**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 **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

О Т Ч Е Т

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

**Дисциплина:** Потоки

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

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

Москва, 2023

**Цель:** ознакомиться с базовыми принципами языка Java для работы с большими данными

**Вариант 1:**

1. 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.
2. 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.

**Решение:**

package lr8\_v1\_task2;  
  
public class Main {  
 public static void main(String[] args) {  
 Robot robot = new Robot();  
 robot.leftMove();  
 robot.rightMove();  
 }  
}

package lr8\_v1\_task2;  
  
public class Robot {  
 private Thread leftLegMoving;  
 private Thread rightLegMoving;  
 public void leftMove(){  
 leftLegMoving = new Thread(() -> {  
 while (true){  
 System.*out*.println("LEFT");  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 }  
 });  
 leftLegMoving.start();  
 }  
  
 public void rightMove(){  
 rightLegMoving = new Thread(() -> {  
 while (true){  
 System.*out*.println("RIGHT");  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 }  
 });  
 rightLegMoving.start();  
 }  
}

На рисунке 1 представлен результат выполнения.

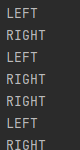


Рисунок 1 – Результат выполнения

package lr8\_v1\_task3;  
  
public class Customer implements Runnable {  
 private Store store;  
 public Customer(Store store){  
 this.store = store;  
 }  
  
 @Override  
 public void run() {  
 while (true) {  
 int amount = (int) (Math.*random*() \* 5) + 1;  
 try {  
 store.consume(amount);  
 Thread.*sleep*((int) (Math.*random*() \* 5000));  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 }  
 }  
}

package lr8\_v1\_task3;  
  
public class Main {  
 public static void main(String[] args) {  
 Store store = new Store();  
 new Thread(new Producer(store)).start();  
 new Thread(new Customer(store)).start();  
 }  
}

package lr8\_v1\_task3;  
  
public class Producer implements Runnable {  
 private Store store;  
 public Praoducer(Store store){  
 this.store = store;  
 }  
  
 @Override  
 public void run() {  
 while (true) {  
 int amount = (int) (Math.*random*() \* 5) + 1;  
 try {  
 store.produce(amount);  
 Thread.*sleep*((int) (Math.*random*() \* 5000));  
 } catch (InterruptedException e) {  
 throw new RuntimeException(e);  
 }  
 }  
 }  
}

package lr8\_v1\_task3;  
  
import java.util.concurrent.locks.Condition;  
import java.util.concurrent.locks.Lock;  
import java.util.concurrent.locks.ReentrantLock;  
  
public class Store {  
 private int stock = 0;  
 private final int MAX\_STOCK = 10;  
 private final Lock lock = new ReentrantLock();  
 private final Condition conditionCustomer = lock.newCondition();  
 private final Condition conditionProducer = lock.newCondition();  
  
 public void produce(int quantity) throws InterruptedException {  
 lock.lock();  
 try {  
 while (stock + quantity >= MAX\_STOCK){  
 System.out.println("Не хватает места на складе для " + quantity + " товаров.");  
 conditionProducer.await();  
 }  
 stock += quantity;  
 System.*out*.println("Производитель поставил на склад " + quantity + " товаров. Всего на складе: " + stock);  
 conditionCustomer.signalAll();  
 }  
 finally {  
 lock.unlock();  
 }  
 }  
  
 public void consume(int quantity) throws InterruptedException {  
 lock.lock();  
 try {  
 while (stock - quantity < 0){  
 System.*out*.println("Товара в магазине недостаточно. Ожидаем...");  
 conditionCustomer.await();  
 }  
 stock -= quantity;  
 System.*out*.println("Покупатель купил " + quantity + " товаров. Осталось на складе: " + stock);  
 conditionProducer.signalAll();  
 }  
 finally {  
 lock.unlock();  
 }  
  
 }  
}

На рисунке 2 представлен результат выполнения.

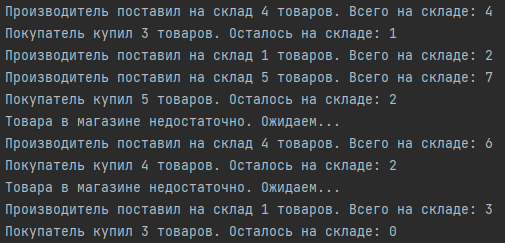


Рисунок 2 – Результат выполнения

**Вывод:** были разработаны классы и методы согласно варианту