一

＊ 此程序存在线程的安全问题，打印车票时，会出现重票和错票
＊ 1，线程安全问题存在的原因？
＊　　由于一个线程在操作共享数据过程中，未执行完毕的情况下，另外的线程参与进来，导致共享数据存在了安全问题
＊ 2，如何来解决线程的安全问题？
＊　　必须让一个线程操作共享数据完毕之后，草能让其他线程有机会参与共享数据的操作
＊ 3，java如何实现线程的安全：线程的同步机制
＊　　方式1：同步代码块
＊　　synchronized（同步监视器）{
＊　　//需要被同步的代码块（即为操作共享数据的代码）
＊　　}
＊　　共享数据：多个线程共同操作的同一个数据（变量）
＊　　同步监视器：(可以由任何对象的实例来充当，应该是一个成员变量)由一个类的对象来充当，那个线程获取此监视器，谁就执行大括号里被同步的代码，即为锁
＊　　要求：所有的线程必须公用同一把锁
＊

判断对象是不是同一个对象，需要使用“==”
＊　　方法2：同步方法
＊　　将操作共享数据的方法声明为synchronized，即此方法为同步方法，能够保证当其中一个线程执行此方法
＊　　时，其它线程在外等待直至此线程执行完此方法
＊　　>同步方法的锁：this充当的锁

*线程的同步的弊端：由于同一个时间只能有一个线程访问共享数据，效率变低了*

二，

＊　　在java中，引入了对象互斥锁的概念来保证共享数据操作的完整性
＊　　①每一个对象都对应一个可以称为互斥锁的标记，这个标记用来保证在任一时刻，只能有一个线程访问该对象

*　②关键字synchronized来与对象的互斥锁联系，当某个对象用synchronized修饰时，
*　　表明该对象在任一时刻只能由一个线程进行访问
*　③同步的局限性：导致程序的执行效率要降低
*　④同步方法（非静态的）的锁是this
*　⑤同步方法（静态的）的锁为当前对象

三，

关于懒汉式的线程安全问题，使用同步机制
对于一般的方法内，使用同步代码块，可以考虑使用this
对于静态方法而言，使用当前类本身充当锁

四，释放锁
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：
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

五，线程死锁：（处理线程同步时容易出现）
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
解决方法：专门的算法、原则

尽量减少同步资源的定义