|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«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 (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 8 |

**Название:** Потоки Threads

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-21М |  |  | Ю.А. Вишневская |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П. В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Вариант 1**

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.

RightFoot

|  |
| --- |
| public class RightFoot extends Thread{  Object CommunicationBetweenFoots = new Object();  public RightFoot(Object communicationBetweenFoots) {  this.CommunicationBetweenFoots = communicationBetweenFoots;  }  public RightFoot(Runnable target, Object communicationBetweenFoots) {  super(target);  CommunicationBetweenFoots = communicationBetweenFoots;  }  public RightFoot(ThreadGroup group, Runnable target, Object communicationBetweenFoots) {  super(group, target);  CommunicationBetweenFoots = communicationBetweenFoots;  }  public RightFoot(String name, Object communicationBetweenFoots) {  super(name);  CommunicationBetweenFoots = communicationBetweenFoots;  }  public Object getCommunicationBetweenFoots() {  return CommunicationBetweenFoots;  }  public void setCommunicationBetweenFoots(Object communicationBetweenFoots) {  CommunicationBetweenFoots = communicationBetweenFoots;  }  @Override  public void run() {  while(true) {  try {  synchronized(CommunicationBetweenFoots) {  CommunicationBetweenFoots.wait();  }  } catch (InterruptedException e) {}  System.out.println("RIGHT");  try {  sleep(1000);  } catch (InterruptedException e1) {}  synchronized(CommunicationBetweenFoots) {  CommunicationBetweenFoots.notify();  }  }  }  } |

LeftFoot

|  |
| --- |
| public class LeftFoot extends Thread {  Object CommunicationBetweenFoots = new Object();  public LeftFoot(Object communicationBetweenFoots) {  this.CommunicationBetweenFoots = communicationBetweenFoots;  }  public LeftFoot(Runnable target, Object communicationBetweenFoots) {  super(target);  CommunicationBetweenFoots = communicationBetweenFoots;  }  public LeftFoot(ThreadGroup group, Runnable target, Object communicationBetweenFoots) {  super(group, target);  CommunicationBetweenFoots = communicationBetweenFoots;  }  public Object getCommunicationBetweenFoots() {  return CommunicationBetweenFoots;  }  public void setCommunicationBetweenFoots(Object communicationBetweenFoots) {  CommunicationBetweenFoots = communicationBetweenFoots;  }  @Override  public void run() {  while (true) {  System.out.println("LEFT");  try {  sleep(2000);  } catch (InterruptedException e1) {  }  synchronized (CommunicationBetweenFoots) {  CommunicationBetweenFoots.notify();  }  try {  synchronized (CommunicationBetweenFoots) {  CommunicationBetweenFoots.wait();  }  } catch (InterruptedException e) {  e.printStackTrace();  }  }  }  } |

LR8\_V1

|  |
| --- |
| public class LR8\_V1 {  public static void main(String[] args) {  Object CommunicationBetweenFoots = new Object();  RightFoot RightFoot = new RightFoot(CommunicationBetweenFoots);  LeftFoot LeftFoot = new LeftFoot(CommunicationBetweenFoots);  RightFoot.start();  LeftFoot.start();  }  } |