停止一个线程意味着在任务处理完任务之前停掉正在做的操作，也就是放弃当前的操作。

在java中有以下3种方法可以终止正在运行的线程：

1. 使用退出标志，使线程正常退出，也就是当run方法完成后线程终止。
2. 使用Thread.stop()方法强行终止，但是不推荐这个方法，因为stop和suspend及resume一样都是过期作废的方法。
3. 使用interrupt方法中断线程。

### 1、停不了的线程

Thread.interrupt()方法的使用效果并不像for + break语句那样，马上就停止循环。中断此线程（此线程不一定是当前线程，而是指调用该方法的Thread实例所代表的线程），但实际上只是给线程设置一个中断标志，线程仍会继续运行。

### 2、判断线程是否停止状态

java.lang.Thread类中提供了两种方法：

* interrupted()：测试当前线程是否已被中断；
* this.isInterrupted()：测试此线程是否已被中断。

区别：

public class Run {

public static void main(String[] args) {  
 MyThread thread = new MyThread();  
 thread.start();  
 try {  
 Thread.*sleep*(2000);  
 // 实际上只是给线程设置一个中断标志，线程仍会继续运行  
 thread.interrupt();  
 System.*out*.println("Stop 1??" + Thread.*interrupted*());  
 System.*out*.println("Stop 2??" + Thread.*interrupted*());  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
}

interrupted()测试当前线程是否已经中断，这个当前线程是main，从未中断过。

public class Run2 {

public static void main(String[] args) {  
 Thread.*currentThread*().interrupt();  
 System.*out*.println("Stop 1??" + Thread.*interrupted*());  
 System.*out*.println("Stop 2??" + Thread.*interrupted*());  
  
 System.*out*.println("End");  
 }  
}

方法interrupted()的确判断出当前线程是否是停止状态。但为什么第2个布尔值是false呢？官方帮助文档中对interrupted方法的解释：

测试当前线程是否已经中断。线程的中断状态由该方法清除。换句话说，如果

连续两次调用该方法，则第二次调用返回false。

isInterrupted()并未清除状态。

### 3、能停止的线程

可以在线程中用for语句判断一下是否是停止状态，如果是停止状态，则后面的代码不再运行即可。

### 4、在沉睡中停止

如果在sleep状态下停止某一线程，会进入catch语句，并且清除停止状态值，使之变为false。

### 5、能停止的线程——暴力停止

使用stop()方法停止线程。

### 6、方法stop()与java.lang.ThreadDeath异常

调用stop()方法时会抛出java.lang.ThreadDeath异常，但通常情况下，此异常不需要显示地铺捉。

stop()方法以及作废，因为如果强制让线程停止有可能使一些清理性的工作得不到完成。另外一个情况就是对锁定的对象进行了解锁，导致数据得不到同步的处理，出现数据不一致的问题。

### 7、释放锁的不良后果

使用stop()释放锁将会给数据造成不一致性的结果。如果出现这样的情况，程序处理的数据就有可能遭到破坏，最终导致程序执行的流程错误，一定要特别注意。不建议使用

### 8、使用return停止线程

将方法interrupt()与return结合使用也能实现停止线程的效果在。

不过还是建议使用“抛异常”的方法来实现线程的停止，因为在catch块中还可以将异常向上抛，使线程停止事件得以传播。