File - C:\Users\pugalol\Desktop\programming\lab6-25.03.19\HadesMod\src\Spammer.java

```
1  import java.io.IOException;
2  import java.io.ObjectInputStream;
3  import java.io.ObjectOutputStream;
4  import java.net.InetAddress;
5  import java.net.Socket;
6
7  /**
8   * Класс {@code Spammer} осуществляет замер времени для многопопточного и последовательного обращений к серверу.
9   * РАБОТАЕТ НЕ ПРАВИЛЬНО!
10  */
11 public class Spammer {
12
13     private static int counter = 10;
14     static long start_time;
15
16     public static void main(String[] args) {
17         System.out.println("Запускаю злюкера.");
18         parallel();
19         nonParallel();
20     }
21
22     /**
23      * Параллельно обращение.
24      * 1) Создаём отпечаток времени старта.
25      * 2) Запускаем клиентов в потоках.
26      * 3) После работы последнего замеряем время.
27      */
28     public static void parallel() {
29         start_time = System.currentTimeMillis();
30         Thread[] zombies = new Thread[counter];
31         for (int i = 0; i < counter; i++) {
32             zombies[i] = new Thread(() -> {
33                 try (Socket target = new Socket(InetAddress.getLocalHost(), 8800)) {
34                     try (ObjectOutputStream outputStream = new ObjectOutputStream(target.getOutputStream());
35                          ObjectInputStream inputStream = new ObjectInputStream(target.getInputStream())) {
36                         inputStream.readObject();
37                 //       while (true) { если убрать получится дудос
38                           outputStream.writeObject("info");
39                           inputStream.readObject();
40                     //    }
41                         counter--;
42                         if (counter == 0) System.out.println("Параллельно: " + (System.currentTimeMillis() - start_time) + " millis");
43                     } catch (ClassNotFoundException e) {
44                         e.printStackTrace();
45                 }
46             } catch (IOException e) {
47                 e.printStackTrace();
48             }
49         });
50         }
51         for (Thread t: zombies) t.start();
52     }
53
54     /**
55      * Последовательное обращение.
56      * 1) Создаём отпечаток времени старта.
57      * 2) Запускаем каждого клиента, только после работы предыдущего.
58      * 3) После работы последнего замеряем время.
59      */
60     public static void nonParallel() {
61         start_time = System.currentTimeMillis();
62         for (int i = 0; i < counter; i++) {
63             try (Socket target = new Socket(InetAddress.getLocalHost(), 8800)) {
64                 try (ObjectOutputStream outputStream = new ObjectOutputStream(target.getOutputStream());
65                      ObjectInputStream inputStream = new ObjectInputStream(target.getInputStream())) {
66                     inputStream.readObject();
67                     outputStream.writeObject("info");
```

```
68                      inputStream.readObject();
69                  } catch (ClassNotFoundException e) {
70                      e.printStackTrace();
71                  }
72              } catch (IOException e) {
73                  e.printStackTrace();
74              }
75          }
76          System.out.println("Последовательно: " + (System.currentTimeMillis() -
    start_time) + " millis");
77      }
78  }
```