|  |  |
| --- | --- |
| **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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных.**

**Отчет**

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

**Название:**

Потоки

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | Д.Ю.Хотин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В.Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Вариант 1, задание 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.

Код

|  |
| --- |
| import java.lang.Runnable;  import java.lang.Thread;  public class Main {  public static void main(String[] args) {  LegMovement leftLeg = new LegMovement("LEFT");  LegMovement rightLeg = new LegMovement("RIGHT");  Thread leftLegThread = new Thread(leftLeg);  Thread rightLegThread = new Thread(rightLeg);  leftLegThread.start();  rightLegThread.start();  }  }  class LegMovement implements Runnable {  private final String legName;  public LegMovement(String legName) {  this.legName = legName;  }  @Override  public void run() {  while (true) {  try {  Thread.sleep(1000); // ждем одну секунду между шагами  } catch (InterruptedException e) {  e.printStackTrace();  }  System.out.println(legName);  }  }  } |

**Вариант 1, задание 3:** Реализовать многопоточное приложение “Магазин”. Вся цепочка: производитель-магазин-покупатель. Пока производитель не поставит на склад продукт, покупатель не может его забрать. Реализовать приход товара от производителя в магазин случайным числом. В том случае, если товара в магазине не хватает– вывести сообщение.

Код

|  |
| --- |
| import java.util.concurrent.\*;  public class Shop {  private int stock = 0;  private final int maxStock = 10;  public synchronized void produce() throws InterruptedException {  while (stock == maxStock) {  wait();  }  int randomAmount = (int) (Math.random() \* 10) + 1;  stock += randomAmount;  System.out.println("The producer has stocked " + randomAmount + " products. Amount: " + stock);  notifyAll();  }  public synchronized void consume() throws InterruptedException {  while (stock == 0) {  wait();  }  int randomAmount = (int) (Math.random() \* 10) + 1;  if (stock >= randomAmount) {  stock -= randomAmount;  System.out.println("Consumer buy " + randomAmount + " products. Stock: " + stock);  } else {  System.out.println("Not enough products in stock!");  }  notifyAll();  }  public static void main(String[] args) {  Shop shop = new Shop();  ExecutorService executor = Executors.newFixedThreadPool(3);  executor.submit(new Producer(shop));  executor.submit(new Consumer(shop));  executor.submit(new Consumer(shop));  executor.shutdown();  }  }  class Producer implements Runnable {  private final Shop shop;  public Producer(Shop shop) {  this.shop = shop;  }  @Override  public void run() {  try {  while (true) {  shop.produce();  Thread.sleep(1000); // ждем одну секунду перед следующей поставкой  }  } catch (InterruptedException e) {  e.printStackTrace();  }  }  }  class Consumer implements Runnable {  private final Shop shop;  public Consumer(Shop shop) {  this.shop = shop;  }  @Override  public void run() {  try {  while (true) {  shop.consume();  Thread.sleep(2000); // ждем две секунды перед следующей покупкой  }  } catch (InterruptedException e) {  e.printStackTrace();  }  }  } |