

```
1 package com.example;
2
3
4 public class LiveLock {
5
6     private static boolean thread1WantsToPass = true;
7     private static boolean thread2WantsToPass = true;
8
9     public static void main(String[] args) {
10         Thread thread1 =
11             new Thread(
12                 () -> {
13                     while (thread1WantsToPass) {
14                         System.out.println("Поток 1: Я уступаю дорогу Потoku 2...");
15                         ThreadUtils.sleep(100);
16                     }
17
18                     System.out.println("Поток 1 прошел!");
19                 });
20
21         Thread thread2 =
22             new Thread(
23                 () -> {
24                     while (thread2WantsToPass) {
25                         System.out.println("Поток 2: Я уступаю дорогу Потoku 1...");
26                         ThreadUtils.sleep(100);
27                     }
28
29                     System.out.println("Поток 2 прошел!");
30                 });
31
32         thread1.start();
33         thread2.start();
34     }
35 }
```

```

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
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                                 try {

```

```

37                                     } catch (InterruptedException e) {
38                                         Thread.currentThread().interrupt();
39                                         return;
40                                     }
41                                 }
42
43                                 System.out.print("2");
44                                 turnOne = true;
45                                 lock.notifyAll();
46                             }
47                         }
48                     }
49                 });
50
51         thread1.start();
52         thread2.start();
53     }
54 }
55

```