学习时正好碰到这两个方法，就查阅相关资料，并通过程序实现，进行区别一下：

**1、**每个对象都有一个锁来控制同步访问，Synchronized关键字可以和对象的锁交互，来实现同步方法或同步块。**sleep()方法**正在执行的线程主动让出CPU（然后CPU就可以去执行其他任务），在sleep指定时间后CPU再回到该线程继续往下执行(注意：sleep方法只让出了CPU，而并不会释放同步资源锁！！！即需要用到此对象的线程也因为无法取得该对象的锁从而无法继续执行，只可能进入锁池了，不用到该对象锁的线程可以因为sleep而获得CPU的使用时间了)；**wait()方法**则是指当前线程让自己**暂时退让出同步资源锁**，以便其他正在等待该资源的线程得到该资源进而运行**（进入等待队列**），只有调用了notify()方法，之前调用wait()的线程才会解除wait状态，可以去参与竞争同步资源锁，进而得到执行。（注意：notify的作用相当于叫醒睡着的人，而并不会给他分配任务，就是说notify只是让之前调用wait的线程有权利重新参与线程的调度）；

**2、sleep()方法可以在任何地方使用；wait()方法则只能在同步方法或同步块中使用（wait与notify是java同步机制中重要的组成部分。结合与synchronized关键字使用，可以建立很多优秀的同步模型。）；**

**3、**sleep()是线程线程类（Thread）的方法，调用会暂停此线程指定的时间，但监控依然保持，不会释放对象锁，到时间自动恢复；wait()是Object的方法，调用会放弃对象锁，进入等待队列，待调用notify()/notifyAll()唤醒指定的线程或者所有线程，才会进入锁池，不再次获得对象锁才会进入运行状态；

以程序说明：

**[java]** [view plain](https://blog.csdn.net/u012050154/article/details/50903326) [copy](https://blog.csdn.net/u012050154/article/details/50903326)

1. **public** **class** MultiThread {
3. **private** **static** **class** Thread1 **implements** Runnable{
4. @Override
5. **public** **void** run() {
6. //由于 Thread1和下面Thread2内部run方法要用同一对象作为监视器，如果用this则Thread1和Threa2的this不是同一对象
7. //所以用MultiThread.class这个字节码对象，当前虚拟机里引用这个变量时指向的都是同一个对象
8. **synchronized**(MultiThread.**class**){
9. System.out.println("enter thread1 ...");
10. System.out.println("thread1 is waiting");
12. **try**{
13. //释放锁有两种方式：(1)程序自然离开监视器的范围，即离开synchronized关键字管辖的代码范围
14. //(2)在synchronized关键字管辖的代码内部调用监视器对象的wait()方法。这里使用wait方法
15. MultiThread.**class**.wait();
16. }**catch**(InterruptedException e){
17. e.printStackTrace();
18. }
20. System.out.println("thread1 is going on ...");
21. System.out.println("thread1 is being over!");
22. }
23. }
25. }
27. **private** **static** **class** Thread2 **implements** Runnable{
28. @Override
29. **public** **void** run() {
30. //notify方法并不释放锁，即使thread2调用了下面的sleep方法休息10ms，但thread1仍然不会执行
31. //因为thread2没有释放锁，所以Thread1得不到锁而无法执行
32. **synchronized**(MultiThread.**class**){
33. System.out.println("enter thread2 ...");
34. System.out.println("thread2 notify other thread can release wait status ...");
35. MultiThread.**class**.notify();
36. System.out.println("thread2 is sleeping ten millisecond ...");
38. **try**{
39. Thread.sleep(10);
40. }**catch**(InterruptedException e){
41. e.printStackTrace();
42. }
44. System.out.println("thread2 is going on ...");
45. System.out.println("thread2 is being over!");
46. }
47. }
48. }
50. **public** **static** **void** main(String[] args) {
51. **new** Thread(**new** Thread1()).start();
52. **try**{
53. Thread.sleep(10);
54. }**catch**(InterruptedException e){
55. e.printStackTrace();
56. }
58. **new** Thread(**new** Thread2()).start();
59. }
61. }

程序运行结果如下图所示

