# 一。同步

**class** InterRunnable **implements** Runnable

{ **private** **int** tickets=5;

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

// **TODO** Auto-generated method stub

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

{

**if**(tickets>0){

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

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

e.printStackTrace();

}

System.*out*.println(Thread.*currentThread*().getName()+"出售票"+**this**.tickets--);

}

}

}

}

**public** **class** Ma{

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

{

InterRunnable interRunnable = **new** InterRunnable();

Thread thread = **new** Thread(interRunnable,"票1");

Thread thread2 = **new** Thread(interRunnable,"票2");

thread.start();

thread2.start();

}

}

## .运行

票1出售票5

票2出售票4

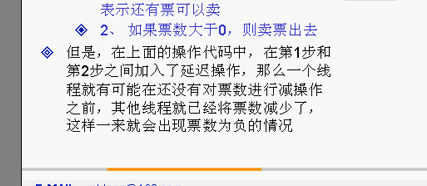
票1出售票3

票2出售票2

票1出售票1

票2出售票0

票2出售票-1



# 解决方法

# 1. 形成同步代码块synchronized (this)，同步操作

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

{ 形成同步代码块

**synchronized** (**this**) {

**if**(**this**.tickets>0){

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

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

e.printStackTrace();

}

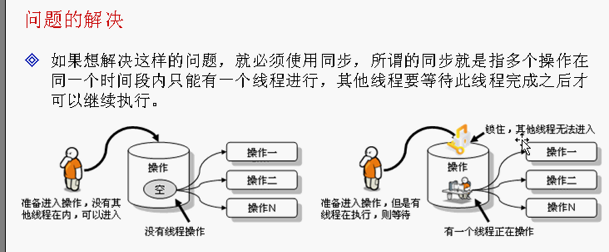
System.*out*.println(Thread.*currentThread*().getName()+"出售票"+**this**.tickets--);

}

}

}

2.



# 2.增加同步方法

@Override

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

// **TODO** Auto-generated method stub

sale();

}

**private** **synchronized** **void** sale() {

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

{

**if**(**this**.tickets>0){

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

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

e.printStackTrace();

}

System.*out*.println(Thread.*currentThread*().getName()+"出售票"+**this**.tickets--);

}

}

}

}

# 二 死锁（过多的同步产生）

**class** Fang

{

**public** **synchronized** **void** say( Zhang zhang) {

System.*out*.println(" 那钱来");

zhang.gave();

}

**public** **synchronized** **void** gave() {

System.*out*.println(" 钱到手了");

}

}

**class** Zhang{

**public** **synchronized** **void** say(Fang fang) {

System.*out*.println(" 我给你钱");

fang.gave();

}

**public** **synchronized** **void** gave() {

System.*out*.println(" 弟弟回来");

}

}

**public** **class** Sisuo **implements** Runnable{

Fang fang = **new** Fang();

Zhang zhang = **new** Zhang();

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

{

Sisuo sisuo = **new** Sisuo();

}

@Override

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

// **TODO** Auto-generated method stub

fang.say(zhang);

}

**public** Sisuo(){

**new** Thread(**this**).start();

zhang.say(fang);

}

}

## 2.运行结果，结果死在这里不能继续运行了

我给你钱

那钱来

# 这种只要了解就行，只需要记住，多个线程要注意使用同步，但是多个同步会造成同步

# 三，生产者和消费者

# 1.产品

**public** **class** Info {

String name;

String content;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getContent() {

**return** content;

}

**public** **void** setContent(String content) {

**this**.content = content;

}

}

# 2.生产者

**public** **class** Product **implements** Runnable{

Info info = **null**;

@Override

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

// **TODO** Auto-generated method stub

//不断的生产

**for** (**int** i = 0; i < 50; i++) {

**try** {

Thread.*sleep*(3000);

} **catch** (InterruptedException e) {

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

e.printStackTrace();

}

**if** (i%2==1) {

info.setContent("www.baidiu.com"+i);

info.setName("zhang"+i);

}**else** {

info.setContent("www.sohu.com"+i);

info.setName("healer"+i);

}

}

}

**public** Product(Info info)

{

**this**.info = info;

}

}

# 3.消费者

**public** **class** costumer **implements** Runnable{

Info info = **null**;

**public** costumer(Info info) {

**super**();

**this**.info = info;

}

//不断的取取内容

@Override

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

// **TODO** Auto-generated method stub

**for** (**int** i = 0; i < 50; i++) {

**try** {

Thread.*sleep*(3000);

} **catch** (InterruptedException e) {

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

e.printStackTrace();

}

System.*out*.println(**this**.info.getContent()+**this**.info.getName());

}

}

}

# 4.测试号

**public** **class** TestInfo {

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

// **TODO** Auto-generated method stub

Info info = **new** Info();

costumer costumer1 = **new** costumer(info);

Product product = **new** Product(info);

**new** Thread(costumer1).start();

**new** Thread(product).start();

}

}

## 5.结果，出现了重复取值

www.baidiu.com1zhang1

www.baidiu.com1zhang1

www.sohu.com2healer2

www.baidiu.com3zhang3

www.sohu.com4healer4

www.baidiu.com5zhang5

www.sohu.com6healer6

www.baidiu.com7zhang7

www.sohu.com8healer8

www.baidiu.com9zhang9

www.sohu.com10healer10

www.baidiu.com11zhang11

www.sohu.com12healer12

# 重新布置

# 1.产品

**public** **class** Info {

String name;

String content;

**public** **synchronized** **void** set(String name,String content){

**this**.name = name;

**this**.content = content;

}

**public** **synchronized** **void** get(String name,String content){

System.*out*.println(**this**.getContent()+**this**.getName());

}

## 2.可以解决部分问题，但是还是不能够避免，

## Supper.wrate. 让其他的线程先执行

## Supper.notify 唤醒这个线程

