用的是上锁的方式的话，需要使用finally执行释放锁的操作。同步代码块无论正常还是异常都

会自动释放锁。

30.如何在Java中创建线程安全的Singletn?

（1）静态代码块内创建。

（2）利用枚举创建实例。（其实也是预先创建好）

（3）使用同步代码块。

（4）使用 volatile修饰

31.写出3条你遵循的多线程最佳实践?

（1）创建线程的名称要和业务功能相关。

（2）锁和同步代码块的范围越小越好。

（3）减少使用wait，notify。

（4）所使用并发集合，少用同步集合。

32.fork join框架是什么？

是JDK1.7新出的一个将大任务分割成小任务计算，最终将小任务结果合并的框架。

能否更好的使用多核计算机的CPU，最大程度的利用了计算机的计算能力。

它的核心是“工作窃取算法”。那么为什么需要使用工作窃取算法呢？假如我们需要做一个

比较大 的任务，我们可以把这个任务分割为若干互不依赖的子任务，为了减少线程间的

竞争，于是把这些子任务分别放到不同的队列里，并为每个队列创建一个单独的线程来执行

队列里的任务，线程和队列一一对应，比如A线程负责处理A队列里的任务。但是有的线程会

先把自己队列里的任务干完，而其他线程对应的队列里还有任务等待处理。干完活的线程与

其等着，不如去帮其他线程干活，于是它就去其他线程的队列里窃取一个任务来执行。而在

这时它们会访问同一个队列，所以为了减少窃取任务线程和被窃取任务线程之间的竞争，通

常会使用双端队列，被窃取任务线程永远从双端队列的头部拿任务执行，而窃取任务的线程

永远从双端队列的尾部拿任务执行。

33.Java多线程中调用wait() 和 sleep()方法有什么不同？

(1)wait是object的方法，针对的是对象，范围更大，sleep是Thread的静态方法，用于与线程。

(2)sleep在等待的时候没有使用锁，而wait释放锁。

(3)sleep可以作用于任何地方，而wait只能使用在sychronzied代码块内。

34.简述synchronized和java.util.concurrent.locks.Lock的异同？

相同点：Lock能完成synchronized的所有功能。

不同点：

（1）Lock是代码级操作，程序的逻辑、性能、扩展都要好了不少。

（2）synchronized无论程序异常正常都会释放锁，而Lock必须手动释放锁。

（3）Lock可以使用tryLock，设置锁超时时间，避免死锁。

35.什么是上下文切换

多核操作系统，多线程任务时，线程之间争夺系统资源，任务频繁出现停止和恢复的线程。

比如，1000个线程并发运行，CPU想要运行这些线程，就必须在线程之间切换运行，可能

1s切换1000次，让任务在表面看上去都同时在运行。其实这样的效率已经低了，因为线程的

上下文切换回消耗时间。