1. 背景

例子：創建三個窗口賣票，總票數為100張.使用實現Runnable接口的方式

1.問題：賣票過程中，出現了重票、錯票 -->出現了線程的安全問題

2.問題出現的原因：當某個線程操作車票的過程中，尚未操作完成時，其他線程參與進來，也操

作車票。

3.如何解決：當一個線程a在操作ticket的時候，其他線程不能參與進來。 直到線程a操作完   
 ticket時，其他線程才可以開始操作ticket。這種情況即使線程a出現了阻塞，也不能被改變。

1. Java解決方案：同步機制  
   在Java中，我們通過同步機制，來解決線程的安全問題。

**方式一：同步代碼塊**

synchronized(同步監視器){

//需要被同步的代碼

　　}

說明：1.操作共享數據的代碼，即為需要被同步的代碼

（不能包含代碼多了，也不能包含代碼少了。）

2.共享數據：多個線程共同操作的變量。比如：ticket就是共享數據。

3.同步監視器，俗稱：鎖。任何一個類的對象，都可以充當鎖。

要求：多個線程必須要共用同一把鎖。

　　補充：在實現Runnable接口創建多線程的方式中，我們可以考慮使用this充當同步監視器

　　　　　 在繼承Thread類創建多線程的方式中，慎用this充當同步監視器，考慮使用當前類充

當同步監視器。

**方式二：同步方法**

如果操作共享數據的代碼完整的聲明在一個方法中，我們不妨將此方法聲明同步的。

關於同步方法的總結：

1. 同步方法仍然涉及到同步監視器，只是不需要我們顯式的聲明。

2. 非靜態的同步方法，同步監視器是：this

靜態的同步方法，同步監視器是：當前類本身

**方式三：Lock鎖 --- JDK5.0新增**

面試題：synchronized 與 Lock的異同？

相同：二者都可以解決線程安全問題

不同：synchronized機制在執行完相應的同步代碼以後，自動的釋放同步監視器

Lock需要手動的啟動同步（lock()），同時結束同步也需要手動的實現（unlock()）

優先使用順序：從效能的角度來看

Lock --> 同步代碼塊（已經進入了方法體，分配了相應資源） --> 同步方法（在方法體之外）

1. 利弊  
   同步的方式，解決了線程的安全問題。---好處

操作同步代碼時，只能有一個線程參與，其他線程等待。相當於是一個單線程的過程，效率低。

1. 面試題：Java是如何解決線程安全問題的，有幾種方式？並對比幾種方式的不同

面試題：synchronized和lock方式解決線程安全問題的對比