Java面向对象程序设计课程设计实验报告五

实验人：王维

学号：320210942931

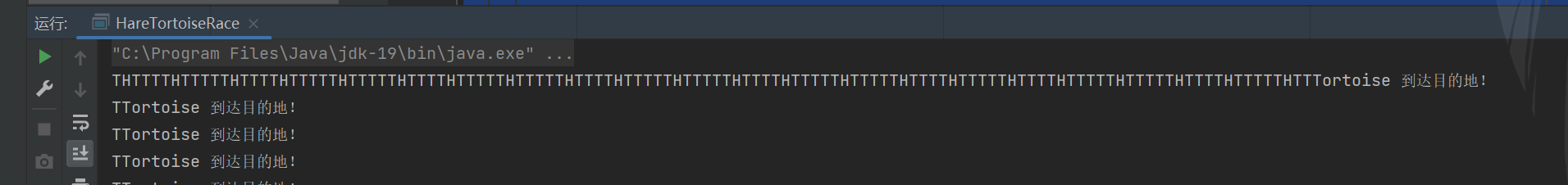
班级：2021级计算机科学与技术二班

1. 程序补全题

* 下面程序用多线程演示龟兔赛跑，兔子的速度是乌龟速度的3倍，但每次休息时间是乌龟的5倍，请填空使程序完整
* 源代码：

|  |
| --- |
| package ExperimentSourceCode\_5**.**fill**;**  class Tortoise **extends** Thread **{**  int sleepTime**=**0**,**targetLength**=**100**;**  Tortoise**(**int sleepTime**,**int targetLength**){**  **this.**sleepTime**=**sleepTime**;**  **this.**targetLength**=**targetLength**;**  **this.**setName**(**"Tortoise "**);** // 设置线程的名字为Tortoise  **}**  public void run**()** **{**  **while(true){**  targetLength**=**targetLength**-**1**;**  System**.**out**.**print**(**"T"**);**  **try** **{**  sleep**(**sleepTime**);** // 让线程调用 sleep方法进入休息状态，  // sleepTime毫秒后线程重新排队，等待CPU资源  **}**  **catch(**InterruptedException e**)** **{}**  **if(**targetLength**<=**0**){**  System**.**out**.**printf**(**getName**()+**"到达目的地！\n"**);**  interrupt**();** // 结束线程  **}**  **}**  **}**  **}**  class Hare **extends** Thread **{**  int sleepTime**=**0**,**targetLength**=**100**;**  Hare**(**int sleepTime**,**int targetLength**){**  **this.**sleepTime**=**sleepTime**;**  **this.**targetLength**=**targetLength**;**  setName**(**"Hare"**);** // 设置线程的名字为Hare  **}**  public void run**()** **{**  **while(true)** **{**  targetLength**=**targetLength**-**3**;**  System**.**out**.**printf**(**"H"**);**  **try** **{**  sleep**(**sleepTime**);** // 让线程调用 sleep方法进入中断状态，  //sleepTime毫秒后 线程重新排队，等待CUP资源  **}catch(**InterruptedException e**)** **{** **}**  **if(**targetLength**<=**0**)** **{**  System**.**out**.**printf**(**getName**()+**"到达目的地！\n"**);**  interrupt**();**  **}**  **}**  **}}**  public class HareTortoiseRace  **{**  public static void main**(**String args**[** **])**  **{**  Hare hare**;**  hare**=new** Hare**(**500**,**100**);** // 新建线程hare  Tortoise tortoise**;**  tortoise**=new** Tortoise**(**100**,**100**);** // 新建线程tortoise  hare**.**start**();** // 启动线程hare  tortoise**.**start**();** // 启动线程tortoise  **}**  **}** |

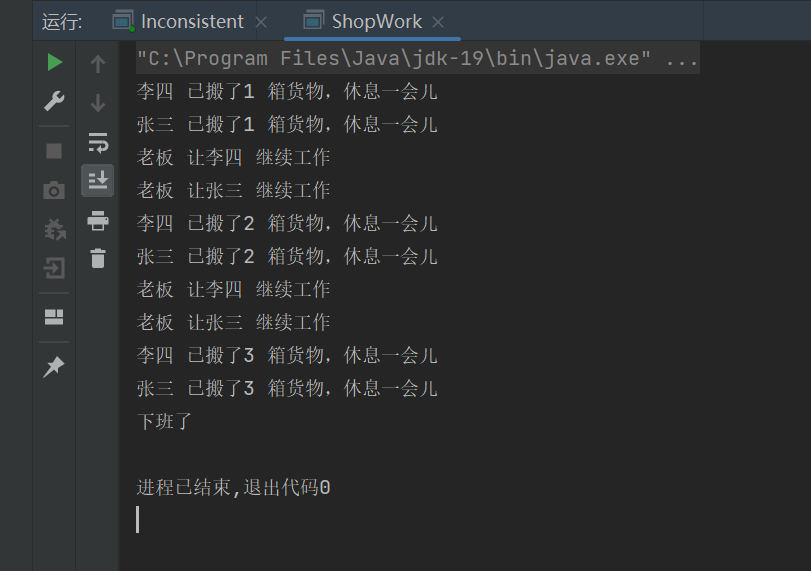
* 运行结果：



* 下面程序演示线程休息和中断，请填空使程序完整
* 源代码：

|  |
| --- |
| package ExperimentSourceCode\_5**.**fill**;**  class ShopWorker **implements** Runnable  **{**  static Thread zhangSan**,**liSi**,**boss**;**  ShopWorker**()**  **{**  boss**=new** Thread**(this);** // 创建boss对象  zhangSan**=new** Thread**(this);** // 创建zhangSan对象  liSi**=new** Thread**(this);** // 创建liSi对象  zhangSan**.**setName**(**"张三"**);**  liSi**.**setName**(**"李四"**);**  boss**.**setName**(**"老板"**);**  **}**  public void run**()** **{**  int i**=**0**;**  **if(**Thread**.**currentThread**()==**zhangSan**){**  **while(true){**  **try{**  i**++;**  System**.**out**.**println**(**Thread**.**currentThread**().**getName**()+**" 已搬了"**+**i**+**" 箱货物，休息一会儿"**);**  **if(**i**==**3**)** **return;**  Thread**.**sleep**(**10000**);** // zhangSan睡眠10秒（10000毫秒）  **}catch(**InterruptedException e**){**  System**.**out**.**println**(**boss**.**getName**()+**" 让"**+**Thread**.**currentThread**().**getName**()+**" 继续工作"**);**  **}**  **}**  **}else** **if(**Thread**.**currentThread**()==**liSi**){**  **while(true){**  **try{**  i**++;**  System**.**out**.**println**(**Thread**.**currentThread**().**getName**()+**" 已搬了"**+**i**+**" 箱货物，休息一会儿"**);**  **if(**i**==**3**)** **return;**  Thread**.**sleep**(**10000**);** // liSi睡眠10秒（10000毫秒）  **}catch(**InterruptedException e**){**  System**.**out**.**println**(**boss**.**getName**()+**" 让"**+**Thread**.**currentThread**().**getName**()+**" 继续工作"**);**  **}**  **}**  **}else** **if(**Thread**.**currentThread**()==**boss**){**  **while(true){**  zhangSan**.**interrupt**();** // 吵醒zhangSan  liSi**.**interrupt**();** // 吵醒liSi  **if(!(**zhangSan**.**isAlive**()||**liSi**.**isAlive**())){**  System**.**out**.**println**(**"下班了"**);**  **return;**  **}**  **}**  **}**  **}**  **}**  class ShopWork  **{**  public static void main**(**String args**[])**  **{**  ShopWorker shop**=new** ShopWorker**();**  shop**.**zhangSan**.**start**();**  shop**.**liSi**.**start**();**  shop**.**boss**.**start**();**  **}**  **}** |

* 运行结果：



* 已经练习了许多独立运行的线程，但在实际编程中，经常会有许多线程需要访问同一个资源或者数据的情况，有可能两个线程会同时修改同一个资源对象，这样就会出现资源不一致的情况，在主教材中已经讲解和演示了这个问题。但由于Java中的内存模型，在使用多线程的应用程序中，有时一个线程看不到另一个线程所做的修改，这就需要对数据的操作使用同步规则才能解决。如下面程序的演示，类Inconsistent演示了主线程看不到子线程的修改，请填空完成Consistent类的同步处理。
* 源代码：

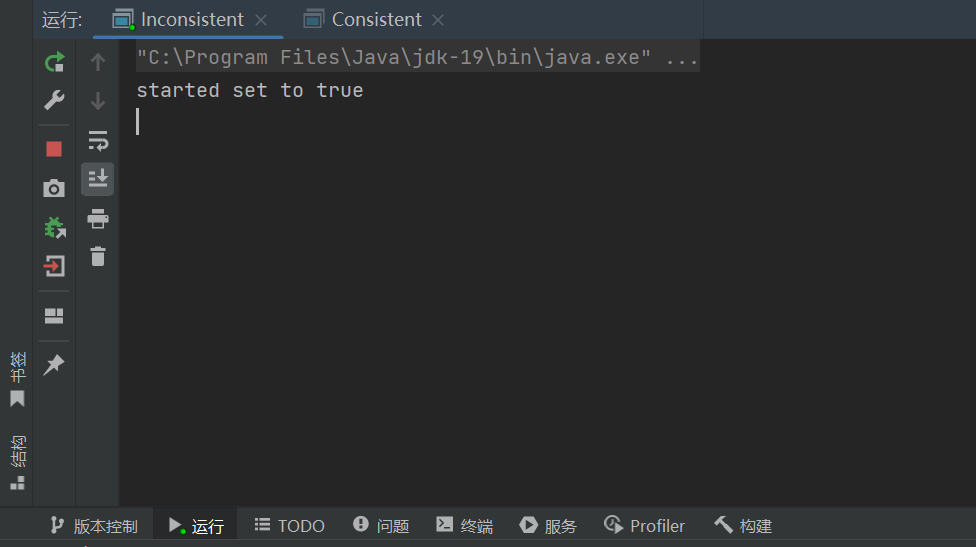
(1)Inconsistent.java

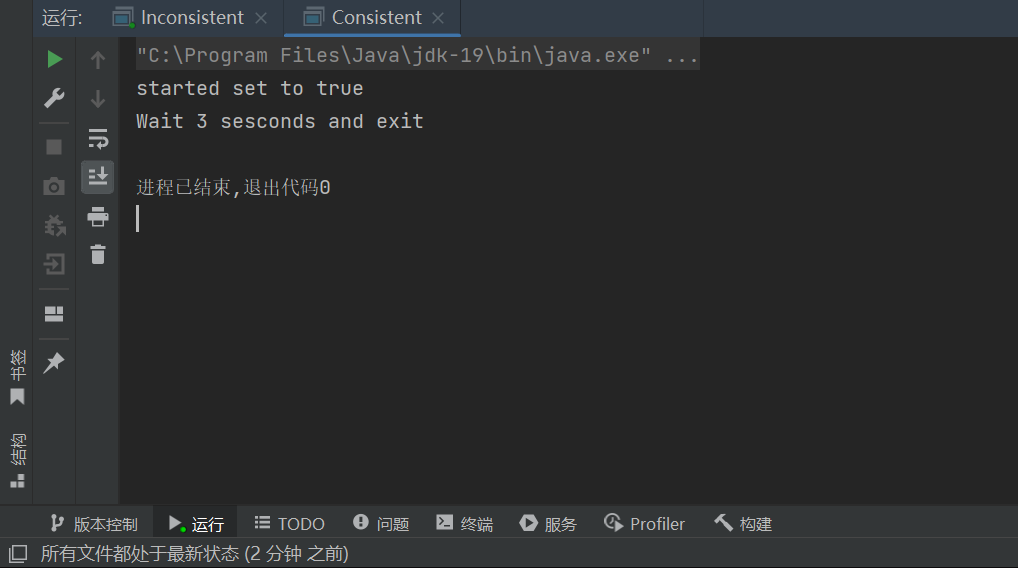
|  |
| --- |
| package ExperimentSourceCode\_5**.**fill**;**  //Inconsistent.java  public class Inconsistent **{**  static boolean started**=false;**  public static void main**(**String**[]** args**){**  Thread thread1**=new** Thread**(new** Runnable**(){**  public void run**(){**  **try** **{**  Thread**.**sleep**(**3000**);**  **}catch(**InterruptedException e**){**  **}**  System**.**out**.**println**(**"started set to true"**);**  started**=true;**  **}**  **});**  thread1**.**start**();**  **while(!**started**){**  **}**  System**.**out**.**println**(**"Wait 3 seconds and exit"**);**  **}**  **}** |

(2) Consistent.java

|  |
| --- |
| package ExperimentSourceCode\_5**.**fill**;**  // Consistent.java  public class Consistent **{**  static boolean started **=** **false;**  public synchronized static void setStarted**(){**  started **=** **true;**  **}**  public synchronized static boolean getStarted**(){**  **return** started**;**  **}**  public static void main**(**String args**[]){**  Thread thread1 **=** **new** Thread**(new** Runnable**()** **{**  public void run**()** **{**  **try{**  Thread**.**sleep**(**3000**);**  **}catch(**InterruptedException e**){**  **}**  setStarted**();**  System**.**out**.**println**(**"started set to true"**);**  **}**  **});**  thread1**.**start**();**  **while(!**getStarted**()){}**  System**.**out**.**println**(**"Wait 3 sesconds and exit"**);**  **}**  **}** |

* 运行结果：

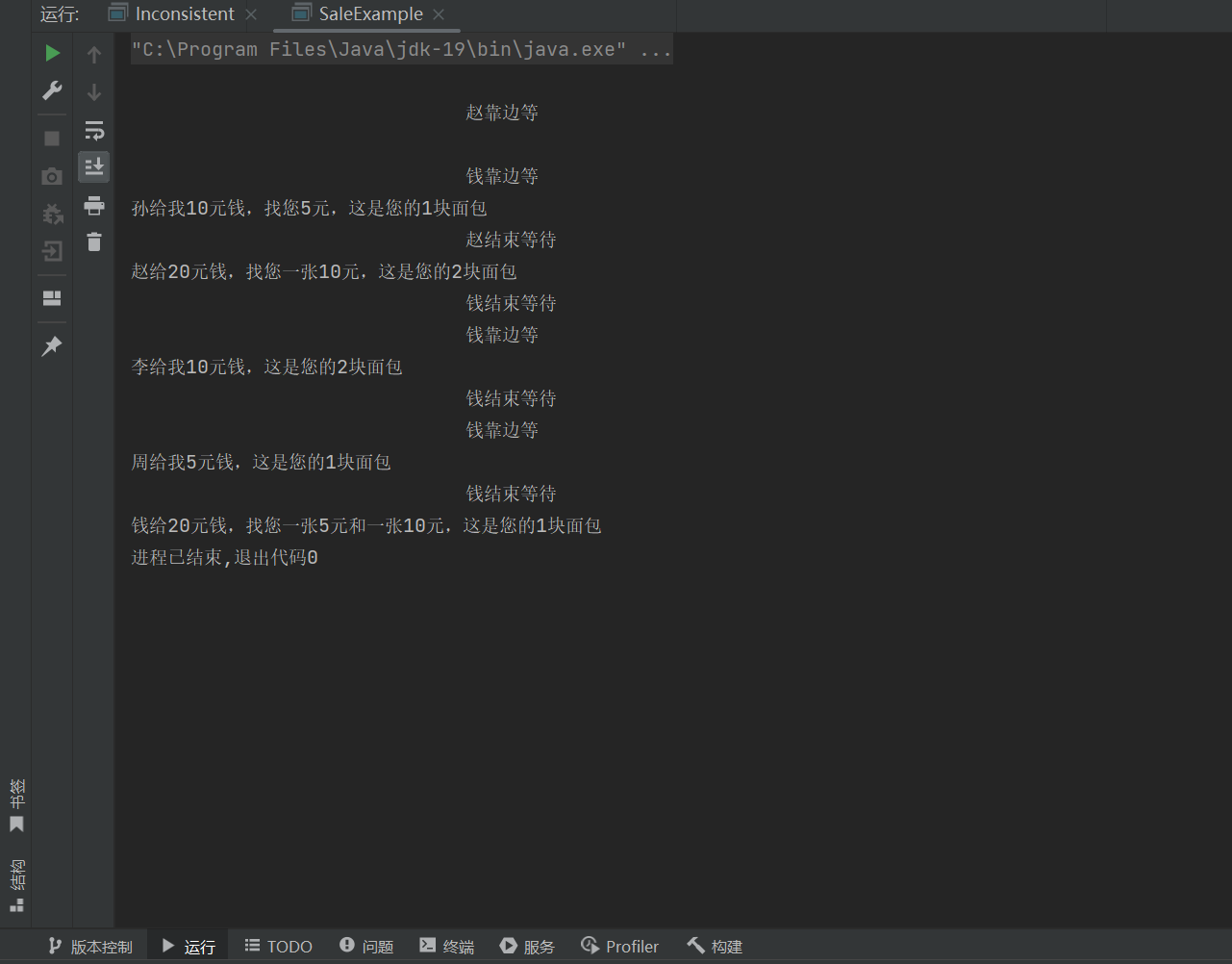




* 线程间通信。Java提供了一套简单易用的线程问通信机制，使用wit()、notfy(orlyAl()方法来进行通信，以下程序演示了线程间的通信技巧，请填空完成程序
* 源代码：

|  |
| --- |
| package ExperimentSourceCode\_5**.**fill**;**  class BreadSeller **{** //负责卖面包的类，一块面包5元钱。  int fiveNum**=**1**,**tenNum**=**0**,**twentyNum**=**0**;** // 面包师傅现有钱各面额的数量  public synchronized void sellBread**(**int receiveMoney**,**int buyNumber**){**  **if(**receiveMoney**==**5**){**  fiveNum**=**fiveNum**+**1**;**  System**.**out**.**printf**(**"\n%s给我5元钱，这是您的1块面包"**,**Thread**.**currentThread**().**getName**());**  **}else** **if(**receiveMoney**==**10**&&**buyNumber**==**2**){**  tenNum**=**tenNum**+**1**;**  System**.**out**.**printf**(**"\n%s给我10元钱，这是您的2块面包"**,**  Thread**.**currentThread**().**getName**());**  **}else** **if(**receiveMoney**==**10**&&**buyNumber**==**1**)** **{**  **while(**fiveNum**<**1**){**  **try** **{**  System**.**out**.**printf**(**"\n%30s靠边等"**,**Thread**.**currentThread**().**getName**());**  wait**();** //如果线程占有CPU期间执行了wait,就进入等待状态。  System**.**out**.**printf**(**"\n%30s结束等待\n"**,**Thread**.**currentThread**().**getName**());**  **}catch(**InterruptedException e**){}**  **}**  fiveNum**=**fiveNum**-**1**;**  tenNum**=**tenNum**+**1**;**  System**.**out**.**printf**(**"\n%s给我10元钱，找您5元，这是您的1块面包"**,**Thread**.**currentThread**().**getName**());**  **}else** **if(**receiveMoney**==**20**&&**buyNumber**==**1**){**  **while((**fiveNum**<**1**||**tenNum**<**1**)&&** **!(**fiveNum**>**3**)){**  **try** **{**  System**.**out**.**printf**(**"\n%30s靠边等"**,**Thread**.**currentThread**().**getName**());**  wait**();** //如果线程占有CPU期间执行了wait,就进入中断状态。  System**.**out**.**printf**(**"\n%30s结束等待"**,**Thread**.**currentThread**().**getName**());**  **}catch(**InterruptedException e**){}**  **}**  **if(**fiveNum**>**3**){**  fiveNum**=**fiveNum**-**3**;**  twentyNum**=**twentyNum**+**1**;**  System**.**out**.**printf**(**"\n%s给20元钱，找您三张5元，这是您的1块面包"**,**Thread**.**currentThread**().**getName**());**  **}else{**  fiveNum**=**fiveNum**-**1**;**  tenNum**=**tenNum**-**1**;**  twentyNum**=**twentyNum**+**1**;**  System**.**out**.**printf**(**"\n%s给20元钱，找您一张5元和一张10元，这是您的1块面包"**,**Thread**.**currentThread**().**getName**());**  **}**  **}else** **if(**receiveMoney**==**20**&&**buyNumber**==**2**){**  **while(**tenNum**<**1**&&**fiveNum**<**2**){**  **try** **{**  System**.**out**.**printf**(**"\n%30s靠边等\n"**,**Thread**.**currentThread**().**getName**());**  wait**();**  System**.**out**.**printf**(**"\n%30s结束等待"**,**Thread**.**currentThread**().**getName**());**  **}catch(**InterruptedException e**){}**  **}**  **if(**fiveNum**<**2**){**  tenNum**=**tenNum**-**1**;**  twentyNum**=**twentyNum**+**1**;**  System**.**out**.**printf**(**"\n%s给20元钱，找您一张10元，这是您的2块面包"**,**Thread**.**currentThread**().**getName**());**  **}**  **}else{**  fiveNum**=**fiveNum**-**2**;**  twentyNum**=**twentyNum**+**1**;**  System**.**out**.**printf**(**"\n%s给20元钱，找您两张5元，这是您的2块面包"**,**Thread**.**currentThread**().**getName**());**  **}**  notifyAll**();**  **}**  **}**  class Breadshop **implements** Runnable**{** //实现Runnable接口的类（面包店）。  Thread zhao**,**qian**,**sun**,**li**,**zhou**;**//来面包店中买面包的线程。  BreadSeller seller**;** //面包店服务员  Breadshop**(){**  zhao**=new** Thread**(this);**  qian**=new** Thread**(this);**  sun**=new** Thread**(this);**  li**=new** Thread**(this);**  zhou**=new** Thread**(this);**  zhao**.**setName**(**"赵"**);**  qian**.**setName**(**"钱"**);**  sun**.**setName**(**"孙"**);**  li**.**setName**(**"李"**);**  zhou**.**setName**(**"周"**);**  seller**=new** BreadSeller**();**  **}**  public void run**(){**  **if(**Thread**.**currentThread**()==**zhao**){**  seller**.**sellBread**(**20**,**2**);**  **}else** **if(**Thread**.**currentThread**()==**qian**){**  seller**.**sellBread**(**20**,**1**);**  **}else** **if(**Thread**.**currentThread**()==**sun**){**  seller**.**sellBread**(**10**,**1**);**  **}else** **if(**Thread**.**currentThread**()==**li**){**  seller**.**sellBread**(**10**,**2**);**  **}else** **if(**Thread**.**currentThread**()==**zhou**){**  seller**.**sellBread**(**5**,**1**);**  **}**  **}**  **}**  public class SaleExample**{**  public static void main**(**String args**[]){**  Breadshop myshop**=new** Breadshop**();**  myshop**.**zhao**.**start**();**  **try** **{** Thread**.**sleep**(**1000**);**  **}catch(**InterruptedException e**){}**  myshop**.**qian**.**start**();**  **try** **{** Thread**.**sleep**(**1000**);**  **}** **catch(**InterruptedException e**){}**  myshop**.**sun**.**start**();**  **try** **{** Thread**.**sleep**(**1000**);**  **}** **catch(**InterruptedException e**){}**  myshop**.**li**.**start**();**  **try** **{** Thread**.**sleep**(**1000**);**  **}** **catch(**InterruptedException e**){}**  myshop**.**zhou**.**start**();**  **}**  **}** |

* 运行结果：



1. 程序设计题：设计一个哲学家就餐问题的程序。
2. 问题描述：

哲学家就餐问题是在计算机科学中的一个经典问题，用来演示在并行计算中多线程同步产生的问题。哲学家就餐问题可以这样表述，假设有五位哲学家围坐在一张圆形餐桌旁，做以下两件事情之一：吃饭或思考。如果他们在吃东西时，则停止思考；如果他们在思考时，则停止吃东西。餐桌上每人有一大碗意大利面，每两人之间有一根筷子，如图9-4所示。由于用一根子无法吃到意大利面，他们只能同时拿到自己左右两边的筷子才能吃。请抽象建模并利用Java中的线程来模拟这一过程。

1. 建模：哲学家拿筷子，其实就是线程使用共用资源，而无法吃到面的情况就是使用同步锁时发生了死锁。

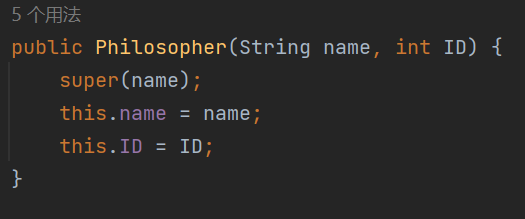
* 哲学家：五个线程
* 筷子：公共资源，每个筷子都有一个锁
* 解决问题：死锁

1. 解决思路：哲学家问题，一个很经典的死锁问题，死锁是一种由于两个锁嵌套导致程序无法进行错误，解决常见有两种思路：

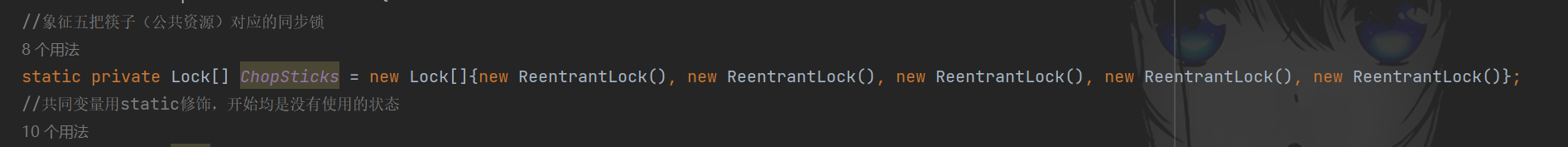
* **避免锁的嵌套**：如在本问题，让哲学家必须一次拿到两只筷子，也就是让这原本的两个锁必须同时触发，即两个锁变成一个锁，不存在锁的嵌套。
* **避免死锁状态的出现：**在本问题中的死锁状态是五个哲学家每个人手上都有一只筷子，我们只要避免这种状态的出现即可：可以增加一个服务员的角色，在死锁状态发生时，让一个哲学家放下筷子；可以改变哲学家拿筷子的顺序，增加一个“左撇子”，拿筷子的顺序就一定存在冲突，有哲学家需要等待，进而就不可能出现都拿到筷子的情况。

1. **具体建模实现：**

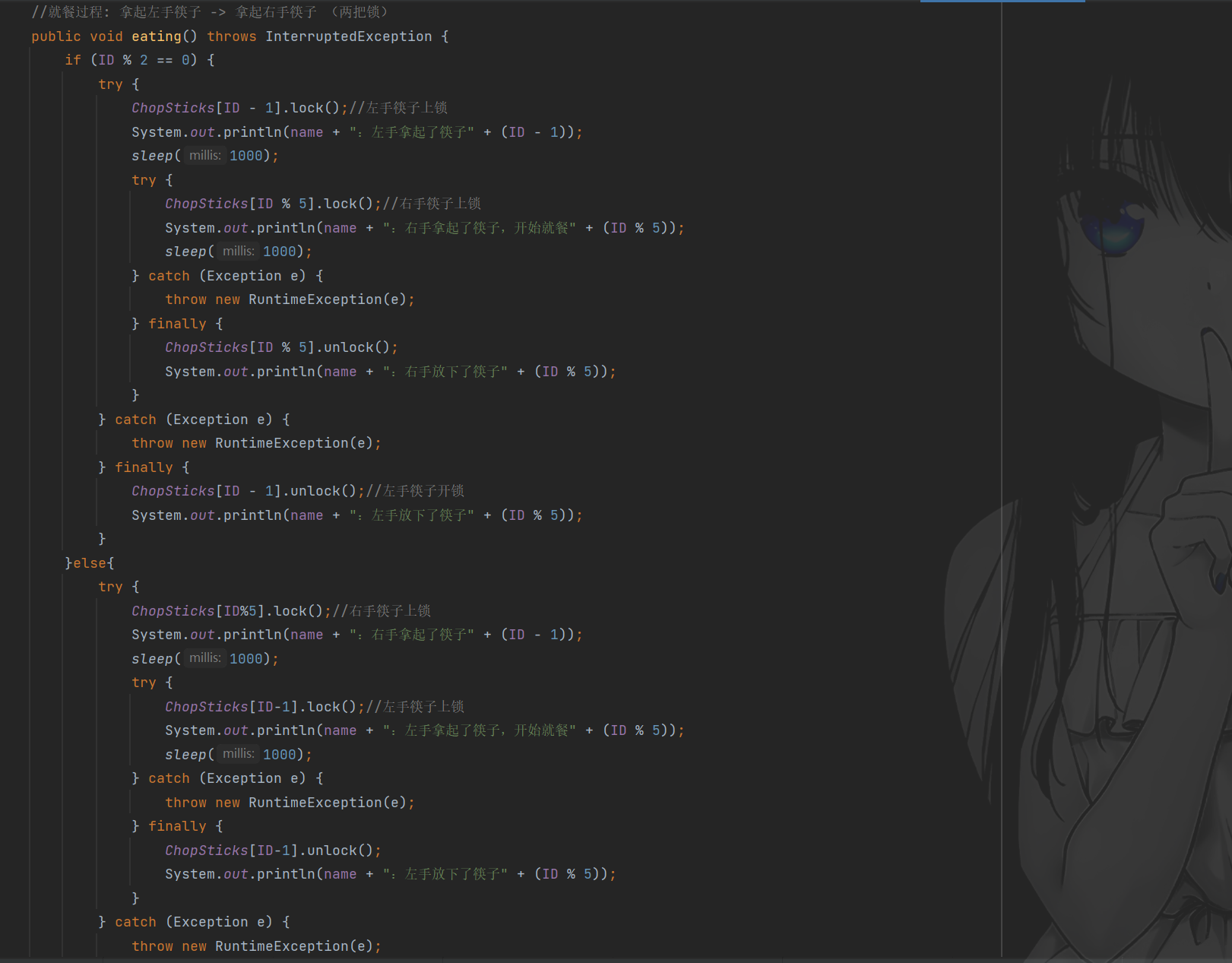
* 哲学家：即线程，采用继承Thread的方式创建线程，注意哲学家应该有ID，代表他们坐的顺序，对应他们使用的两双筷子。



* 筷子： 线程间的公共资源，定义五个锁代表它们的使用权，注意锁必须用static修饰。



* 具体过程：奇数编号哲学家优先使用右手筷子，偶数编号哲学家优先使用左手筷子，相邻两筷子编号：ID%5、ID-1。



1. 源码：

* Philosopher.java

|  |
| --- |
| package ExperimentSourceCode\_5**.**Philosopher\_dining**;**  **import** java**.**util**.**ArrayList**;**  **import** java**.**util**.**Arrays**;**  **import** java**.**util**.**concurrent**.**locks**.**Lock**;**  **import** java**.**util**.**concurrent**.**locks**.**ReentrantLock**;**  **import** static java**.**lang**.**Thread**.**sleep**;**  /\*\*  \* 哲学家类 => 进程：思考 -> 吃饭  \* 死锁状态：每个哲学家手中都有一根筷子，都无法拿到另一根筷子  \*/  public class Philosopher **extends** Thread **{**  //象征五把筷子（公共资源）对应的同步锁  static private Lock**[]** ChopSticks **=** **new** Lock**[]{new** ReentrantLock**(),** **new** ReentrantLock**(),** **new** ReentrantLock**(),** **new** ReentrantLock**(),** **new** ReentrantLock**()};**  //共同变量用static修饰，开始均是没有使用的状态  private String name**;**  private int ID**;**  public Philosopher**(**String name**,** int ID**)** **{**  **super(**name**);**  **this.**name **=** name**;**  **this.**ID **=** ID**;**  **}**  @Override  public void run**()** **{**  //System.out.println(name+" "+ID);  **try** **{**  thinking**();**  eating**();**  **}** **catch** **(**InterruptedException e**)** **{**  **throw** **new** RuntimeException**(**e**);**  **}**  **}**  //思考  public void thinking**()** **throws** InterruptedException **{**  System**.**out**.**println**(**name **+** ":正在思考"**);**  sleep**(**1000**);**  **}**  //就餐过程: 拿起左手筷子 -> 拿起右手筷子 （两把锁）  public void eating**()** **throws** InterruptedException **{**  **if** **(**ID **%** 2 **==** 0**)** **{**  **try** **{**  ChopSticks**[**ID **-** 1**].**lock**();**//左手筷子上锁  System**.**out**.**println**(**name **+** "：左手拿起了筷子" **+** **(**ID **-** 1**));**  sleep**(**1000**);**  **try** **{**  ChopSticks**[**ID **%** 5**].**lock**();**//右手筷子上锁  System**.**out**.**println**(**name **+** "：右手拿起了筷子，开始就餐" **);**  sleep**(**1000**);**  **}** **catch** **(**Exception e**)** **{**  **throw** **new** RuntimeException**(**e**);**  **}** **finally** **{**  ChopSticks**[**ID **%** 5**].**unlock**();**  System**.**out**.**println**(**name **+** "：右手放下了筷子" **+** **(**ID **%** 5**));**  **}**  **}** **catch** **(**Exception e**)** **{**  **throw** **new** RuntimeException**(**e**);**  **}** **finally** **{**  ChopSticks**[**ID **-** 1**].**unlock**();**//左手筷子开锁  System**.**out**.**println**(**name **+** "：左手放下了筷子" **+** **(**ID **%** 5**));**  **}**  **}else{**  **try** **{**  ChopSticks**[**ID**%**5**].**lock**();**//右手筷子上锁  System**.**out**.**println**(**name **+** "：右手拿起了筷子" **+** **(**ID **-** 1**));**  sleep**(**1000**);**  **try** **{**  ChopSticks**[**ID**-**1**].**lock**();**//左手筷子上锁  System**.**out**.**println**(**name **+** "：左手拿起了筷子，开始就餐" **);**  sleep**(**1000**);**  **}** **catch** **(**Exception e**)** **{**  **throw** **new** RuntimeException**(**e**);**  **}** **finally** **{**  ChopSticks**[**ID**-**1**].**unlock**();**  System**.**out**.**println**(**name **+** "：左手放下了筷子" **+** **(**ID **%** 5**));**  **}**  **}** **catch** **(**Exception e**)** **{**  **throw** **new** RuntimeException**(**e**);**  **}** **finally** **{**  ChopSticks**[**ID**%**5**].**unlock**();**//右手筷子开锁  System**.**out**.**println**(**name **+** "：右手放下了筷子" **+** **(**ID **%** 5**));**  **}**  **}**  **}**  **}** |

* PhilosopherDining.java

|  |
| --- |
| package ExperimentSourceCode\_5**.**Philosopher\_dining**;**  public class PhilosopherDining **{**  /\*\*  \* 五个哲学家即五个线程，五双筷子即五个线程间的共同资源  \* =>线程同步机制  \* =>处理死锁  \*/  public static void main**(**String**[]** args**)** **{**  Philosopher t1 **=** **new** Philosopher**(**"哲学家1"**,**1**);**  Philosopher t2 **=** **new** Philosopher**(**"哲学家2"**,**2**);**  Philosopher t3 **=** **new** Philosopher**(**"哲学家3"**,**3**);**  Philosopher t4 **=** **new** Philosopher**(**"哲学家4"**,**4**);**  Philosopher t5 **=** **new** Philosopher**(**"哲学家5"**,**5**);**  t1**.**start**();**  t2**.**start**();**  t3**.**start**();**  t4**.**start**();**  t5**.**start**();**  **}**  **}** |

1. 实验结果：

