1、课程名称：同步与死锁

魔乐科技 JAVA就业专家 www.MLDNJAVA.cn

2、本课程预计讲解的知识点

本章目标

- 了解线程同步的作用
- 了解同步代码块及同步方法的作用
- 了解死锁的产生

说明：

在多线程的开发中，同步与死锁的概念是非常重要的，在本章中，读者一定要重点掌握以下几点：

- 那里需要同步
- 如何实现同步，代码了解即可
- 以及实现同步之后会有那些的负作用，了解

代码并不不要求可以完整的编写，但是概念必须清楚。

魔乐科技 JAVA就业专家 www.MLDNJAVA.cn

3、具体内容

3.1、问题的引出

以卖火车票为例，如果现在要是想买火车票的话可以去火车站买或者去各个售票点，但是不管有多少个地方可以买火车票，最终一趟列车的车票数量是固定的，如果把各个售票点理解为各个线程的话，则所有线程应该共同拥有同一份的票数。

```
class MyThread implements Runnable{
	private int ticket = 5 ;	// 假设一共有 5 张票
	public void run(){
		for(int i=0;i<100;i++){
			if(ticket>0){	// 还有票
				try{
					Thread.sleep(300) ;	// 加入延迟
				}catch(InterruptedException e){
					e.printStackTrace() ;
				}
				System.out.println("卖票：ticket = " + ticket-- );
			}
		}
	}
};
public class SyncDemo01{
	public static void main(String args[]){
		MyThread mt = new MyThread() ;// 定义线程对象
		Thread t1 = new Thread(mt) ;	// 定义 Thread 对象
		Thread t2 = new Thread(mt) ;	// 定义 Thread 对象
		Thread t3 = new Thread(mt) ;	// 定义 Thread 对象
		t1.start() ;
		t2.start() ;
		t3.start() ;
	}
};
```


魔乐科技 JAVA就业专家 www.MLDNJAVA.cn

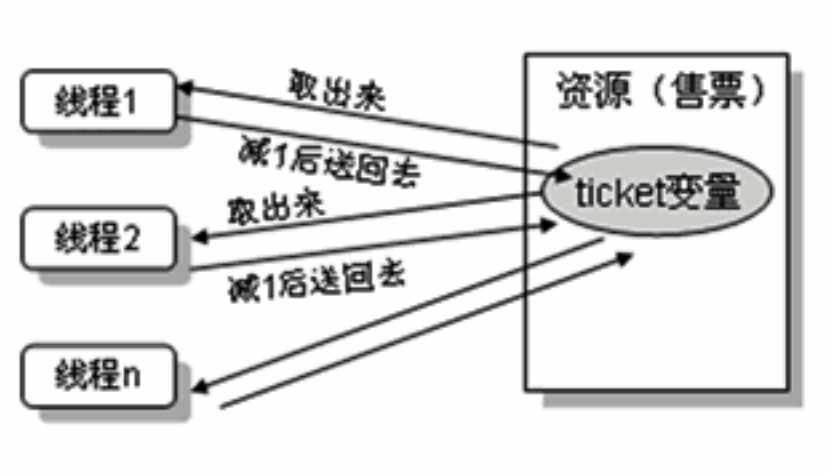
```
C:\WINDOWS\system32\cmd.exe
D:\syndemo>java SyncDemo01
卖票: ticket = 5
卖票: ticket = 4
卖票: ticket = 3
卖票: ticket = 2
卖票: ticket = 1
卖票: ticket = 0
卖票: ticket = -1

D:\syndemo>
```

从执行结果可以发现，卖出的票数成负数，程序代码出现了问题。

程序的问题

- 从运行结果可以发现，程序中加入了延迟操作，所以在运行的最后出现了负数的情况，那么为什么现在会产生这样的问题呢？
- 从上面的操作代码可以发现对于票数的操作步骤如下：
  - 1、判断票数是否大于0，大于0则表示还有票可以卖
  - 2、如果票数大于0，则卖票出去
- 但是，在上面的操作代码中，在第1步和第2步之间加入了延迟操作，那么一个线程就有可能在还没有对票数进行减操作之前，其他线程就已经将票数减少了，这样一来就会出现票数为负的情况



魔乐科技 JAVA就业专家 www.MLDNJAVA.cn

问题的解决

- 如果想解决这样的问题，就必须使用同步，所谓的同步就是指多个操作在同一个时间段内只能有一个线程进行，其他线程要等待此线程完成之后才可以继续执行。

3.2、使用同步问题

要想解决资源共享的同步操作问题，可以使用同步代码块及同步方法两种方式完成。

3.2.1、同步代码块

之前已经介绍过了代码块分为四种：

- 普通代码块：是直接定义在方法之中的
- 构造块：是直接定义在类中的，优先于构造方法执行，重复调用
- 静态块：是使用 static 关键字声明的，优先于构造块执行，只执行一次。
- 同步代码块：使用 synchronized 关键字声明的代码块，称为同步代码块。

魔乐科技 JAVA就业专家 www.MLDNJAVA.cn

同步代码块

- 在代码块上加上"synchronized"关键字的话，则此代码块就称为同步代码块。
- 同步代码块格式：
  - synchronized(同步对象){
    需要同步的代码 ;
    }

同步的时候必须指明同步的对象，一般情况下会将当前对象作为同步的对象，使用 this 表示。

```
class MyThread implements Runnable{
	private int ticket = 5 ;	// 假设一共有 5 张票
	public void run(){
		for(int i=0;i<100;i++){
			synchronized(this){	// 要对当前对象进行同步
				if(ticket>0){	// 还有票
					try{
						Thread.sleep(300) ;	// 加入延迟
					}catch(InterruptedException e){
						e.printStackTrace() ;
					}
					System.out.println("卖票：ticket = " + ticket-- );
				}
			}
```


魔乐科技 JAVA就业专家 www.MLDNJAVA.cn

```
		}
	}
};
public class SyncDemo02{
	public static void main(String args[]){
		MyThread mt = new MyThread() ;// 定义线程对象
		Thread t1 = new Thread(mt) ;	// 定义 Thread 对象
		Thread t2 = new Thread(mt) ;	// 定义 Thread 对象
		Thread t3 = new Thread(mt) ;	// 定义 Thread 对象
		t1.start() ;
		t2.start() ;
		t3.start() ;
	}
};
```

```
C:\WINDOWS\system32\cmd.exe
卖票: ticket = 1

D:\syndemo>java SyncDemo02
卖票: ticket = 5
卖票: ticket = 4
卖票: ticket = 3
卖票: ticket = 2
卖票: ticket = 1

D:\syndemo>
```

从运行结果可以发现，成许加入了同步操作，所以不会产生负数的情况，但是，程序的执行效率明显降低很多的。

魔乐科技 JAVA就业专家 www.MLDNJAVA.cn

3.2.2、同步方法

同步方法

- 除了可以将需要的代码设置成同步代码块之外，也可以使用synchronized关键字将一个方法声明成同步方法。
- 同步方法定义格式：
  - synchronized 方法返回值 方法名称(参数列表){}


```
class MyThread implements Runnable{
	private int ticket = 5 ;	// 假设一共有 5 张票
	public void run(){
		for(int i=0;i<100;i++){
			this.sale() ;	// 调用同步方法
		}
	}
	public synchronized void sale(){	// 声明同步方法
		if(ticket>0){	// 还有票
			try{
				Thread.sleep(300) ;	// 加入延迟
```


魔乐科技 JAVA就业专家 www.MLDNJAVA.cn

```
			}catch(InterruptedException e){
				e.printStackTrace() ;
			}
			System.out.println("卖票：ticket = " + ticket-- );
		}
	}
};
public class SyncDemo03{
	public static void main(String args[]){
		MyThread mt = new MyThread() ;// 定义线程对象
		Thread t1 = new Thread(mt) ;	// 定义 Thread 对象
		Thread t2 = new Thread(mt) ;	// 定义 Thread 对象
		Thread t3 = new Thread(mt) ;	// 定义 Thread 对象
		t1.start() ;
		t2.start() ;
		t3.start() ;
	}
};
```

3.3、死锁