线程

### 一、学习目标

1. 能够描述Java中多线程运行原理  
2. 能够使用继承类的方式创建多线程  
3. 能够使用实现接口的方式创建多线程  
4. 能够说出实现接口方式的好处  
5. 能够解释安全问题的出现的原因  
6. 能够使用同步代码块解决线程安全问题  
7. 能够使用同步方法解决线程安全问题  
8. 能够说出线程6个状态的名称

### 二、学习指南

1. 线程程序的随机性&开启线程的第二种方式
2. 开启线程两种方式的区别
3. 线程安全问题&同步技术
4. 线程状态&线程通讯

### 第一堂课

本节知识点

01\_今日内容介绍

02\_多线程原理\_随机性打印结果

03\_多线程原理\_多线程内存图解

04\_Thread类的常用方法\_获取线程名称的方法

05\_Thread类的常用方法\_设置线程名称的方法

视频时间

27分33秒

本节目标

了解今天所学内容

理解多线程程序的随机性

了解多线程程序在内存中的基本划分

掌握获取线程名称的两种方式

掌握设置线程名称的两种方式

#### 今日内容介绍

##### 1.1.1知识概述

* 今日内容介绍

##### 1.1.2视频详情



##### 1.1.3总结与补充

无

##### 1.1.4课堂提问与练习

无

##### 1.1.5习题答案

无

#### 多线程原理\_随机性打印结果

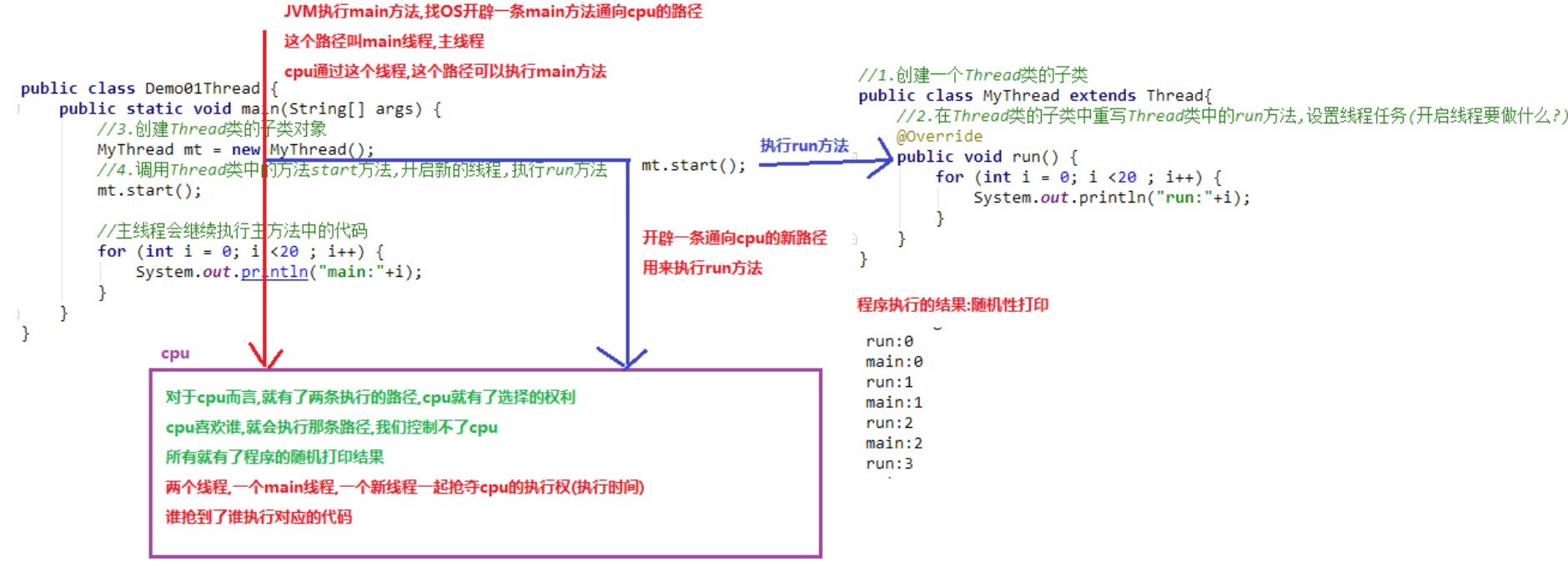
##### 1.2.1知识概述

* 多线程原理\_随机性打印结果

##### 1.2.2视频详情



##### 1.2.3总结与补充



##### 1.2.4课堂提问与练习

无

##### 1.2.5习题答案

无

#### 多线程原理\_多线程内存图解

##### 1.3.1知识概述

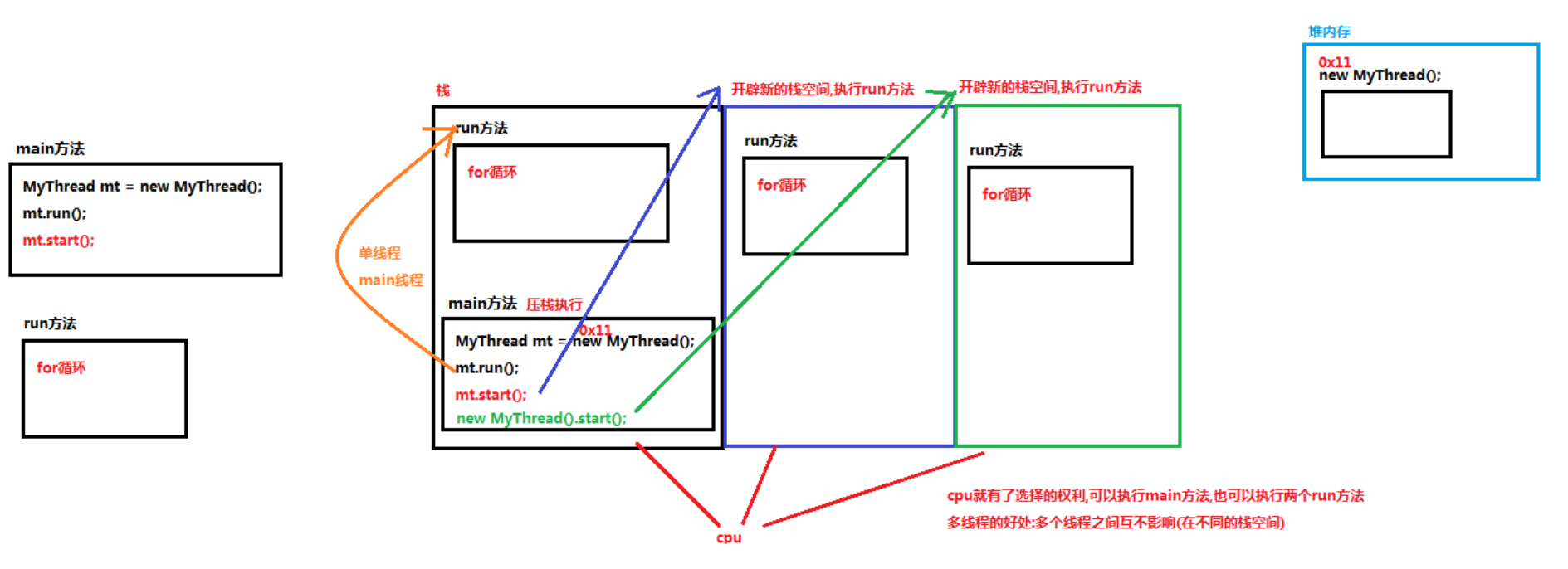
* 每个线程都有自己独立的一块栈空间

##### 1.3.2视频详情



##### 1.3.3总结与补充

多线程执行时，在栈内存中，其实每一个执行线程都有一片自己所属的栈内存空间。进行方法的压栈和弹栈



##### 1.3.4课堂提问与练习

无

##### 1.3.5习题答案

无

#### Thread类的常用方法\_获取线程名称的方法

##### 1.4.1知识概述

* getName();
* currentThread.getName();

##### 1.4.2视频详情



##### 1.4.3总结与补充

public String getName() :获取当前线程名称。

public static Thread currentThread() :返回对当前正在执行的线程对象的引用

##### 1.4.4课堂提问与练习

无

##### 1.4.5习题答案

无

#### Thread类的常用方法\_设置线程名称的方法

##### 1.5.1知识概述

* setName();
* new Thread(“线程名字”);

##### 1.5.2视频详情



##### 1.5.3总结与补充

public Thread(String name) :分配一个指定名字的新的线程对象

public Thread(Runnable target,String name) :分配一个带有指定目标新的线程对象并指定名字

public void setName(String name) : 为线程对象设置名字.

##### 1.5.4课堂提问与练习

无

##### 1.5.5习题答案

无

### 第二堂课

本节知识点：

06\_Thread类的常用方法\_sleep

07\_创建多线程程序的第二种方式\_实现Runnable接口

08\_Thread和Runnable的区别

09\_匿名内部类方式实现线程的创建

视频时间：

21分28秒

本节目标 ：

1. 掌握sleep方法的使用
2. 掌握实现Runnable接口开启线程的实现方式
3. 独立阐述两种开启线程的区别
4. 独立编写代码, 使用匿名内部类开启线程

#### Thread类的常用方法\_sleep

##### 2.1.1知识概述

* sleep方法演示

##### 2.1.2视频详情



##### 2.1.3总结与补充

无

##### 2.1.5习题答案

无

#### 创建多线程程序的第二种方式\_实现Runnable接口

##### 2.2.1知识概述

* 开启线程的第二种方式

##### 2.2.2视频详情



##### 2.2.3总结与补充

1. 定义Runnable接口的实现类，并重写该接口的run()方法，该run()方法的方法体同样是该线程的线程执行体。  
2. 创建Runnable实现类的实例，并以此实例作为Thread的target来创建Thread对象，该Thread对象才是真正的线程对象。  
3. 调用线程对象的start()方法来启动线程



##### 2.2.4课堂提问与练习

无

##### 2.2.5习题答案

无

#### Thread和Runnable的区别

##### 2.3.1知识概述

* 两种开启线程的区别

##### 2.3.2视频详情



##### 2.3.3总结与补充

实现Runnable接口创建多线程程序的好处:

1.避免了单继承的局限性

一个类只能继承一个类(一个人只能有一个亲爹),类继承了Thread类就不能继 承其他的类

实现了Runnable接口,还可以继承其他的类,实现其他的接口

2.增强了程序的扩展性,降低了程序的耦合性(解耦)

实现Runnable接口的方式,把设置线程任务和开启新线程进行了分离(解耦)

实现类中,重写了run方法:用来设置线程任务

创建Thread类对象,调用start方法:用来开启新线程

##### 2.3.4课堂提问与练习

无

##### 2.3.5习题答案

无

#### 匿名内部类方式实现线程的创建

##### 2.4.1知识概述

* 使用匿名内部类开启线程

##### 2.4.2视频详情



##### 2.4.3总结与补充



**2.4.4课堂提问与练习**

无

##### 2.4.5习题答案

无

### 第三堂课

本节知识点：

10\_线程安全问题的概述

11\_线程安全问题的代码实现

12\_线程安全问题产生的原理

13\_解决线程安全问题\_同步代码块

视频时间:

31分31秒

本节目标 ：

1. 了解多线程程序共享资源的时候, 会有安全隐患
2. 独立编写售票案例代码
3. 理解线程安全问题的产生原理
4. 掌握同步代码块的使用

#### 线程安全问题的概述

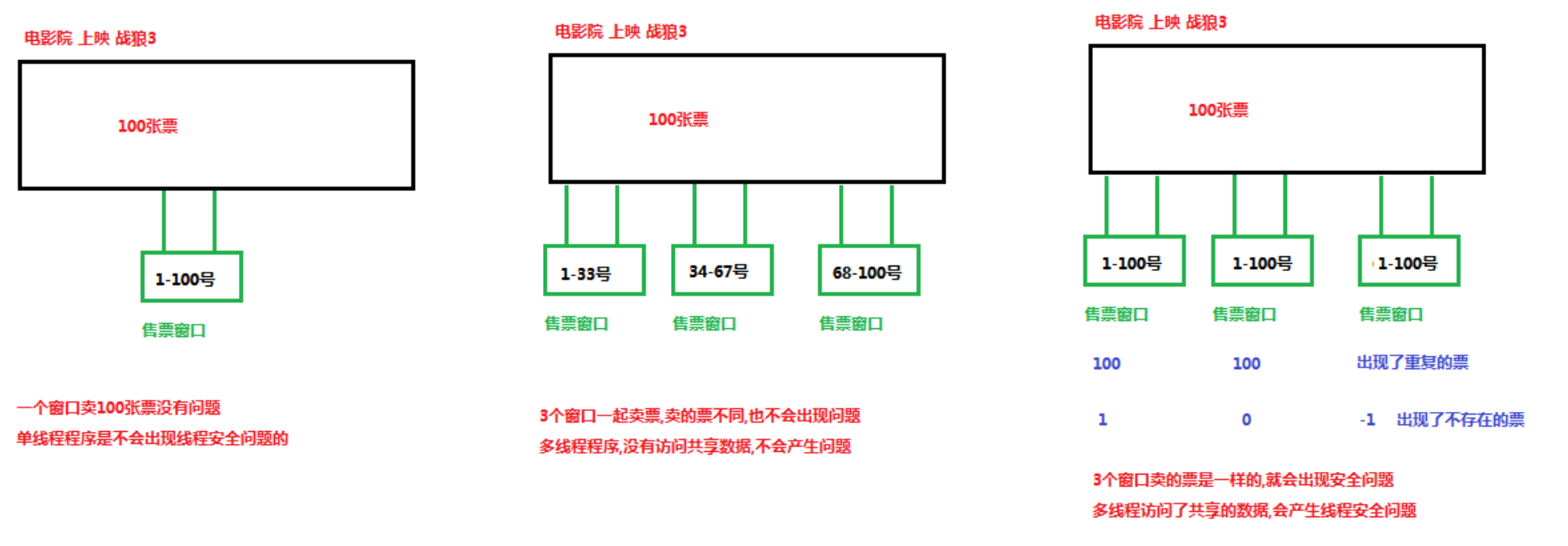
##### 3.1.1知识概述

* 线程安全问题引入

##### 3.1.2视频详情



##### 3.1.3总结与补充



##### 3.1.4课堂提问与练习

无

##### 3.1.5习题答案

无

#### 线程安全问题的代码实现

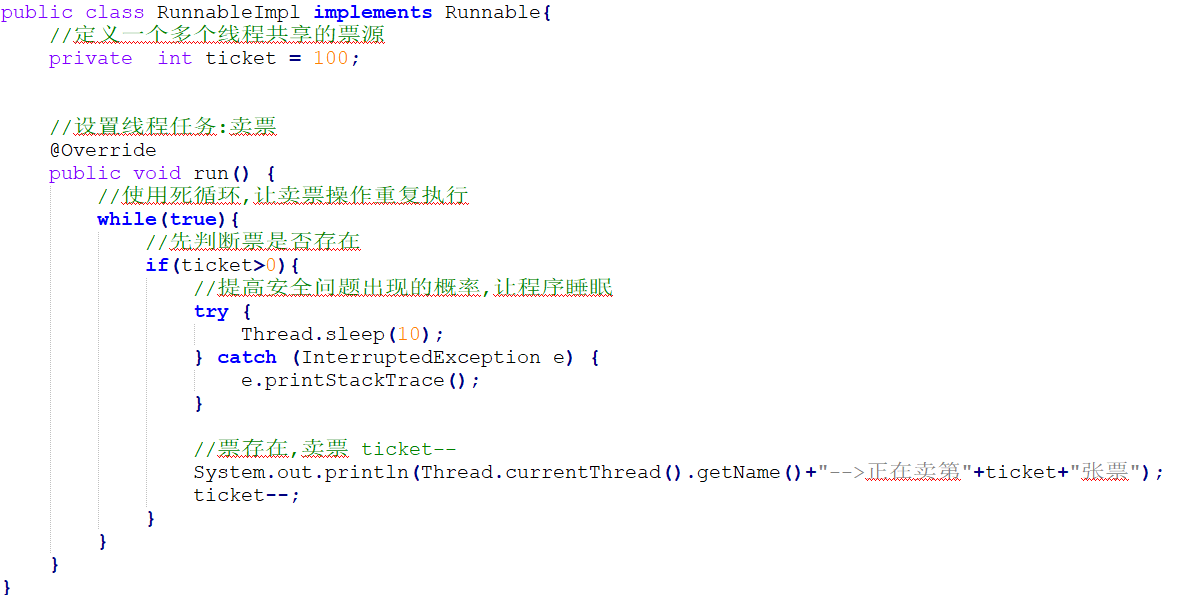
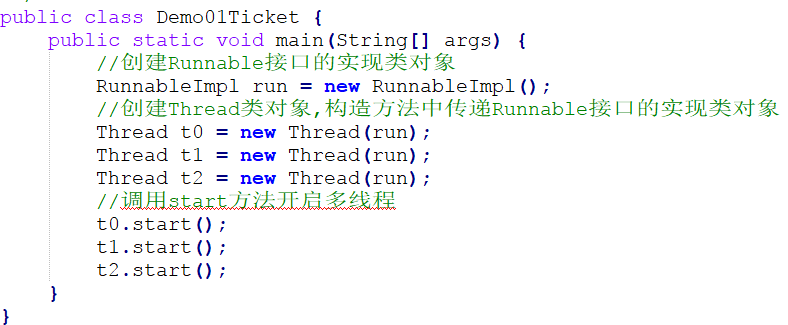
##### 3.2.1知识概述

* 多线程售票案例代码实现

##### 3.2.2视频详情



##### 3.2.3总结与补充



##### 3.2.4课堂提问与练习

无

##### 3.2.5习题答案

无

#### 线程安全问题产生的原理

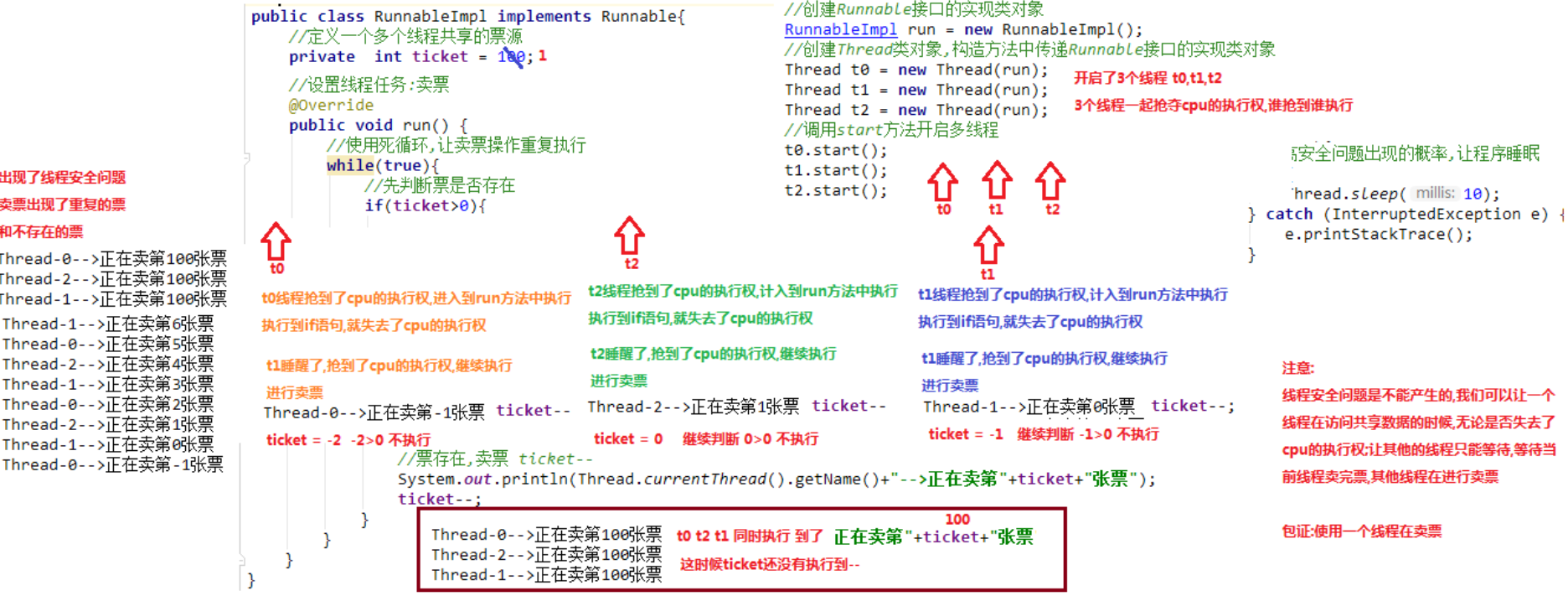
##### 3.3.1知识概述

* 画图分析线程安全问题所在

##### 3.3.2视频详情



##### 3.3.3总结与补充



##### 3.3.4课堂提问与练习

无

##### 3.3.5习题答案

无

#### 解决线程安全问题\_同步代码块

##### 3.3.1知识概述

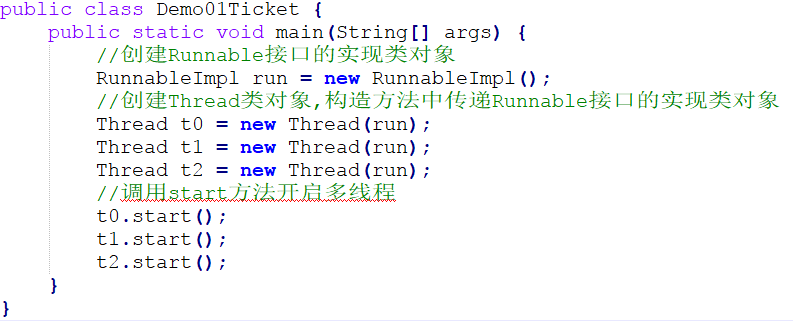
* 同步代码块解决线程安全问题

##### 3.3.2视频详情



##### 3.3.3总结与补充

同步代码块： synchronized 关键字可以用于方法中的某个区块中，表示只对这个区块的资源实行互斥访问。



格式:

synchronized(锁对象){

可能会出现线程安全问题的代码(访问了共享数据的代码)

}

注意:

1.通过代码块中的锁对象,可以使用任意的对象

2.但是必须保证多个线程使用的锁对象是同一个

3.锁对象作用:

把同步代码块锁住,只让一个线程在同步代码块中执行

##### 3.3.4课堂提问与练习

无

##### 3.3.5习题答案

无

### 第四堂课

本节知识点：

14\_同步技术的原理

15\_解决线程安全问题\_同步方法

16\_静态同步方法

17\_解决线程安全问题\_Lock锁

视频时间：

25分26秒

本节目标 ：

理解同步技术的原理

掌握同步方法的使用

了解静态同步方法的使用

掌握Lock锁的使用方式

#### 4.1 同步技术的原理

##### 4.1.1知识概述

* 同步技术的原理

##### 4.1.2视频详情



##### 4.1.3总结与补充



##### 4.1.4课堂提问与练习

无

##### 4.1.5习题答案

无

#### 4.2 解决线程安全问题\_同步方法

##### 4.2.1知识概述

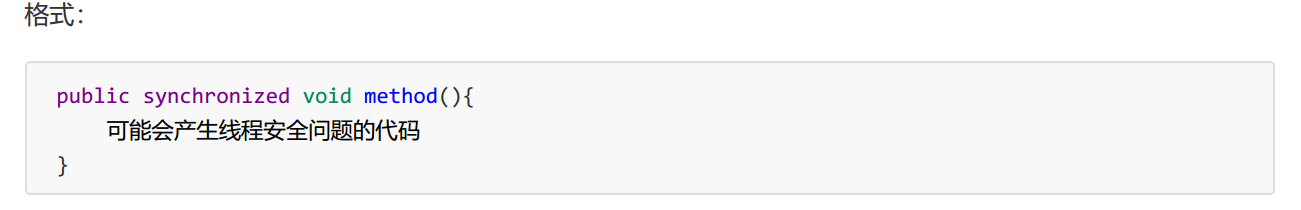
* 同步方法 & 同步方法的锁对象

##### 4.2.2视频详情



##### 4.2.3总结与补充

同步方法:使用synchronized修饰的方法,就叫做同步方法,保证A线程执行该方法的时候,其他线程只能在方法外等着。



##### 4.2.4课堂提问与练习

无

##### 4.2.5习题答案

无

#### 4.3 静态同步方法

##### 4.3.1知识概述

* **静态同步方法**

##### 4.3.2视频详情



##### 4.3.3总结与补充

对于static方法,我们使用当前方法所在类的字节码对象(类名.class)。

##### 4.3.4课堂提问与练习

无

##### 4.3.5习题答案

无

#### 4.4 解决线程安全问题\_Lock锁

##### 4.4.1知识概述

* lock方法, unlock方法

##### 4.4.2视频详情



##### 4.4.3总结与补充

public void lock() :加同步锁。  
public void unlock() :释放同步锁



##### 4.4.4课堂提问与练习

无

##### 4.4.5习题答案

无

### 第五堂课

本节知识点：

18\_线程状态概述

19\_等待唤醒案例分析

20\_等待唤醒案例代码实现

21\_Object类中wait带参方法和notifyAll方法

视频时间：

28分38秒

本节目标 ：

了解API中线程的6种状态

理解等待唤醒实现线程通讯的思路

独立编写等待唤醒案例

了解wait带参方法和notifyAll方法的特点

#### 5.1 线程状态概述

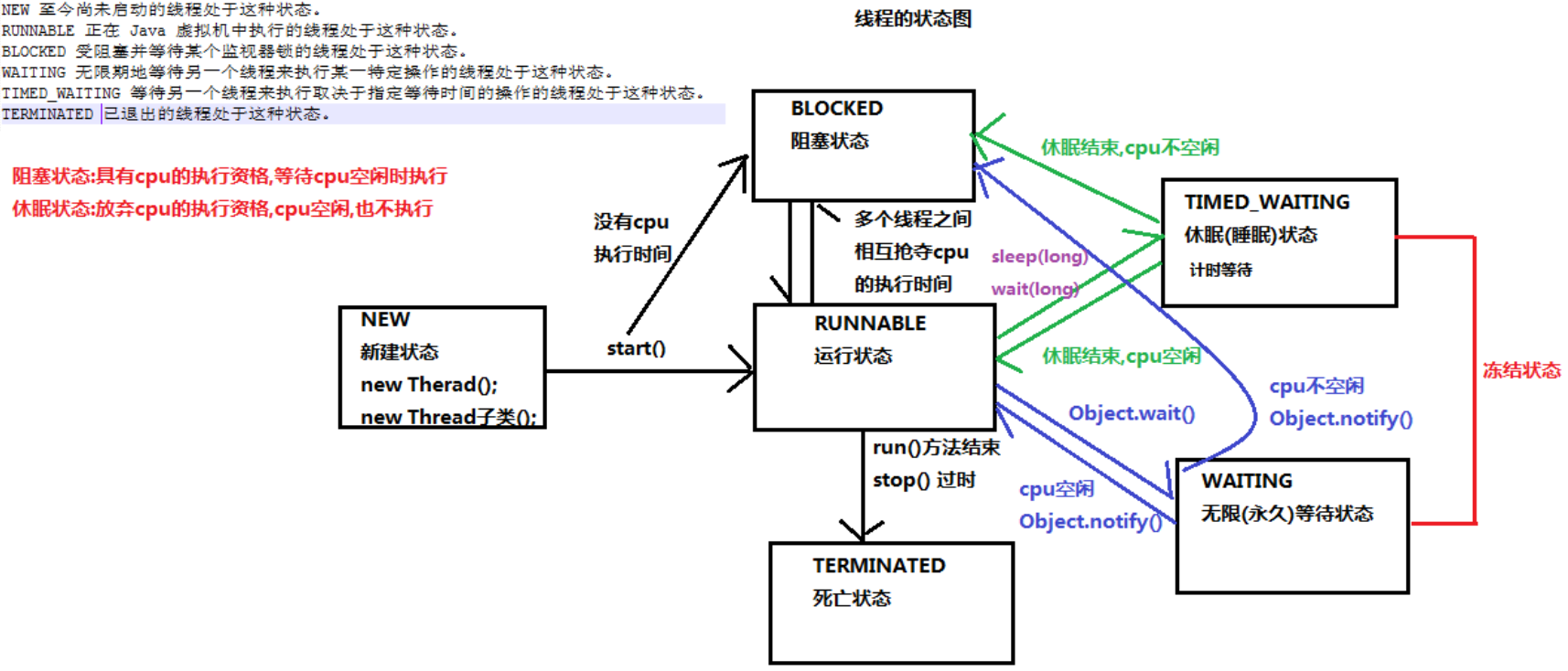
##### 5.1.1知识概述

* 线程的6中状态

##### 5.1.2视频详情



##### 5.1.3总结与补充



##### 5.1.4课堂提问与练习

无

##### 5.1.5习题答案

无

#### 5.2 等待唤醒案例分析

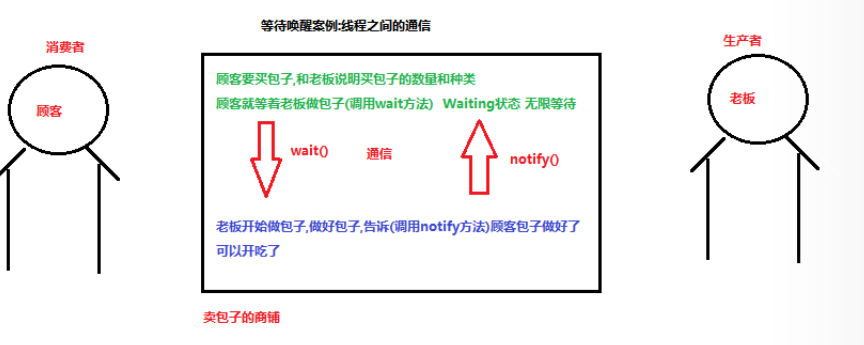
##### 5.2.1知识概述

* 线程通讯案例分析

##### 5.2.2视频详情



##### 5.2.3总结与补充



##### 5.2.4课堂提问与练习

无

##### 5.2.5习题答案

无

#### 5.3 等待唤醒案例代码实现

##### 5.3.1知识概述

* 线程通讯代码实现
* wait方法
* notify方法

##### 5.3.2视频详情



##### 5.3.3总结与补充

等待唤醒案例:线程之间的通信

创建一个顾客线程(消费者):告知老板要的包子的种类和数量,调用wait方法,放弃cpu的执行,进入到WAITING状态(无限等待)

创建一个老板线程(生产者):花了5秒做包子,做好包子之后,调用notify方法,唤醒顾客吃包子

注意:

顾客和老板线程必须使用同步代码块包裹起来,保证等待和唤醒只能有一个在执行

同步使用的锁对象必须保证唯一

只有锁对象才能调用wait和notify方法

Obejct类中的方法

void wait()

在其他线程调用此对象的 notify() 方法或 notifyAll() 方法前，导致当前线程等待。

void notify()

唤醒在此对象监视器上等待的单个线程。会继续执行wait方法之后的代码



##### 5.3.4课堂提问与练习

无

##### 5.3.5习题答案

无

#### 5.4 Object类中wait带参方法和notifyAll方法

##### 5.3.1知识概述

* Object类中wait带参方法和notifyAll方法

##### 5.3.2视频详情



##### 5.3.3总结与补充

进入到TimeWaiting(计时等待)有两种方式

1.使用sleep(long m)方法,在毫秒值结束之后,线程睡醒进入到Runnable/Blocked状态

2.使用wait(long m)方法,wait方法如果在毫秒值结束之后,还没有被notify唤醒,就会自动醒来,线程睡醒进入到Runnable/Blocked状态

唤醒的方法:

void notify() 唤醒在此对象监视器上等待的单个线程。

void notifyAll() 唤醒在此对象监视器上等待的所有线程。

##### 5.3.4课堂提问与练习

无

##### 5.3.5习题答案

无