congmingyige

哲学家问题(java)的三个解法

```java
//加synchronize进行同步
//释放资源又很快获得自身的资源，这样不妥，吃完的话休息100ms

//每个人先申请编号小的筷子

public class Philosopher implements Runnable {
    int[] fork=new int[5];
    Thread thread1=new Thread(this,"1");
    Thread thread2=new Thread(this,"2");
    Thread thread3=new Thread(this,"3");
    Thread thread4=new Thread(this,"4");
    Thread thread5=new Thread(this,"5");
    public void run() {
        try {
            while (true) {
                if (Thread.currentThread().getName().equals("1")) {
                    while (fork[0]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[0]=1;
                    while (fork[1]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[1]=1;
                    System.out.println("1 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[0]=0;
                    fork[1]=0;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
                else if (Thread.currentThread().getName().equals("2")) {
                    while (fork[1]==1) {
                        synchronized(this) {
                            wait();
                        }
                    }
                    fork[1]=1;
                    while (fork[2]==1) {
                        synchronized(this) {
                            wait();
                        }
                    }
                    fork[2]=1;
                    System.out.println("2 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[1]=0;
                    fork[2]=0;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
                else if (Thread.currentThread().getName().equals("3")) {
                    while (fork[2]==1) {
                        synchronized(this) {
                            wait();
                        }
```

公告

昵称：congmingyige
园龄：1年5个月
粉丝：2
关注：1

<			2018年6月			>
日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

编程注意(21)
新思想(17)
易错(17)
天梯赛(13)
题解(12)
证明(9)
word(8)
比对算法(7)
斐波那契数列(7)
汇编(7)
更多

随笔档案

2018年5月 (4)
2018年4月 (35)
2018年3月 (12)
2017年12月 (1)
2017年10月 (1)
2017年9月 (4)
2017年8月 (1)
2017年7月 (22)

```
                    }
                    fork[2]=1;
                    while (fork[3]==1) {
                        synchronized(this) {
                            wait();
                        }
                    }
                    fork[3]=1;
                    System.out.println("3 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[2]=0;
                    fork[3]=0;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
                else if (Thread.currentThread().getName().equals("4")) {
                    while (fork[3]==1) {
                        synchronized(this) {
                            wait();
                        }
                    }
                    fork[3]=1;
                    while (fork[4]==1) {
                        synchronized(this) {
                            wait();
                        }
                    }
                    fork[4]=1;
                    System.out.println("4 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[3]=0;
                    fork[4]=0;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
                else if (Thread.currentThread().getName().equals("5")) {
                    while (fork[0]==1) {
                        synchronized(this) {
                            wait();
                        }
                    }
                    fork[0]=1;
                    while (fork[4]==1) {
                        synchronized(this) {
                            wait();
                        }
                    }
                    fork[4]=1;
                    System.out.println("5 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[0]=0;
                    fork[4]=0;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Philosopher phi=new Philosopher();
        for (int i=0;i<5;i++)
            phi.fork[i]=0;
        phi.thread1.start();
        phi.thread2.start();
        phi.thread3.start();
        phi.thread4.start();
        phi.thread5.start();
    }
}
```

2017年6月 (9)
2017年5月 (31)
2017年4月 (29)
2017年3月 (28)
2016年12月 (3)

最新评论

1. Re:牛客练习赛16
@hk_lin添加了一些证明...
--congmingyige
2. Re:牛客练习赛16
@hk_lin是我写错了，已修改，见上文...
--congmingyige
3. Re:牛客练习赛16
D为什么是这样的啊，有没有啥定理之类的0 0
--hk_lin
4. Re:SenseTime Ace Coder Challenge 暨 商汤在线编程挑战赛 D. 白色相簿
@日常膜大佬~太强了1249591860@qq.com谢谢了！...
--congmingyige
5. Re:SenseTime Ace Coder Challenge 暨 商汤在线编程挑战赛 D. 白色相簿
没有网址
--日常膜大佬~太强了

阅读排行榜

1. word中批量修改图片大小的两个方法(5832)
2. 波利亚罐子模型证明(731)
3. 网易2017秋招编程题——回文序列 解题报告(667)
4. codeblocks下的汇编语言(625)
5. csp20170304地铁修建_Solution(523)

评论排行榜

1. SenseTime Ace Coder Challenge 暨 商汤在线编程挑战赛 D. 白色相簿(11)
2. 牛客练习赛16(3)
3. 团体程序设计天梯赛 L1-011. A-B(2)
4. 团体程序设计天梯赛 L1-010. 比较大小(1)
5. 团体程序设计天梯赛 L1-009. N个数求和(1)

推荐排行榜

1. 网易2017秋招编程题——回文序列 解题报告(1)
2. ZOJ_3950_How Many Nines 解题报告及如何对程序进行测试修改(1)

```
//当某个线程试图等待一个自己并不拥有的对象（O）的监控器或者通知其他线程等待该对象（O）的监控器时，抛出该异常。
```

```
//让刚吃完的一个人阻塞，5根筷子供4个人选，则必有一个人获得在其左右的两双筷子

public class Philosopher1 implements Runnable {
    int[] ifeat=new int[5];
    int[] fork=new int[5];
    int noteat;
    Thread thread1=new Thread(this,"1");
    Thread thread2=new Thread(this,"2");
    Thread thread3=new Thread(this,"3");
    Thread thread4=new Thread(this,"4");
    Thread thread5=new Thread(this,"5");
    public void run() {
        try {
            while (true) {
                if (Thread.currentThread().getName().equals("1")) {
                    while (ifeat[0]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    while (fork[0]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[0]=1;
                    while (fork[1]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[1]=1;
                    System.out.println("1 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[0]=0;
                    fork[1]=0;
                    ifeat[noteat]=0;
                    noteat=0;
                    ifeat[0]=1;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
                else if (Thread.currentThread().getName().equals("2")) {
                    while (ifeat[1]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    while (fork[1]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[1]=1;
                    while (fork[2]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[2]=1;
                    System.out.println("2 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[1]=0;
                    fork[2]=0;
                    ifeat[noteat]=0;
                    noteat=1;
                    ifeat[1]=1;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
                else if (Thread.currentThread().getName().equals("3")) {
                    while (ifeat[2]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    while (fork[2]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[2]=1;
                    while (fork[3]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[3]=1;
                    System.out.println("3 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[2]=0;
                    fork[3]=0;
                    ifeat[noteat]=0;
                    noteat=2;
                    ifeat[2]=1;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
                else if (Thread.currentThread().getName().equals("4")) {
                    while (ifeat[3]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    while (fork[3]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[3]=1;
                    while (fork[4]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[4]=1;
                    System.out.println("4 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[3]=0;
                    fork[4]=0;
                    ifeat[noteat]=0;
                    noteat=3;
                    ifeat[3]=1;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
                else if (Thread.currentThread().getName().equals("5")) {
                    while (ifeat[4]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    while (fork[4]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[4]=1;
                    while (fork[0]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[0]=1;
                    System.out.println("5 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[4]=0;
                    fork[0]=0;
                    ifeat[noteat]=0;
                    noteat=4;
                    ifeat[4]=1;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
            }
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Philosopher1 phi=new Philosopher1();
        for (int i=0;i<5;i++)
            phi.fork[i]=0;

        phi.ifeat[0]=1;
        for (int i=0;i<5;i++)
            phi.ifeat[i]=0;
        phi.noteat=0;

        phi.thread1.start();
        phi.thread2.start();
        phi.thread3.start();
        phi.thread4.start();
        phi.thread5.start();
    }
}
```

```
//只有两双筷子都有，才获取，且同时获取两双筷子

public class Philosopher2 implements Runnable {
    int[] fork=new int[5];
    Thread thread1=new Thread(this,"1");
    Thread thread2=new Thread(this,"2");
    Thread thread3=new Thread(this,"3");
    Thread thread4=new Thread(this,"4");
    Thread thread5=new Thread(this,"5");
    public void run() {
        try {
            while (true) {
                if (Thread.currentThread().getName().equals("1")) {
                    while (fork[0]==1 || fork[1]==1) {
                        synchronized (this) {
                            wait();
                        }
                    }
                    fork[0]=1;
                    fork[1]=1;
                    System.out.println("1 eats for 3 seconds");
                    Thread.sleep(3000);
                    fork[0]=0;
                    fork[1]=0;
                    synchronized(this) {
                        notifyAll();
                    }
                    Thread.sleep(100);
                }
                else if (Thread.currentThread().getName().equals("2")) {
                    while (fork[1]==1 || fork[2]==1) {
                        synchronized (this) {
```

```
                        wait();
                    }
                }
                fork[1]=1;
                fork[2]=1;
                System.out.println("2 eats for 3 seconds");
                Thread.sleep(3000);
                fork[1]=0;
                fork[2]=0;
                synchronized(this) {
                    notifyAll();
                }
                Thread.sleep(100);
            }
            else if (Thread.currentThread().getName().equals("3")) {
                while (fork[2]==1 || fork[3]==1) {
                    synchronized (this) {
                        wait();
                    }
                }
                fork[2]=1;
                fork[3]=1;
                System.out.println("3 eats for 3 seconds");
                Thread.sleep(3000);
                fork[2]=0;
                fork[3]=0;
                synchronized(this) {
                    notifyAll();
                }
                Thread.sleep(100);
            }
            else if (Thread.currentThread().getName().equals("4")) {
                while (fork[3]==1 || fork[4]==1) {
                    synchronized (this) {
                        wait();
                    }
                }
                fork[3]=1;
                fork[4]=1;
                System.out.println("4 eats for 3 seconds");
                Thread.sleep(3000);
                fork[3]=0;
                fork[4]=0;
                synchronized(this) {
                    notifyAll();
                }
                Thread.sleep(100);
            }
            else if (Thread.currentThread().getName().equals("5")) {
                while (fork[0]==1 || fork[4]==1) {
                    synchronized (this) {
                        wait();
                    }
                }
                fork[0]=1;
                fork[4]=1;
                System.out.println("5 eats for 3 seconds");
                Thread.sleep(3000);
                fork[0]=0;
                fork[4]=0;
                synchronized(this) {
                    notifyAll();
                }
                Thread.sleep(100);
            }
        }
    } catch(Exception e) {
        e.printStackTrace();
    }
}

public static void main(String[] args) {
    Philosopher2 phi=new Philosopher2();
    for (int i=0;i<5;i++)
        phi.fork[i]=0;
    phi.thread1.start();
    phi.thread2.start();
    phi.thread3.start();
    phi.thread4.start();
```

```
            phi.thread5.start();
        }
}
```

标签： 哲学家问题， java， 新思想

好文要顶 关注我 收藏该文

congmingyige
关注 - 1
粉丝 - 2

0 0

« 上一篇：把矩阵分成n*m个块，从任意一个块出发，问是否可以一笔画遍历矩阵中所有的块
» 下一篇：团体程序设计天梯赛 L1-006. 连续因子

posted @ 2017-12-31 21:02 congmingyige 阅读(17) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

发表评论

昵称： congmingyige

评论内容：

提交评论 退出

[Ctrl+Enter快捷键提交]

最新IT新闻:
· HoloLens 2再传2019年Q1上市，更轻更便宜
· 微软的AI芯片之路，其实已经走了七八年
· 中兴市值一日蒸发超160亿 中金维持推荐评级
· 发财机会：破译这份密码，获取6千万美元

· ofo：B2B业务营收突破1亿 全国百城盈利
» 更多新闻...

最新知识库文章:
· 如何提升你的能力？给年轻程序员的几条建议
· 程序员的那些反模式
· 程序员的宇宙时间线
· 突破程序员思维
· 云、雾和霭计算如何一起工作
» 更多知识库文章...