**Thread.interrupt()**

A forma recomendada de interromper threads é com o método Thread.interrupt(). Este método não interrompe a thread de fato, apenas seta uma flag nela (o status de interrupção). Você pode saber se a thread atual foi interrompida pelo uso dos métodos Thread.interrupted() e Thread.isInterrupted(). Estes métodos retornam um boolean dizendo se a thread foi interrompida ou não. A diferença entre eles é que o Thread.interrupted() limpa o status de interrupção (o define como false), enquanto que o Thread.isInterrupted() não o altera. Além disso, o Thread.interrupted() é um método estático que verifica e limpa o estado de interrupção da thread que o invocou (Thread.currentThread()), enquanto que o Thread.isInterrupted() é um método de instância da classe Thread.

Vários métodos da JDK (ex: Thread.sleep e Object.wait) lançam a exceção InterruptedException. Esta exceção é lançada quando a thread que está executando estes métodos (Thread.sleep, Object.wait ou outros) é interrompida pelo uso do Thread.interrupt(). Desta forma, é possível interromper uma thread que esteja presa em um Thread.sleep ou Object.wait. É importante que a thread interrompida trate a InterruptedException de forma satisfatória (simplesmente capturá-la e ignorá-la é uma má prática de programação). Quando a InterruptedException é lançada, o status de interrupção da thread é limpo (setada para false), uma vez que quando a exceção estiver sendo capturada, a interrupção já estará sendo tratada.

Se você for projetar um método que pode ficar preso por muito tempo, uma boa alternativa é verificar o Thread.isInterrupted() ou o Thread.interrupted() e lançar o InterruptedException. Por exemplo:

public void metodoDemorado() throws InterruptedException {

while (/\*alguma coisa\*/) {

// faz qualquer coisa

if (Thread.interrupted()) throw new InterruptedException();

}

}

A maneira apropriada para parar uma thread rodando é a de ajustar uma variável que a thread fique verificando temporariamente. Quando a thread detecta que a variável está modificada, a thread deve parar sua execução.

*// Cria e inicia a thread*  
    *MyThread* thread = new *MyThread*();  
      
 thread.start();  
  
*// processamento...*  
*// Pára a thread*  
    thread.allDone = true;  
      
    class *MyThread* extends Thread {  
          
 boolean allDone = false;  
      
*// Este método é chamado quando thread é iniciada*  
          
 public void run() {  
              
 while (true) {  
  
*// processamento...*  
                  
 if (allDone) {  
                    return;  
                }  
  
*// processamento...*  
            }  
        }  
    }

Leia mais em: [Saiba como parar uma tread](http://www.devmedia.com.br/saiba-como-parar-uma-tread/1332#ixzz44QGRfreP) <http://www.devmedia.com.br/saiba-como-parar-uma-tread/1332#ixzz44QGRfreP>

**Como interromper a execução de uma thread**  
  
A classe Thread fornece um método chamado **interrupt()** que serve para interrompermos a execução de uma determinada thread. Os métodos **sleep()** e **yield()** transferem a thread sendo executada no momento, do estado "executando" para o estado "pronta". O método **interrupt()**, por sua vez, faz com que uma thread vá para o estado "finalizada", ou seja, ela não voltará mais a ser executada.  
  
**O problema é que, nem sempre, o método interrupt() parece surtir o efeito desejado, ou seja, fazer com que a thread pare de executar imediatamente.** Por esta razão, **a forma mais fácil de "matar" uma thread é forçar a saída de seu método run()**. Veja um exemplo:

// criamos uma classe que servirá como thread

class MinhaThread extends Thread{

private String nome;

public MinhaThread(String nome){

this.nome = nome;

}

public void run(){

for(int i = 1; i <= 20; i++){

System.out.println(nome + ": " + i);

if((i >= 3) && (nome.equals("Thread 1"))){

try{

System.out.println(nome + " foi eliminada");

return; // interrompe a execução do método run()

}

catch(SecurityException e){

System.out.println(e.getMessage());

}

}

}

}

}

public class Estudos{

public static void main(String[] args){

// vamos criar duas threads

MinhaThread t1 = new MinhaThread("Thread 1");

t1.start();

MinhaThread t2 = new MinhaThread("Thread 2");

t2.start();

}

}

Ao usar a instrução **return** nós forçamos a saída do método **run()** e consequentemente, o fim da tarefa sendo realizada pela thread.