

```

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

```

```

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 }

```

```

1 package com.example;
2
3 public class LiveLock {
4     static class Corridor {
5         private Person currentPerson;
6
7         public Corridor(Person person) {
8             currentPerson = person;
9         }
10
11         public synchronized void setPerson(Person person) {
12             currentPerson = person;
13         }
14
15         public synchronized void pass() {
16             System.out.printf("%s прошёл по коридору!\n", currentPerson.name);
17         }
18     }
19
20     static class Person {
21         private String name;
22         private boolean wantsToPass;
23
24         public Person(String personName) {
25             name = personName;
26             wantsToPass = true;
27         }
28
29         public String getName() {
30             return name;
31         }
32
33         public boolean wantsToPass() {
34             return wantsToPass;
35         }

```

```

4     public void walkThrough(Corridor corridor, Person otherPerson) {
5         while (wantsToPass) {
6             // Если сейчас право пройти принадлежит другому человеку,
7             // немного ждём и снова проверяем ситуацию.
8             if (corridor.currentPerson != this) {
9                 try {
10                     Thread.sleep(1);
11                 } catch (InterruptedException e) {
12                     continue;
13                 }
14                 continue;
15             }
16
17             // Если второй человек тоже хочет пройти,
18             // вежливо уступаем ему дорогу.
19             if (otherPerson.wantsToPass()) {
20                 System.out.printf("%s: Проходи первым, %s!\n", name, otherPerson.getName());
21
22                 corridor.setPerson(otherPerson);
23                 continue;
24             }
25
26             // Если другой человек уже прошёл,
27             // наконец сами идём по коридору.
28             corridor.pass();
29             wantsToPass = false;
30
31             System.out.printf("%s: Спасибо, %s!\n", name, otherPerson.getName());
32
33             corridor.setPerson(otherPerson);
34         }
35     }
36
37     public static void main(String[] args) {

```

```

26 public static void main(String[] args) {
25
24     final Person ivan = new Person("Иван");
23     final Person petr = new Person("Пётр");
22
21     // Изначально право пройти по коридору принадлежит Ивану.
20     final Corridor corridor = new Corridor(ivan);
19
18     // Первый поток — Иван.
17     new Thread(
16         new Runnable() {
15             public void run() {
14                 ivan.walkThrough(corridor, petr);
13             }
12         })
11         .start();
10
9     // Второй поток — Пётр.
8     new Thread(
7         new Runnable() {
6             public void run() {
5                 petr.walkThrough(corridor, ivan);
4             }
3         })
2         .start();
1     }

```