彻底明白Java的多线程-线程间的通信 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/145/2021_2022__E5_BD_BB_E5_BA_95_E6_98_8E_E7_c104_145637.htm 一． 实现多线程1. 虚假的多线程例1：publicclassTestThread { inti=0, j=0. publicvoidgo(intflag){ while(true){ try{ java/lang/Thread.java.html" target="_blank">Thread.sleep(100). } catch(java/lang/InterruptedException.java.html" target="_blank">InterruptedExceptione){ java/lang/System.java.html" target="_blank">System.out.println("Interrupted"). } if(flag==0) i . java/lang/System.java.html" target="_blank">System.out.println("i=" i). } else{ j . java/lang/System.java.html" target="_blank">System.out.println("j=" j). } } } publicstaticvoidmain(java/lang/String.java.html" target="_blank">String[] args){ newTestThread().go(0). newTestThread().go(1). } } 上面程序的运行结果为：i=1i=2i=3。。。结果将一直打印出I的值。我们的意图是当在while循环中调用sleep()时，另一个线程就将起动，打印出j的值，但结果却并不是这样。关于sleep()为什么不会出现我们预想的结果，在下面将讲到。2. 实现多线程通过继承class Thread或实现Runnable接口，我们可以实现多线程2.1 通过继承class Thread实现多线程class Thread中有两个最重要的函数run()和start()。1) run()函数必须进行覆写，把要在多个线程中并行处理的代码放到这个函数中。2) 虽然run()函数实现了多个线

程的并行处理，但我们不能直接调用run()函数，而是通过调用start()函数来调用run()函数。在调用start()的时候，start()函数会首先进行与多线程相关的初始化（这也是为什么不能直接调用run()函数的原因），然后再调用run()函数。例2：publicclassTestThread extendsjava/lang/Thread.java.html" target="_blank">Thread{ privatestaticintthreadCount = 0. privateintthreadNum = threadCount. privateinti = 5. publicvoidrun(){ while(true){ try{ java/lang/Thread.java.html" target="_blank">Thread.sleep(100). } catch(java/lang/InterruptedException.java.html" target="_blank">InterruptedExceptione){ java/lang/System.java.html" target="_blank">System.out.println("Interrupted"). } java/lang/System.java.html" target="_blank">System.out.println("Thread " threadNum " = " i). if(--i==0) return. } } >publicstaticvoidmain(java/lang/String.java.html" target="_blank">String[] args){ for(inti=0. inewTestThread().start(). } } 运行结果为：Thread 1 = 5Thread 2 = 5Thread 3 = 5Thread 4 = 5Thread 5 = 5Thread 1 = 4Thread 2 = 4Thread 3 = 4Thread 4 = 4Thread 1 = 3Thread 2 = 3Thread 5 = 4Thread 3 = 3Thread 4 = 3Thread 1 = 2Thread 2 = 2Thread 5 = 3Thread 3 = 2Thread 4 = 2Thread 1 = 1Thread 2 = 1Thread 5 = 2Thread 3 = 1Thread 4 = 1Thread 5 = 1从结果可见，例2能实现多线程的并行处理。＊＊：在上面的例子中，我们只用new产生Thread对象，并没有用reference来记录所产生的Thread对象

。根据垃圾回收机制，当一个对象没有被reference引用时，它将被回收。但是垃圾回收机制对Thread对象“不成立”。因为每一个Thread都会进行注册动作，所以即使我们在产生Thread对象时没有指定一个reference指向这个对象，实际上也会在某个地方有个指向该对象的reference，所以垃圾回收器无法回收它们。3) 通过Thread的子类产生的线程对象是不同对象的线程classTestSynchronized extendsjava/lang/Thread.java.html" target="_blank">Thread{ publicTestSynchronized(java/lang/String.java.html" target="_blank">Stringname){ super(name). } publicsynchronizedstaticvoidprt(){ for(inti=10. ijava/lang/System.java.html" target="_blank">System.out.println(java/lang/Thread.java.html" target="_blank">Thread.currentThread().getName() " : " i). try{ java/lang/Thread.java.html" target="_blank">Thread.sleep(100). } catch(java/lang/InterruptedException.java.html" target="_blank">InterruptedExceptione){ java/lang/System.java.html" target="_blank">System.out.println("Interrupted"). } } } publicsynchronizedvoidrun(){ for(inti=0. ijava/lang/System.java.html" target="_blank">System.out.println(java/lang/Thread.java.html" target="_blank">Thread.currentThread().getName() " : " i). try{ java/lang/Thread.java.html" target="_blank">Thread.sleep(100). } catch(java/lang/InterruptedException.java.html" target="_blank">InterruptedExceptione){

java/lang/System.java.html"
target="_blank">System.out.println("Interrupted"). } } } }
publicclassTestThread{
publicstaticvoidmain(java/lang/String.java.html"
target="_blank">String[] args){ TestSynchronized t1 =
newTestSynchronized("t1"). TestSynchronized t2 =
newTestSynchronized("t2"). t1.start(). t1.start(). //（1）//t2.start().
（2）} } 运行结果为：t1 : 0t1 : 1t1 : 2t1 : 0t1 : 1t1 : 2由于是同一个对象启动的不同线程，所以run()函数实现了synchronized。如果去掉（2）的注释，把代码（1）注释掉，结果将变为：t1 : 0t2 : 0t1 : 1t2 : 1t1 : 2t2 : 2由于t1和t2是两个对象，所以它们所启动的线程可同时访问run()函数。2.2 通过实现Runnable接口实现多线程如果有一个类，它已继承了某个类，又想实现多线程，那就可以通过实现Runnable接口来实现。1) Runnable接口只有一个run()函数。2) 把一个实现了Runnable接口的对象作为参数产生一个Thread对象，再调用Thread对象的start()函数就可执行并行操作。如果在产生一个Thread对象时以一个Runnable接口的实现类的对象作为参数，那么在调用start()函数时，start()会调用Runnable接口的实现类中的run()函数。例3.1：publicclassTestThread
implementsjava/lang/Runnable.java.html"
target="_blank">Runnable{ privatestaticintthreadCount = 0.
privateintthreadNum = threadCount. privateinti = 5.
publicvoidrun(){ while(true){ try{ java/lang/Thread.java.html"
target="_blank">Thread.sleep(100). }
catch(java/lang/InterruptedException.java.html"

target="_blank">InterruptedExceptione){
java/lang/System.java.html"
target="_blank">System.out.println("Interrupted"). }
java/lang/System.java.html"
target="_blank">System.out.println("Thread " threadNum " = " i).
if(--i==0) return. } }
publicstaticvoidmain(java/lang/String.java.html"
target="_blank">String[] args){ for(inti=0.
inewjava/lang/Thread.java.html"
target="_blank">Thread(newTestThread()).start(). //（1）} }运行结果为：Thread 1 = 5Thread 2 = 5Thread 3 = 5Thread 4 = 5Thread 5 = 5Thread 1 = 4Thread 2 = 4Thread 3 = 4Thread 4 = 4Thread 4 = 3Thread 5 = 4Thread 1 = 3Thread 2 = 3Thread 3 = 3Thread 4 = 2Thread 5 = 3Thread 1 = 2Thread 2 = 2Thread 3 = 2Thread 4 = 1Thread 5 = 2Thread 1 = 1Thread 2 = 1Thread 3 = 1Thread 5 = 1例3是对例2的修改，它通过实现Runnable接口来实现并行处理。代码（1）处可见，要调用TestThread中的并行操作部分，要把一个TestThread对象作为参数来产生Thread对象，再调用Thread对象的start()函数。3) 同一个实现了Runnable接口的对象作为参数产生的所有Thread对象是同一对象下的线程。例3.2：packagemypackage1.
publicclassTestThread implementsjava/lang/Runnable.java.html"
target="_blank">Runnable{ publicsynchronizedvoidrun(){
for(inti=0. ijava/lang/System.java.html"
target="_blank">System.out.println(java/lang/Thread.java.html"
target="_blank">Thread.currentThread().getName() " : " i). try{

java/lang/Thread.java.html" target="_blank">Thread.sleep(100). } catch(java/lang/InterruptedException.java.html" target="_blank">InterruptedException e){ java/lang/System.java.html" target="_blank">System.out.println("Interrupted"). } } } publicstaticvoidmain(java/lang/String.java.html" target="_blank">String[] args){ TestThread testThread = newTestThread(). for(inti=0. i//new Thread(testThread, "t" i).start(). （1）newjava/lang/Thread.java.html" target="_blank">Thread(newTestThread(), "t" i).start(). （2） } } 运行结果为：t0 : 0t1 : 0t2 : 0t3 : 0t4 : 0t0 : 1t1 : 1t2 : 1t3 : 1t4 : 1t0 : 2t1 : 2t2 : 2t3 : 2t4 : 2t0 : 3t1 : 3t2 : 3t3 : 3t4 : 3t0 : 4t1 : 4t2 : 4t3 : 4t4 : 4由于代码（2）每次都是用一个新的TestThread对象来产生Thread对象的，所以产生出来的Thread对象是不同对象的线程，所以所有Thread对象都可同时访问run()函数。如果注释掉代码（2），并去掉代码（1）的注释，结果为：t0 : 0t0 : 1t0 : 2t0 : 3t0 : 4t1 : 0t1 : 1t1 : 2t1 : 3t1 : 4t2 : 0t2 : 1t2 : 2t2 : 3t2 : 4t3 : 0t3 : 1t3 : 2t3 : 3t3 : 4t4 : 0t4 : 1t4 : 2t4 : 3t4 : 4由于代码（1）中每次都是用同一个TestThread对象来产生Thread对象的，所以产生出来的Thread对象是同一个对象的线程，所以实现run()函数的同步。 