*Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана  (национальный исследовательский университет)»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Компьютерные системы и сети

ДИСЦИПЛИНА Языки программирования для работы с большими данными

**Отчет**

**по лабораторной работе № 8**

**Вариант 2**

Студент гр. ИУ6-21М **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_**Дейкин И. И.\_\_\_

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

**ЗАДАНИЕ 1**

*Условие*

Вариант 2: Реализовать многопоточное приложение “Робот”. Надо написать робота, который умеет ходить. За движение каждой его ноги отвечает отдельный поток. Шаг выражается в выводе в консоль LEFT или RIGHT.

*Решение*

Класс Календарь с внутренним классом приведен ниже:

|  |
| --- |
| package com.company;  import java.util.Scanner;  import java.util.concurrent.Semaphore;  public class Main {  static Object lock = new Object();  public static void main(String[] args){  Scanner scan = new Scanner(System.in);  System.out.print("How many steps should we make? ");  int steps = scan.nextInt();  Thread left = new Thread(new Runnable() {  public void run() {  int k = steps % 2 == 0 ? steps / 2 : steps / 2 + 1; // Начинаем здесь с левой  for (int i = 0; i < k; i++) { // поэтому при нечетном числе  synchronized (lock) { // шагов левая сделает лишний шаг  if (i == 0) System.out.print("Left");  else System.out.print(" Left");  try {  lock.notify();  if(i == k - 1) break;  lock.wait();  } catch (InterruptedException e) {  e.printStackTrace();  }  }  }  }  });  Thread right = new Thread(new Runnable() {  public void run() {  for (int i = 0; i < steps/2; i++) {  synchronized (lock) {  System.out.print(" Right");  try {  lock.notify();  if(i == steps/2 - 1) break;  lock.wait();  } catch (InterruptedException e) {  e.printStackTrace();  }  }  }  }  });  try {  left.start();  right.start();  left.join();  right.join();  return;  } catch (Exception e) {  }  }  } |

**ВЫВОД**

В ходе выполнения данной работы было реализовано многопоточное приложение, были использованы операции синхронизации двух потоков. Один поток отвечал за левую ногу робота, второй – за правую, после выполнения шага потоки ожидали друг друга.