线程中断机制（interrupt）

优雅地中断线程，是一门艺术

众所周知，Thread.stop、Thread.suspend、Thread.resume都已经被废弃了。因为它们太暴力了，是不安全的，这种暴力中断线程是一种不安全的操作。举个例子来说明其可能造成的问题：

public class ThreadTest {

public static void main(String[] args) throws InterruptedException {

StopThread thread = new StopThread();

thread.start();

// 休眠1秒，确保线程进入运行

Thread.sleep(1000);

// 暂停线程

thread.stop();

// thread.interrupt();

// 确保线程已经销毁

while (thread.isAlive()) { }

// 输出结果

thread.print();

}

private static class StopThread extends Thread {

private int x = 0; private int y = 0;

@Override

public void run() {

// 这是一个同步原子操作

synchronized (this) {

++x;

try {

// 休眠3秒,模拟耗时操作

Thread.sleep(3000);

} catch (InterruptedException e) {

e.printStackTrace();

}

++y;

}

}

public void print() {

System.out.println("x=" + x + " y=" + y);

}

}

}

上述代码中，run方法里是一个同步的原子操作，x和y必须要共同增加，然而这里如果调用thread.stop()方法强制中断线程，输出如下：

x=1 y=0

没有异常，也破坏了我们的预期。如果这种问题出现在我们的程序中，会引发难以预期的异常。因此这种不安全的方式很早就被废弃了，取而代之的是interrupt()。上述代码如果采用thread.interrupt()方法，输出结果如下：

x=1 y=1

java.lang.InterruptedException: sleep interrupted

at java.lang.Thread.sleep(Native Method)

at ThreadTest$StopThread.run(ThreadTest.java:28)

x=1,y=1这个结果是符合我们的预期的，同时还抛出了个异常，这个异常下文详说。