1. sleep()使当前线程进入停滞状态（阻塞当前线程），让出CUP的使用、目的是不让当前线程独自霸占该进程所获的CPU资源，以留一定时间给其他线程执行的机会。sleep()是Thread类的Static(静态)的方法；因此他不能改变对象的机锁，所以当在一个Synchronized块中调用Sleep()方法是，线程虽然休眠了，但是对象的机锁并没有被释放，其他线程无法访问这个对象（即使睡着也持有对象锁）。在sleep()休眠时间期满后，该线程不一定会立即执行，这是因为其它线程可能正在运行而且没有被调度为放弃执行，除非此线程具有更高的优先级。
2. wait()方法是Object类里的方法；当一个线程执行到wait()方法时，它就进入到一个和该对象相关的等待池中，同时失去（释放）了对象的机锁（暂时失去机锁，wait(long timeout)超时时间到后还需要返还对象锁）；其他线程可以访问；wait()使用notify或者notifyAlll或者指定睡眠时间来唤醒当前等待池中的线程。wiat()必须放在synchronized block中，否则会在program runtime时扔出”java.lang.IllegalMonitorStateException“异常。
3. 例子程序

package snippet;

public class MultiThread {

public static void main(String[] args) {

new Thread(new Thread1()).start();

try {

Thread.sleep(10);

} catch (InterruptedException e) {

e.printStackTrace();

}

new Thread(new Thread2()).start();

}

private static class Thread1 implements Runnable {

@Override

public void run() {

// 由于这里的Thread1和下面的Thread2内部run方法要用同一对象作为监视器，

//我们这里不能用this，因为在Thread2里面的this和这个Thread1的this不是同一个对象。

//我们用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。

synchronized (MultiThread.class) {

System.out.println("enter thread1...");

System.out.println("thread1 is waiting");

try {

// 释放锁有两种方式，第一种方式是程序自然离开监视器的范围，也就是离开了synchronized关键字管辖的代码范围，

//另一种方式就是在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方法释放锁。

MultiThread.class.wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("thread1 is going on...");

System.out.println("thread1 is being over!");

}

}

}

private static class Thread2 implements Runnable {

@Override

public void run() {

synchronized (MultiThread.class) {

System.out.println("enter thread2...");

System.out.println("thread2 notify other thread can release wait status..");

// 由于notify方法并不释放锁，即使thread2调用下面的sleep方法休息了10毫秒，

//但thread1仍然不会执行，因为thread2没有释放锁，所以Thread1无法得不到锁。

MultiThread.class.notify();

System.out.println("thread2 is sleeping ten millisecond...");

try {

Thread.sleep(10);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println("thread2 is going on...");

System.out.println("thread2 is being over!");

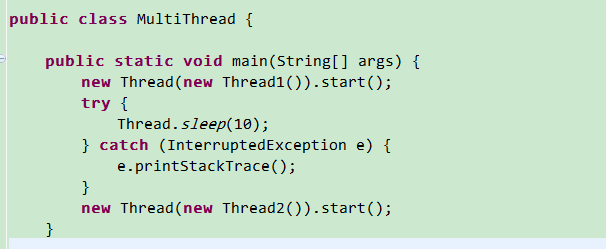
}

}

}

}

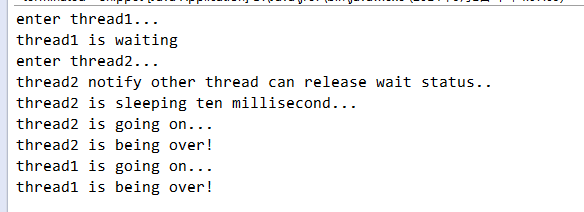
图片：







输出结果：



1. wait、sleep、notify、notifyAll
   1. wait():使一个线程处于等待状态，并且释放所持有的对象的lock。
   2. sleep():使一个正在运行的线程处于睡眠状态，是一个静态方法，调用此方法要捕捉InterruptedException(中断异常)异常。
   3. notify():唤醒一个处于等待状态的线程，注意的是在调用此方法的时候，并不能确切的唤醒某一个等待状态的线程，而是由JVM确定唤醒哪个线程，而且不是按优先级。
   4. notifyAll():唤醒所有处入等待状态的线程，注意并不是给所有唤醒线程一个对象的锁，而是让它们竞争。