1. **优雅的停止线程**

在多线程的操作之中，如果要启动多线程，我们肯定使用的时Thread类之中的start方法，而如果对于多线程我们需要停止多线程，原本我们Thread类之中提供有stop()方法，但是方法在1.2版本之后便不再使用了。而除了stop方法之后我们还有一些方法也被禁用了。

停止多线程: public void stop() ;

销毁多线程: public void destory () ;

挂起线程: public final void suspend() ;

恢复挂起线程: public final void resume() ;

之所以废除这些方法，是因为这些方法容易造成线程的死锁。所以JDK1.2之后不建议使用这些方法。

而如果我们想要停止线程我们需要使用一种柔和的方法

范例:

|  |
| --- |
| package com.company;  /\*\*  \* @author Jackwen  \*/  public class StopThread {  public static boolean flag = true ;  public static void main(String[] args) throws InterruptedException {  new Thread(()->{  long num = 0 ;  while(flag) {  try {  Thread.sleep(50);  } catch (InterruptedException e) {  e.printStackTrace();  }  System.out.println("正在运行，num = " + num++);  }  },"执行线程").start() ;  //程序运行200毫秒  Thread.sleep(200);  flag = false ;  }  } |

万一现在有其它线程去控制这个flag的内容，那么对于这个线程来说不是立刻停止，而是在下一次检测flag的值的时候停止。

1. **后台守护线程**

假设有一个人并且它有个保镖，那么只有在这个人活着的时候，这个保镖才会守护她。守护线程的意思就是在整体线程还运行的情况下，她才会运行，而一旦程序停止了，守护线程也会停止。

在Thread类里面提供有如下的守护线程的操作方法:

·设置为守护线程：public final void setDaemon(boolean on) ;

·判断是否为守护线程：public final boolean isDaemon();

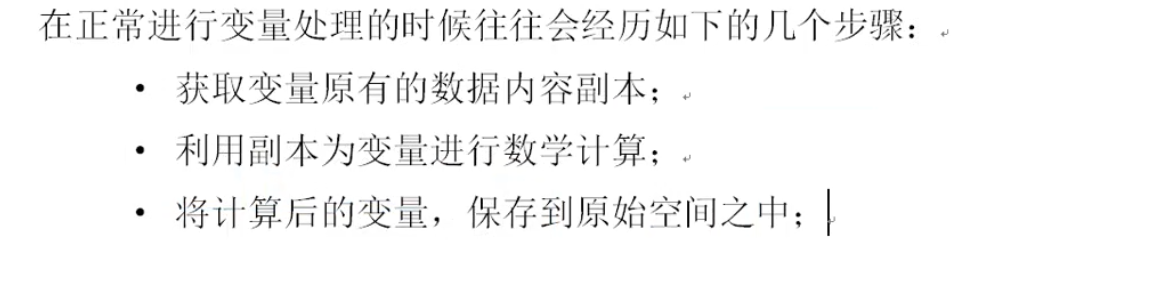
范例：

|  |
| --- |
| package com.company;  public class Daemon {  public static boolean flag = true ;  public static void main(String[] args) throws InterruptedException {  Thread userThread = new Thread(()->{  for (int i = 0; i < 10; i++) {  try {  Thread.sleep(50);  } catch (InterruptedException e) {  e.printStackTrace();  }  System.out.println(Thread.currentThread().getName() + "正在运行，num = " + i);  }  },"用户线程");  Thread daemonThread = new Thread(()->{  for (int x = 0; x < Integer.MAX\_VALUE; x++) {  try {  Thread.sleep(50);  } catch (InterruptedException e) {  e.printStackTrace();  }  System.out.println(Thread.currentThread().getName() + "正在运行，num = " + x);  }  },"守护线程");  //设置为守护线程  daemonThread.setDaemon(true);  userThread.start();  daemonThread.start();  //程序运行200毫秒  Thread.sleep(200);  flag = false ;  }  } |

守护线程，都是围绕在用户线程的周围，如果用户程序结束那么守护线程也会结束。而JVM里面最大的线程就是GC线程。程序执行中GC线程会一直纯在而如果程序执行完毕则GC线程也将消失。

1. **Volatile关键字**

在多线程的定义之中，volatile关键字主要用于是在属性上的定义，表示此属性为直接数据操作，而不进行我们副本的拷贝处理。但是volatile不是同步属性！



而如果一个属性上追加了volatile关键字，表示不适用副本而直接使用原始数据。

