

```

1 package com.example;
2
3
4 public class DeadLock {
5
6     private static final Object LOCK_A = new Object();
7     private static final Object LOCK_B = new Object();
8
9     public static void main(String[] args) {
10
11         Thread thread1 =
12             new Thread(
13                 () -> {
14                     synchronized (LOCK_A) {
15                         System.out.println("Thread 1: получил LOCK_A");
16
17                         ThreadUtils.sleep(100);
18
19                         System.out.println("Thread 1: пытается получить LOCK_B");
20
21                         synchronized (LOCK_B) {
22                             System.out.println("Thread 1: получил LOCK_B");
23                         }
24                     }
25                 });
26
27         Thread thread2 =
28             new Thread(
29                 () -> {
30                     synchronized (LOCK_B) {
31                         System.out.println("Thread 2: получил LOCK_B");
32
33                         ThreadUtils.sleep(100);
34
35                         System.out.println("Thread 2: пытается получить LOCK_A");
36
37                         synchronized (LOCK_A) {

```

```

37     public static void main(String[] args) {
38
39         Thread thread1 =
40             new Thread(
41                 () -> {
42                     synchronized (LOCK_A) {
43                         System.out.println("Thread 1: получил LOCK_A");
44
45                         ThreadUtils.sleep(100);
46
47                         System.out.println("Thread 1: пытается получить LOCK_B");
48
49                         synchronized (LOCK_B) {
50                             System.out.println("Thread 1: получил LOCK_B");
51                         }
52                     }
53                 });
54
55         Thread thread2 =
56             new Thread(
57                 () -> {
58                     synchronized (LOCK_B) {
59                         System.out.println("Thread 2: получил LOCK_B");
60
61                         ThreadUtils.sleep(100);
62
63                         System.out.println("Thread 2: пытается получить LOCK_A");
64
65                         synchronized (LOCK_A) {
66                             System.out.println("Thread 2: получил LOCK_A");
67                         }
68                     }
69                 });
70
71         thread1.start();
72         thread2.start();
73     }
74 }

```

```

1 package com.example;
2
3 public class TwoThreads {
4
5     private static final Object lock = new Object();
6     private static boolean turnOne = true;
7
8     public static void main(String[] args) {
9
10         Thread thread1 =
11             new Thread(
12                 () -> {
13                     while (true) {
14                         synchronized (lock) {
15                             while (!turnOne) {
16                                 try {
17                                     lock.wait();
18                                 } catch (InterruptedException e) {
19                                     Thread.currentThread().interrupt();
20                                     return;
21                                 }
22                             }
23
24                             System.out.print("1");
25                             turnOne = false;
26                             lock.notifyAll();
27                         }
28                     }
29                 });
30
31         Thread thread2 =
32             new Thread(
33                 () -> {
34                     while (true) {
35                         synchronized (lock) {
36                             while (turnOne) {
37

```

```

37         } catch (InterruptedException e) {
38             Thread.currentThread().interrupt();
39             return;
40         }
41     }
42
43     System.out.print("1");
44     turnOne = false;
45     lock.notifyAll();
46 }
47
48 });
49
50 Thread thread2 =
51     new Thread(
52         () -> {
53             while (true) {
54                 synchronized (lock) {
55                     while (turnOne) {
56                         try {
57                             lock.wait();
58                         } catch (InterruptedException e) {
59                             Thread.currentThread().interrupt();
60                             return;
61                         }
62                     }
63
64                     System.out.print("2");
65                     turnOne = true;
66                     lock.notifyAll();
67                 }
68             }
69
70             thread1.start();
71             thread2.start();
72         }
73     });
74
75 thread1.start();
76 thread2.start();
77 }

```

```

@@ -13,14 +13,18 @@ public class LiveLock {
    currentPerson = person;
}

+ public synchronized Person getPerson() {
+     return currentPerson;
+ }

public synchronized void pass() {
    System.out.printf("%s прошёл по коридору!\n", currentPerson.name);
}

static class Person {
-     private String name;
-     private boolean wantsToPass;
+     private final String name;
+     private volatile boolean wantsToPass;

    public Person(String personName) {
        name = personName;
@@ -40,11 +44,12 @@ public class LiveLock {

    // Если сейчас право пройти принадлежит другому человеку,
    // немного ждём и снова проверяем ситуацию.
-     if (corridor.currentPerson != this) {
+     if (corridor.getPerson() != this) {
        try {
            Thread.sleep(1);
        } catch (InterruptedException e) {
-             continue;
+             Thread.currentThread().interrupt();
+             return;
        }
        continue;
    }

@@ -81,6 +86,7 @@ public class LiveLock {
    // Первый поток – Иван.

```

```

-     private String name;
-     private boolean wantsToPass;
+     private final String name;
+     private volatile boolean wantsToPass;

    public Person(String personName) {
        name = personName;
@@ -40,11 +44,12 @@ public class LiveLock {

    // Если сейчас право пройти принадлежит другому человеку,
    // немного ждём и снова проверяем ситуацию.
-     if (corridor.currentPerson != this) {
+     if (corridor.getPerson() != this) {
        try {
            Thread.sleep(1);
        } catch (InterruptedException e) {
-             continue;
+             Thread.currentThread().interrupt();
+             return;
        }
        continue;
    }

@@ -81,6 +86,7 @@ public class LiveLock {
    // Первый поток – Иван.
    new Thread(
        new Runnable() {
+             @Override
            public void run() {
                ivan.walkThrough(corridor, petr);
            }
@@ -90,6 +96,7 @@ public class LiveLock {
    // Второй поток – Пётр.
    new Thread(
        new Runnable() {
+             @Override
            public void run() {
                petr.walkThrough(corridor, ivan);
            }

```

(END)

```
package com.example;
```

```
public class LiveLock {  
  
    static class Corridor {  
        private Person currentPerson;  
  
        public Corridor(Person person) {  
            currentPerson = person;  
        }  
  
        public synchronized void setPerson(Person  
person) {  
            currentPerson = person;  
        }  
  
        public synchronized Person getPerson() {  
            return currentPerson;  
        }  
  
        public synchronized void pass() {  
            System.out.printf("%s прошёл по коридору!%n",  
currentPerson.name);  
        }  
    }  
  
    static class Person {  
        private final String name;  
        private volatile boolean wantsToPass;  
  
        public Person(String personName) {  
            name = personName;  
            wantsToPass = true;  
        }  
  
        public String getName() {  
            return name;  
        }  
    }  
}
```

```
        public boolean wantsToPass() {  
            return wantsToPass;  
        }  
  
        public void walkThrough(Corridor corridor,  
Person otherPerson) {  
            while (wantsToPass) {  
                // Если сейчас право пройти принадлежит  
другому человеку,  
                // немного ждём и снова проверяем ситуацию.  
                if (corridor.getPerson() != this) {  
                    try {  
                        Thread.sleep(1);  
                    } catch (InterruptedException e) {  
                        Thread.currentThread().interrupt();  
                        return;  
                    }  
                    continue;  
                }  
  
                // Если второй человек тоже хочет пройти,  
                // вежливо уступаем ему дорогу.  
                if (otherPerson.wantsToPass()) {  
                    System.out.printf("%s: Проходи первым, %s!  
%n", name, otherPerson.getName());  
  
                    corridor.setPerson(otherPerson);  
                    continue;  
                }  
  
                // Если другой человек уже прошёл,  
                // наконец сами идём по коридору.  
                corridor.pass();  
                wantsToPass = false;  
  
                System.out.printf("%s: Спасибо, %s!%n", @@@
```

```
        public static void main(String[] args) {  
  
            final Person ivan = new Person("Иван");  
            final Person petr = new Person("Пётр");  
  
            // Изначально право пройти по коридору  
принадлежит Ивану.  
            final Corridor corridor = new Corridor(ivan);  
  
            // Первый поток — Иван.  
            new Thread(  
                new Runnable() {  
                    @Override  
                    public void run() {  
                        ivan.walkThrough(corridor, petr);  
                    }  
                })  
                .start();  
  
            // Второй поток — Пётр.  
            new Thread(  
                new Runnable() {  
                    @Override  
                    public void run() {  
                        petr.walkThrough(corridor, ivan);  
                    }  
                })  
                .start();  
        }  
}
```