什么是死锁:

所谓死锁，是指多个线程在运行过程中因争夺资源而造成的一种僵局，当进程处于这种僵持状态时，若无外力作用，它们都将无法再向前推进(在多线程程序中,使用了多把锁,造成线程之间相互等待.程序不往下走了。)

产生死锁的必要条件：

产生死锁必须同时满足以下四个条件，只要其中任一条件不成立，死锁就不会发生。

互斥条件：进程要求对所分配的资源进行排它性控制，即在一段时间内某资源仅为一进程所占用。

请求和保持条件：当进程因请求资源而阻塞时，对已获得的资源保持不放。

不剥夺条件：进程已获得的资源在未使用完之前，不能剥夺，只能在使用完时由自己释放。

环路等待条件：在发生死锁时，必然存在一个进程--资源的环形链。

如何避免死锁:

加锁顺序（线程按照一定的顺序加锁）:确保所有的线程都是按照相同的顺序获得锁,如果一个线程（比如线程3）需要一些锁，那么它必须按照确定的顺序获取锁。它只有获得了从顺序上排在前面的锁之后，才能获取后面的锁。

加锁时限（线程尝试获取锁的时候加上一定的时限，超过时限则放弃对该锁的请求，并释放自己占有的锁）

```
public class Demo05 {
    public static void main(String[] args) {
        MyRunnable mr = new MyRunnable();

        new Thread(mr).start();
        new Thread(mr).start();
    }
}

class MyRunnable implements Runnable {
    Object objA = new Object();
    Object objB = new Object();

    /*
    嵌套1 objA
    嵌套1 objB
    嵌套2 objB
    嵌套1 objA
```

```
     */
    @Override
    public void run() {
        synchronized (objA) {
            System.out.println("嵌套1 objA");
            synchronized (objB) {// t2, objA, 拿不到B锁,等待
                System.out.println("嵌套1 objB");
            }
        }

        synchronized (objB) {
            System.out.println("嵌套2 objB");
            synchronized (objA) {// t1 , objB, 拿不到A锁,等待
                System.out.println("嵌套2 objA");
            }
        }
    }
}
```