#DeadLock

###实验过程

1.编写Deadlock.java

```
class A{
    synchronized void methodA(B b){
        b.last();
    }
    synchronized void last(){
        System.out.println("Inside A.last()");
    }
}
class B {
    synchronized void methodB(A a){
        a.last();
    }
    synchronized void last(){
        System.out.println("Inside B.last()");
    }
}
class Deadlock implements Runnable{
    A a = new A();
    B b = new B();

    Deadlock(){
        Thread t = new Thread(this);
        int count = 20000;

        t.start();
        while(count-->0);
        a.methodA(b);
    }

    public void run(){
        b.methodB(a);
    }
    public static void main(String args[]){
        new Deadlock();
    }
}
```

2.编译

```
javac Deadlock.java
```

3.编写Deadlock.bat，保存在Deadlock.class同一目录下

```
cd /d %~dp0
@echo off
:start
set /a var+=1
echo %var%
java Deadlock
if %var% leq 1000 GOTO start
pause
```

4.点击运行，产生死锁，理论上产生死锁的次数是随机的。

###死锁原因分析
死锁产生需要四个必要条件：

1.互斥条件：一个资源每次只能被一个进程使用

2.请求与保持条件：一个进程因请求资源而阻塞时，对已获得的资源保持不放

3.不剥夺条件：进程已获得的资源，在末使用完之前，不能强行剥夺

4.循环等待条件：若干进程之间形成一种头尾相接的循环等待资源关系

新线程t被调度时执行run()，调用a的last(),此时的主线程在执行a.methodA(b)时,需要执行b.last(). A,B均为使用关键字synchronized修饰的类，同一时刻只能有一个线程执行其中的代码，主线程和t都会因为没有得到资源而阻塞，等待对方执行结束，产生死锁。