|  |  |
| --- | --- |
| 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**

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

**обработки и интерпретации больших данных.**

**Отчет**

**по лабораторной работе № 8**

**Вариант № 12**

**Название:** потоки

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

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

Москва, 2024

**Цель:** изучить потоки в java

**Задание 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.

Код класса Robot:

public class Robot {

private final Object lock = new Object();

private boolean isLeftTurn = true; // true если очередь левой ноги, иначе очередь правой ноги

public static void main(String[] args) {

Robot robot = new Robot();

Thread leftLeg = new Thread(new Leg(robot, true));

Thread rightLeg = new Thread(new Leg(robot, false));

leftLeg.start();

rightLeg.start();

}

public void step(boolean isLeft) {

synchronized (lock) {

while (isLeft != isLeftTurn) {

try {

lock.wait();

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

System.out.println(isLeft ? "LEFT" : "RIGHT");

isLeftTurn = !isLeftTurn;

lock.notifyAll();

}

}

static class Leg implements Runnable {

private final Robot robot;

private final boolean isLeft;

Leg(Robot robot, boolean isLeft) {

this.robot = robot;

this.isLeft = isLeft;

}

@Override

public void run() {

while (true) {

robot.step(isLeft);

try {

Thread.sleep(500); // Задержка для имитации шага

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

}

}

}Работа программы показана на рисунке 1.

Изображение выглядит как снимок экрана, Мультимедийное программное обеспечение, черный, Графическое программное обеспечение

Автоматически созданное описание

Рисунок 1 – Работа программы

**Задание 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.

Код класса Shop:

//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.

import java.util.Random;

import java.util.concurrent.BlockingQueue;

import java.util.concurrent.LinkedBlockingQueue;

import java.util.concurrent.TimeUnit;

public class Shop {

private static final int MAX\_CAPACITY = 10;

private static final int PRODUCE\_TIME = 1000;

private static final int CONSUME\_TIME = 1500;

private static final int MAX\_PRODUCT\_AMOUNT = 5;

public static void main(String[] args) {

BlockingQueue<Integer> warehouse = new LinkedBlockingQueue<>(MAX\_CAPACITY);

Thread producer = new Thread(new Producer(warehouse));

Thread consumer = new Thread(new Consumer(warehouse));

producer.start();

consumer.start();

}

static class Producer implements Runnable {

private final BlockingQueue<Integer> warehouse;

private final Random random = new Random();

Producer(BlockingQueue<Integer> warehouse) {

this.warehouse = warehouse;

}

@Override

public void run() {

try {

while (true) {

int product = random.nextInt(MAX\_PRODUCT\_AMOUNT) + 1;

System.out.println("Производитель пытается поставить " + product + " товар(ов).");

for (int i = 0; i < product; i++) {

if (warehouse.offer(1, PRODUCE\_TIME, TimeUnit.MILLISECONDS)) {

System.out.println("Производитель поставил товар на склад. Текущий запас: " + warehouse.size());

} else {

System.out.println("Склад переполнен. Не удалось поставить товар.");

}

}

Thread.sleep(PRODUCE\_TIME);

}

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

}

static class Consumer implements Runnable {

private final BlockingQueue<Integer> warehouse;

Consumer(BlockingQueue<Integer> warehouse) {

this.warehouse = warehouse;

}

@Override

public void run() {

try {

while (true) {

int productNeeded = 1; // Каждое потребление — один товар

if (warehouse.poll(CONSUME\_TIME, TimeUnit.MILLISECONDS) != null) {

System.out.println("Покупатель забрал товар. Текущий запас: " + warehouse.size());

} else {

System.out.println("Недостаточно товара на складе для покупателя.");

}

Thread.sleep(CONSUME\_TIME);

}

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

}

}

}

Работа программы показана на рисунке 2.

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 2 – Работа программы

**Вывод:** во время выполнения лабораторной работы были изучены потоки в java.