线程同步机制：

1，前提：如果我们创建多个线程，存在着共享数据，name就有可能出现线程的安全问题，当其中一

个线程操作共享数据时，还未操作完成，另外的线程就参与进来，导致对共享数据的操作出现问题

2，解决方式：要求一个线程操作共享书记处时，只有当其完成操作共享数据之后，其他线程才有机

会执行共享数据

方式1：同步代码块

```
synchronized（同步监视器）
{
//操作共享书记处的代码
}
```

注意：①同步监视器：任何一个类的对象都可以充当锁，要想保证线程的安全，要求所有的线程共同

使用同一把锁

②使用实现runnable接口的方式创建多线程，同步代码块中的锁，可以考虑使用this

如果使用继承thread类的方式，慎用this

③共享数据：多个线程需要共同操作共同的变量，明确哪部分是操作共享数据的代码块

方式2：同步方法，将操作共享数据的方法声明为synchronized

比如：

```
public synchronized void show（）
{
//操作共享数据的代码
}
```

注意：①对于非静态的方法而言，使用同步的话，默认锁为this，如果使用继承的方式实现多线程的

话，慎用

②对于静态的方法，如果使用同步的话，默认的锁为当前类本身

总结：

释放锁的操作：（wait）

①当前线程的同步方法，同步代码块执行结束

②当前线程在同步代码块，同步方法中遇到break、return中止了该代码块，该方法的继续执行
③当前线程在同步代码块，同步方法中出现了未处理的error或exception，导致异常结束
④当前线程在同步代码块、同步方法中执行了线程对象的wait（）方法，当前线程暂停，并释放锁

不会释放锁的操作：（sleep、wield、suspend（过时，可能导致死锁））
①线程执行同步代码块或同步方法时，程序调用thread.sleep（），thread.yield（）方法暂停当前线
程的执行
②线程执行同步代码块时，其他线程调用了该线程的suspend（）方法将线程挂起，该线程不会释放
锁（同步监视器）
尽量避免使用suspend（）和resume（）来控制线程

死锁：不同的线程分别占用对方需要的同步资源不放弃，都在等待对方放弃自己需要的同步资源，就
形成了线程的死锁（死锁是我们在使用同步时需要避免的问题）
解决方法：专门的算法、原则
尽量减少同步资源的定义

线程通信：（如下三个方法必须使用在同步代码块或者在同步代码块中）
wait（）：当在同步中，执行到此方法，则此线程等待，直至其他线程执行notify（）的方法，将其唤
醒，唤醒后继续其wait后面的代码
notify（）/notifyAll（）：在同步中，执行到此方法，则唤醒其他的某一个或所有的被wait的线程
例题：俩个线程交替打印1-100自然数；生产者消费者问题