【案例8-5】 小朋友就餐问题

### 【案例介绍】

**1.任务描述**

一圆桌前坐着5位小朋友，两个人中间有一只筷子，桌子中央有面条。小朋友边吃边玩，当饿了的时候拿起左右两只筷子吃饭，必须拿到两只筷子才能吃饭。但是，小朋友在吃饭过程中，可能会发生5个小朋友都拿起自己右手边的筷子，这样每个小朋友都因缺少左手边的筷子而没有办法吃饭。本案例要求编写一个程序解决小朋友就餐问题，使每个小朋友都能成功就餐。

**2.运行结果**

运行结果如图8-1所示。

手机屏幕截图

描述已自动生成

图8-1 运行结果

### 【案例目标】

* 学会分析”小朋友就餐问题”任务实现的逻辑思路。
* 能够独立完成”小朋友就餐问题”程序的源代码编写、编译以及运行。
* 通过”小朋友就餐问题”程序理解多线程安全问题的方式原因，并掌握如果解决多线程安全问题。

### 【案例思路】

1. 查看运行结果分析后，每个小朋友相当于一个线程，所以先创建一个Philosopher()方法作为小朋友。
2. 查看运行结果分析后，创建eating()方法作为小朋友吃饭时的线程，创建thinking()方法作为小朋友玩耍是的线程。
3. 查看运行结果分析后，需要在获取筷子的方法Fork中先定义一个boolean类型的数组，代表5根筷子的使用情况；再使用synchronized线程锁来控制只有左右手的筷子都未被使用时，才允许获取筷子，且必须同时获取左右手筷子。
4. 查看运行结果分析后，需要在释放左右手筷子的方法putFork中使用synchronized线程锁来释放筷子。
5. 最后在Test测试类中调用5次以上方法，代表5位小朋友。

### 【案例代码】

小朋友就餐问题的程序代码实现如文件8-1所示。

文件8-1 Philosopher.java

1. package chapter04061;
2. /\*每个小朋友相当于一个线程\*/
3. public class Philosopher extends Thread{
4. private String name;
5. private Fork fork;
6. public Philosopher(String name,Fork fork){
7. super(name);
8. this.name=name;
9. this.fork=fork;
10. }
11. public void run(){
12. while(true){
13. thinking();
14. fork.takeFork();
15. eating();
16. fork.putFork();
17. }
18. }
19. public void eating(){
20. System.out.println("小朋友"+name+"在吃饭");
21. try {
22. sleep(1000);//模拟吃饭，占用一段时间资源
23. } catch (InterruptedException e) {
24. // TODO Auto-generated catch block
25. e.printStackTrace();
26. }
27. }
28. public void thinking(){
29. System.out.println("小朋友"+name+"在玩游戏");
30. try {
31. sleep(1000);//模拟思考
32. } catch (InterruptedException e) {
33. // TODO Auto-generated catch block
34. e.printStackTrace();
35. }
36. }
37. }
38. class Fork{
39. /\*5只筷子，初始为都未被用\*/
40. private boolean[] used={false,false,false,false,false,false};
41. /\*只有当左右手的筷子都未被使用时，才允许获取筷子，且必须同时获取左右手筷子\*/
42. public synchronized void takeFork(){
43. String name = Thread.currentThread().getName();
44. int i = Integer.parseInt(name);
45. while(used[i]||used[(i+1)%5]){
46. try {
47. wait();//如果左右手有一只正被使用，等待
48. } catch (InterruptedException e) {
49. // TODO Auto-generated catch block
50. e.printStackTrace();
51. }
52. }
53. used[i ]= true;
54. used[(i+1)%5]=true;
55. }
56. /\*必须同时释放左右手的筷子\*/
57. public synchronized void putFork(){
58. String name = Thread.currentThread().getName();
59. int i = Integer.parseInt(name);
60. used[i]= false;
61. used[(i+1)%5]=false;
62. notifyAll();//唤醒其他线程
63. }
64. }

文件8-1中第3~18行代码封装一个小朋友的方法，第19~27行代码是封装了小朋友吃饭时的方法，第28~37行代码封装了小朋友玩耍时的方法，第28~64行代码封装了筷子使用情况的方法。

测试类的代码如文件8-2所示，调用5次Fork代表5个小朋友。

文件8-2 Test.java

1. package chapter04061;
2. public class Test {
3. public static void main(String []args){
4. Fork fork = new Fork();
5. new Philosopher("0",fork).start();
6. new Philosopher("1",fork).start();
7. new Philosopher("2",fork).start();
8. new Philosopher("3",fork).start();
9. new Philosopher("4",fork).start();
10. }
11. }