# Thread类中的方法：join()、sleep()、yield()之间的区别

2018-07-20 11:47:28 [SCU阳光](https://me.csdn.net/xzp_12345" \t "https://blog.csdn.net/xzp_12345/article/details/_blank) 阅读数 6393更多

分类专栏： [Java面试总结](https://blog.csdn.net/xzp_12345/article/category/7252202" \t "https://blog.csdn.net/xzp_12345/article/details/_blank) [Java并发编程](https://blog.csdn.net/xzp_12345/article/category/7819525" \t "https://blog.csdn.net/xzp_12345/article/details/_blank)

版权声明：本文为博主原创文章，遵循[CC 4.0 BY-SA](http://creativecommons.org/licenses/by-sa/4.0/" \t "https://blog.csdn.net/xzp_12345/article/details/_blank)版权协议，转载请附上原文出处链接和本声明。

本文链接：<https://blog.csdn.net/xzp_12345/article/details/81129735>

****目录****

[1. Thread类中的常用方法简介](https://blog.csdn.net/xzp_12345/article/details/81129735" \l "1. Thread%E7%B1%BB%E4%B8%AD%E7%9A%84%E5%B8%B8%E7%94%A8%E6%96%B9%E6%B3%95%E7%AE%80%E4%BB%8B" \t "https://blog.csdn.net/xzp_12345/article/details/_self)

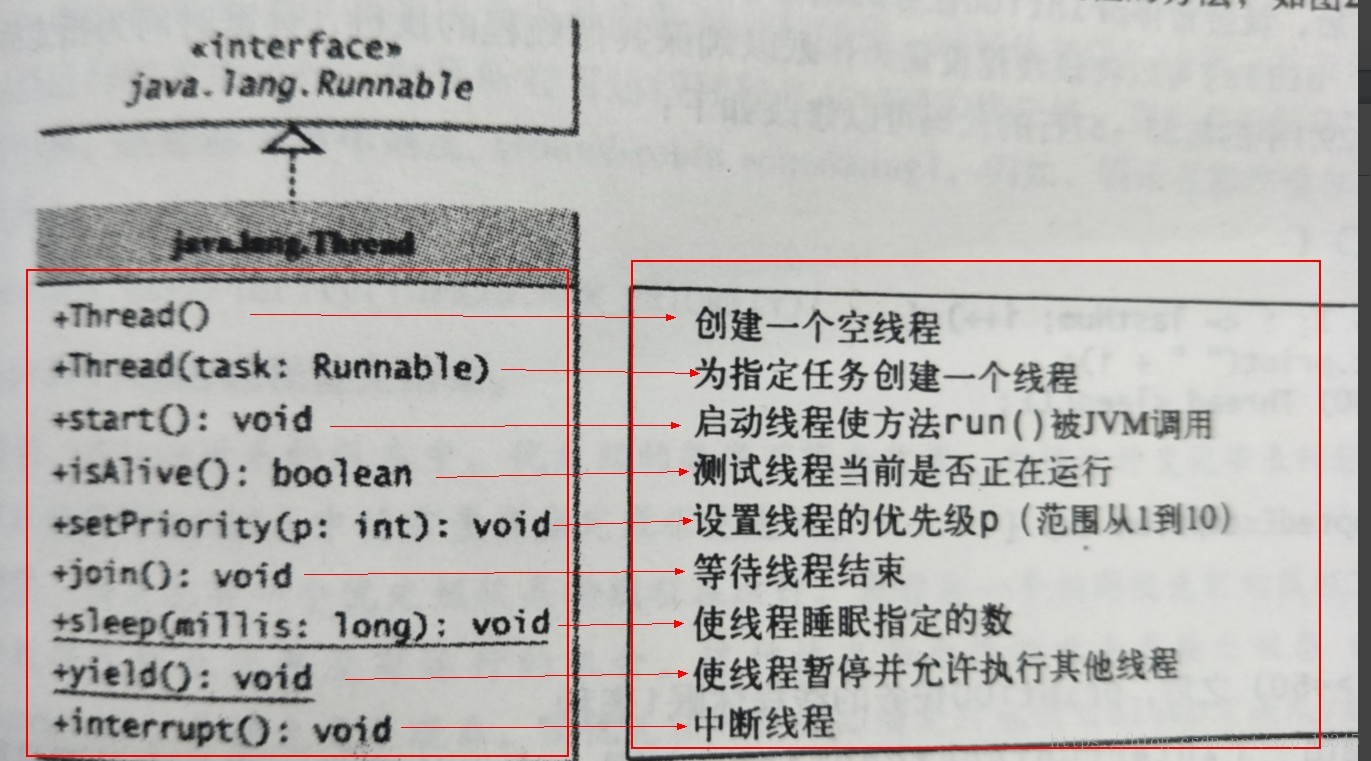
[2. Join()方法](https://blog.csdn.net/xzp_12345/article/details/81129735" \l "2. Join()%E6%96%B9%E6%B3%95" \t "https://blog.csdn.net/xzp_12345/article/details/_self) ：会释放锁

[3. Sleep() 方法：不会释放锁](https://blog.csdn.net/xzp_12345/article/details/81129735" \l "3. Sleep() %E6%96%B9%E6%B3%95%EF%BC%9A%E4%B8%8D%E4%BC%9A%E9%87%8A%E6%94%BE%E9%94%81" \t "https://blog.csdn.net/xzp_12345/article/details/_self)

[4. Yield()方法：线程让步  不会释放锁](https://blog.csdn.net/xzp_12345/article/details/81129735" \l "4. Yield()%E6%96%B9%E6%B3%95%EF%BC%9A%E7%BA%BF%E7%A8%8B%E8%AE%A9%E6%AD%A5%C2%A0 %E4%B8%8D%E4%BC%9A%E9%87%8A%E6%94%BE%E9%94%81" \t "https://blog.csdn.net/xzp_12345/article/details/_self)

# **1. Thread类中的常用方法简介**

首先，先看一下Thread类中的常见方法：



最主要的是区分其中join()、sleep()、yield()的用法区别

# **Join()方法**

看过Join源码的都知道，其中的核心就是

if (millis == 0) { *//由于上一步传入参数为0，因此调用当前判断*

while (isAlive()) { *//判断子线程是否存活*

wait(0); *//调用wait(0)方法*

}

}

* 1
* 2
* 3
* 4
* 5

即，只要子线程还活着，就一直wait，阻塞当前主线程，直到被唤醒。  
即join底层还是wait()，众所周知wait方法会释放锁，所以推测join也会释放锁，不过网上有很多join不释放的说法。

object.wait()和thread.join()  
join()属于Thread类中，thread的对象锁，因为thread.join()这个join里面是this这个锁，也就是thread，即在主线程中调用t.join()相当于t.wait()，我们去掉join这个方法，就相当于

main(){

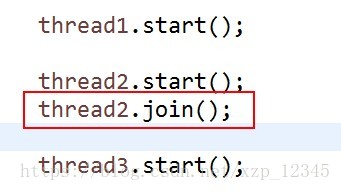
t.wait();}

Thread的非静态方法join()让一个线程等待另外一个线程完成才继续执行。如果线程A执行体中调用B线程的join()方法，则A线程将会被阻塞，直到B线程执行完为止，A才能得以****继续执行****。

如下图：



而且一定要理解“继续执行”的含义。比如有下面的三个线程：



其中thread2线程执行了join()方法。也就是必须等thread2执行结束之后才能继续往下执行thread3。但是，在执行thread2的过程中可以执行thread1(往前执行线程)。****这就是“继续执行”的含义****。执行结果可以看一下：

IMG_259

# **3. Sleep() 方法：不会释放锁**

Sleep——让当前正在执行的线程先暂停一定的时间，并进入阻塞状态。在其睡眠的时间段内，该线程由于不是处于就绪状态，因此不会得到执行的机会。即使此时系统中没有任何其他可执行的线程，处于sleep()中的线程也不会执行。因此sleep()方法常用来暂停线程的执行。当sleep()结束后，然后转入到 Runnable(就绪状态)，这样才能够得到执行的机会。



# **4. Yield()方法：线程让步  不会释放锁**

在第一节中已经介绍了，让一个线程执行了yield()方法后，就会进入Runnable(就绪状态)，【不同于sleep()和join（）方法，因为这两个方法是使线程进入阻塞状态】。除此之外，yield()方法还与线程优先级有关，当某个线程调用yield()方法时，就会从运行状态转换到就绪状态后，CPU从就绪状态线程队列中只会选择与该线程优先级相同或者更高优先级的线程去执行。