18342138

Write a program that could cause a deadlock.

Men1 2 3 4 5 分别代表那五个人，一个人的 gotoeat 过程分三部分：try，check，success，分别含义是：尝试去吃，检查隔壁两个人是不是没在吃（死锁），如果没在吃那么就成功吃

```
1.  package Deadlock;
2.
3.  public class Deadlock {
4.
5.      static class MEN {
6.          private final String name;
7.
8.          public MEN(String name) {
9.              this.name = name;
10.         }
11.
12.         public String getName() {
13.             return this.name;
14.         }
15.
16.         public synchronized void go_eat(MEN a, MEN b) {
17.             System.out.println(this.getName() + " try to eat. ");
18.
19.             try {
20.                 Thread.sleep(50);
21.             } catch (InterruptedException e) {
22.                 e.printStackTrace();
23.             }
24.
25.             a.not_eat();
26.             b.not_eat();
27.             System.out.println(this.getName() + " succeed to eat. ");
28.         }
29.
30.         public synchronized void not_eat() {
31.             System.out.println(this.getName() + " is not eating. ");
32.         }
33.     }
34.
```

```
35.     public static void main(String[] args) {
36.         final MEN men1 = new MEN("men1");
37.         final MEN men2 = new MEN("men2");
38.         final MEN men3 = new MEN("men3");
39.         final MEN men4 = new MEN("men4");
40.         final MEN men5 = new MEN("men5");
41.         //( 1 2 3 4 5 - 1 )
42.         new Thread(new Runnable() {
43.             public void run() {
44.                 men1.go_eat(men2, men5);
45.             }
46.         }).start();
47.         new Thread(new Runnable() {
48.             public void run() {
49.                 men2.go_eat(men1, men3);
50.             }
51.         }).start();
52.         new Thread(new Runnable() {
53.             public void run() {
54.                 men3.go_eat(men2, men4);
55.             }
56.         }).start();
57.         new Thread(new Runnable() {
58.             public void run() {
59.                 men4.go_eat(men3, men5);
60.             }
61.         }).start();
62.         new Thread(new Runnable() {
63.             public void run() {
64.                 men5.go_eat(men4, men1);
65.             }
66.         }).start();
67.     }
68. }
```

```
men1 try to eat.
men2 try to eat.
men3 try to eat.
men4 try to eat.
men5 try to eat.
```

Write another program that can avoid the deadlock so that all the philosophers can eat alternately.

每个哲学家必须确认了自己左右手都能拿到叉之后才能吃东西，吃完后同时放下叉

拿不到叉子的时候（判断条件），用 wait 释放锁，某人吃完放下叉子之后，notifyAll 唤醒全部

```
1.  package Deadlock;
2.
3.  class MEN extends Thread{
4.      private String name;
5.      private Fork cur;
6.      public MEN(String name,Fork cur){
7.          super(name);
8.          this.name=name;
9.          this.cur=cur;
10.     }
11.
12.     public void run(){
13.         while(true){
14.             thinking();
15.             cur.tryTodo();
16.             eating();
17.             cur.OverDo();
18.         }
19.     }
20.
21.     public void eating(){
```

```
22.         System.out.println(name + " is eating.");
23.         try {
24.             sleep(1000);
25.         } catch (InterruptedException e) {
26.             e.printStackTrace();
27.         }
28.     }
29.
30.
31.     public void thinking(){
32.         System.out.println(name + " is thinking.");
33.         try {
34.             sleep(1000);
35.         } catch (InterruptedException e) {
36.             e.printStackTrace();
37.         }
38.     }
39. }
40.
41. class Fork{
42.     private boolean[] flag = {false,false,false,false,false,false};
43.
44.     public synchronized void tryTodo(){
45.         int i = Thread.currentThread().getName().charAt(3)-'0';
46.         int a = i, b = i+1;
47.         if(b == 6) b = 1;
48.         while(flag[a]||flag[b]){
49.             try {
50.                 wait();
51.             } catch (InterruptedException e) {
52.                 e.printStackTrace();
53.             }
54.         }
55.         flag[a]=true;  flag[b]=true;
56.     }
57.
58.     public synchronized void OverDo(){
59.         int i = Thread.currentThread().getName().charAt(3)-'0';
60.         int a = i, b = i+1;
61.         if(b == 6) b = 1;
62.         flag[a]=false;  flag[b]=false;
63.         System.out.println(Thread.currentThread().getName() + " is over eati
    ng.");
64.         notifyAll();
```

```
65.     }
66. }
67. 
68. 
69. public class Deadlock {
70.     public static void main(String []args){
71.         Fork cur = new Fork();
72.         // 1 2 3 4 5 -> 1
73.         for(int i=1; i<=5; i++)
74.             new MEN("men" + i, cur).start();
75.     }
76. }
```

Console Deadlock.java

<terminated> Deadlock [Java Application] D:\Java\jdk-12.0.1\bin\javaw.exe (2019年11月21日 下午11:46:41)

```
men5 is thinking.
men4 is eating.
men2 is over eating.
men2 is thinking.
men1 is eating.
men4 is over eating.
men4 is thinking.
men3 is eating.
men1 is over eating.
men1 is thinking.
men5 is eating.
men3 is over eating.
men3 is thinking.
men2 is eating.
men5 is over eating.
men5 is thinking.
men4 is eating.
men2 is over eating.
men2 is thinking.
men1 is eating.
men4 is over eating.
men4 is thinking.
men3 is eating.
men1 is over eating.
men1 is thinking.
men5 is eating.
men3 is over eating.
men3 is thinking.
men2 is eating.
men5 is over eating.
men5 is thinking.
men4 is eating.
men2 is over eating.
men2 is thinking.
men1 is eating.
```