## 线程状态：

1.就绪状态 （当前进程准备好了可以执行，由cpu决定执行）

2.运行状态 （得到cpu时间片，在运行状态）

3.阻塞状态 （控制台输入信息的时候）

## Java线程两种实现方法：

1. 继承thread类
2. 实现runnable接口

## Join的使用

等另外线程运行完毕才运行自己的，（使用场景：当依赖于其他线程运行的结果）

public class TestJoin **{**

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

MyThread3 myThread3 **=** **new** MyThread3**();**

Thread t1 **=** **new** Thread**(**myThread3**);**

t1**.**start**();**

//主线程运行,如果等于5就让t1线程运行完毕，然后在运行自己的

**for** **(**int i **=** 0**;** i **<**10**;** i**++)** **{**

System**.**out**.**println**(**"main---"**+**i**);**

**if(**i**==**5**)** **{**

t1**.**join**();**

Thread**.**sleep**(**1000**);**

**}**

**}**

**}**

**}**

class MyThread3 **implements** Runnable**{**

public void run**()** **{**

**for** **(**int i **=** 0**;** i **<**10**;** i**++)** **{**

System**.**out**.**println**(**Thread**.**currentThread**().**getName**()+**"---"**+**i**);**

**try** **{**

Thread**.**sleep**(**1000**);**

**}** **catch** **(**InterruptedException e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

**}**

## interrupt

java interrupt()方法只是设置线程的中断标记，当对处于阻塞状态的线程调用interrupt方法时（处于阻塞状态的线程是调用sleep, wait, join 的线程)，会抛出InterruptException异常，而这个异常会清除中断标记。

public class TestInterrupt **{**

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

MyThread3 myThread3 **=** **new** MyThread3**();**

Thread t1 **=** **new** Thread**(**myThread3**);**

t1**.**start**();**

// 主线程运行,如果等于5就让t1线程运行完毕，然后在运行自己的

**for** **(**int i **=** 0**;** i **<** 10**;** i**++)** **{**

System**.**out**.**println**(**"main---" **+** i**);**

**if** **(**i **==** 5**)** **{**

t1**.**interrupt**();**// 设置了一个中断标记 状态为true,没有真正中断

**}**

Thread**.**sleep**(**1000**);**

**}**

**}**

**}**

class MyThread3 **implements** Runnable **{**

int i **=** 0**;**

public void run**()** **{**

**while** **(!**Thread**.**interrupted**())** **{**

System**.**out**.**println**(**Thread**.**currentThread**().**getName**()** **+** "---" **+** i**);**

i**++;**

**try** **{**

Thread**.**sleep**(**1000**);**//消除中断标记

**}** **catch** **(**InterruptedException e**)** **{**

//抛出InterruptException异常，而这个异常会清除中断标记。

e**.**printStackTrace**();**

Thread**.**currentThread**().**interrupt**();**//再次标记

**}**

**}**

**}**

**}**

## 守护线程

public class TestInterrupt **{**

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

MyThread4 myThread4 **=** **new** MyThread4**();**

Thread t1 **=** **new** Thread**(**myThread4**);**

t1**.**setDaemon**(true);**//设置守护线程，没有用户线程守护线程会自动终止

t1**.**start**();**

//用户线程，当用户线程结束，守护线程也会结束

**for** **(**int i **=** 0**;** i **<** 10**;** i**++)** **{**

**if(**i **==**5**)** **{**

Thread**.**currentThread**().**yield**();**//让出当前cpu执行

**}**

System**.**out**.**println**(**"main---" **+** i**);**

Thread**.**sleep**(**1000**);**

**}**

**}**

**}**

class MyThread4 **implements** Runnable **{**

public void run**()** **{**

**for** **(**int i **=** 0**;** i **<** 20**;** i**++)** **{**

System**.**out**.**println**(**Thread**.**currentThread**().**getName**()** **+** "---" **+** i**);**

i**++;**

**try** **{**

Thread**.**sleep**(**1000**);**

**}** **catch** **(**InterruptedException e**)** **{**

// TODO Auto-generated catch block

e**.**printStackTrace**();**

**}**

**}**

**}**

**}**