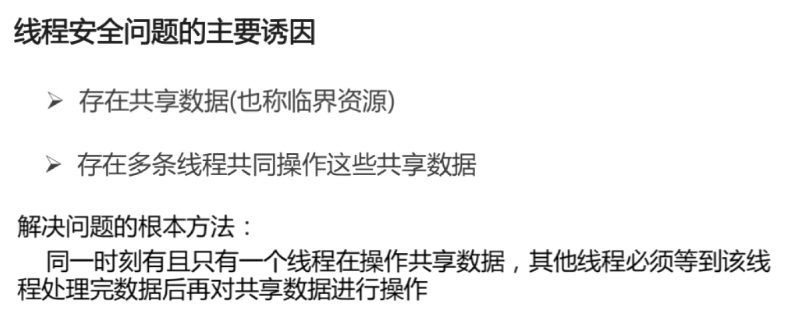
# 线程安全与同步

## 线程安全问题



## 什么情况下需要同步

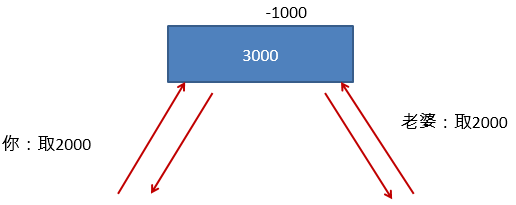
当多线程并发, 有多段代码同时执行时, 我们希望某一段代码执行的过程中CPU不要切换到其他线程工作. 这时就需要同步.

如果两段代码是同步的, 那么同一时间只能执行一段, 在一段代码没执行结束之前, 不会执行另外一段代码.

解决线程安全问题

前提：

如果我们创建的多个线程，存在着共享数据，那么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？



解决方式：

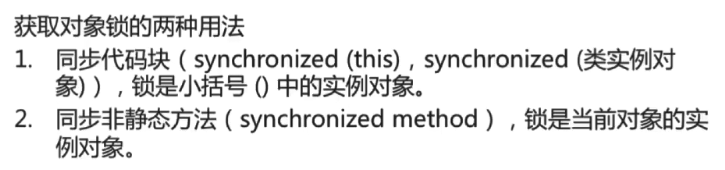
要求一个线程操作共享数据时，只有当其完成操作完成共享数据，其它线程才有机会执行共享数据。

## Synchronized

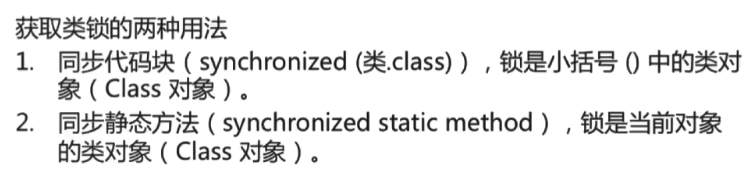
### 锁的本质



### 获取对象锁



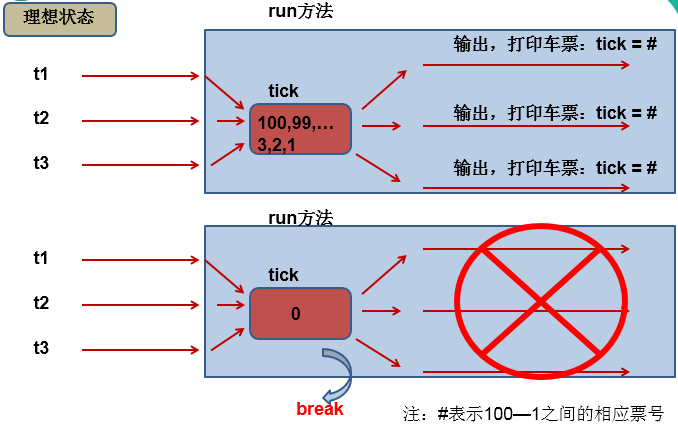
### 获取类锁

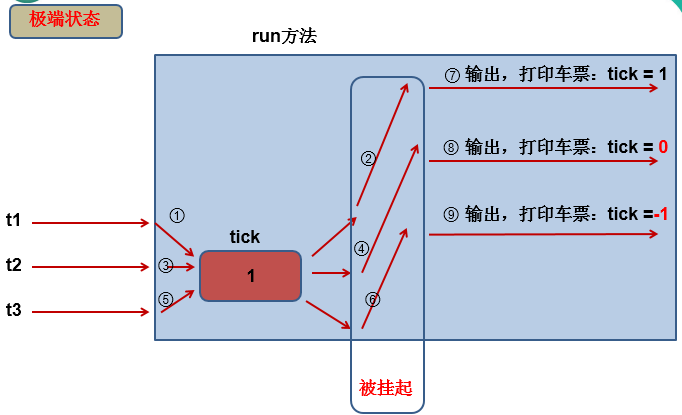


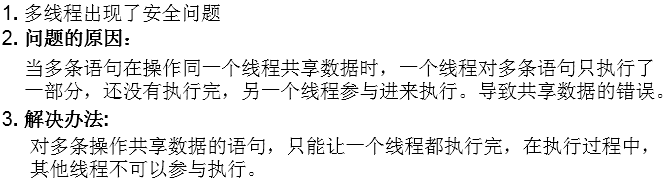
# 取票问题

## 模拟火车站售票程序

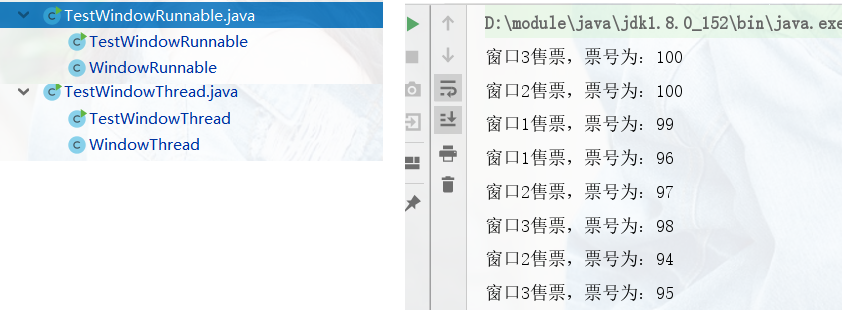
模拟火车站售票程序，开启三个窗口售票。TestWindow1.java







## 原始Runnable和Thread存在线程安全



## 方式一：Synchronized同步代码块：

### 概念

谁是共享数据？

**synchronized**(同步监视器){

//操作共享数据的代码

}

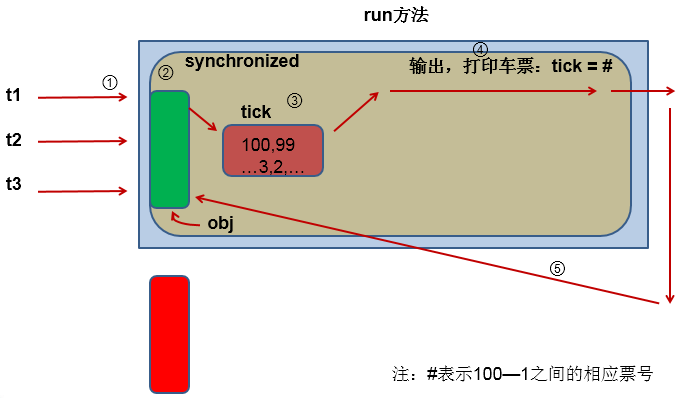
注：

1.同步监视器：俗称锁，任何一个类的对象都可以才充当锁。要想保证线程的安全，必须要求所有的线程共用同一把锁！

2.使用实现Runnable接口的方式创建多线程的话，同步代码块中的锁，可以考虑是this。如果使用继承Thread类的方式，慎用this!

对于静态方法而言，使用当前类本身充当锁

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



### 同步Runnable

**class** WindowRunnableSynchronizedCode **implements** Runnable {

**int** ticket = 100;// 共享

**public** **void** run() {

**while** (**true**) {

**synchronized** (**this**) {//this表示当前对象，本题中即为w

**if** (ticket > 0) {

**try** {

Thread.*currentThread*().*sleep*(10);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

System.*out*.println(Thread.*currentThread*().getName()

+ "售票，票号为：" + ticket--);

}

}

}

}

}

**public** **class** TestWindowRunnableSynchronizedCode {

**public** **static** **void** main(String[] args) {

WindowRunnableSynchronizedCode w = **new** WindowRunnableSynchronizedCode();

Thread t1 = **new** Thread(w);

Thread t2 = **new** Thread(w);

Thread t3 = **new** Thread(w);

t1.setName("窗口1");

t2.setName("窗口2");

t3.setName("窗口3");

t1.start();

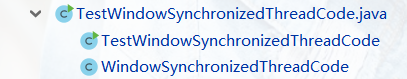
t2.start();

t3.start();

}

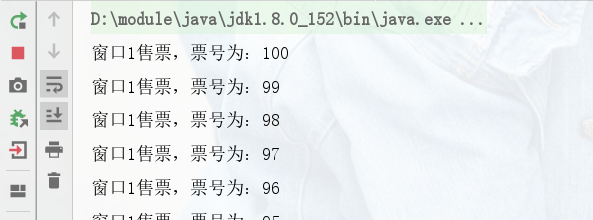
}

### 同步Thread



### 结果

票号按照顺序输出出



## 方式二：Synchronized同步方法：

### 概念

//非静态的同步方法的锁对象是神马?

//答:非静态的同步方法的锁对象是this

//静态的同步方法的锁对象是什么?

//是该类的字节码对象

将操作共享数据的方法声明为synchronized

**public** **synchronized** **void** show (String name){

….

}

//操作共享数据的代码

注：

1.对于非静态的方法而言，使用同步的话，默认锁为：this。如果使用在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，默认的锁为：当前类本身。以单例的懒汉式为例。 Class clazz = Singleton.class

### 同步Runnable

**class** WindowRunnableSynchronizedMethod **implements** Runnable {

**int** ticket = 100;// 共享数据

**public** **void** run() {

**while** (**true**) {

show();

}

}

**public** **synchronized** **void** show() {

**if** (ticket > 0) {

**try** {

Thread.*currentThread*().*sleep*(10);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

System.*out*.println(Thread.*currentThread*().getName() + "售票，票号为："

+ ticket--);

}

}

}

**public** **class** TestWindowRunnableSynchronizedMethod {

**public** **static** **void** main(String[] args) {

WindowRunnableSynchronizedMethod w = **new** WindowRunnableSynchronizedMethod();

Thread t1 = **new** Thread(w);

Thread t2 = **new** Thread(w);

Thread t3 = **new** Thread(w);

t1.setName("窗口1");

t2.setName("窗口2");

t3.setName("窗口3");

t1.start();

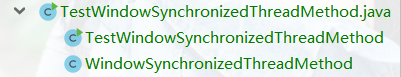
t2.start();

t3.start();

}

}

### 同步Thread



### 结果

票号按照顺序输出出

