|  |  |
| --- | --- |
| 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 №** 8 |  |

**Название:**

Потоки

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-23М | |  |  | Н.М. Иванюк |
|  | | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  | |  |  |  |  |
| Преподаватель | |  |  |  | П.В. Степанов |
|  | |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2022

## Цель работы

Получение навыков работы с потоками в языке программирования Java.

## Ход работы

*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.

– 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.

Листинг 1 – Код основной программы

import java.util.Random;

import java.util.concurrent.TimeUnit;

public class App {

public static void main(String[] args) throws Exception {

// === ROBOT LEGS (2) ===

Runnable rightStep = () -> {

while (true) {

System.out.println("Make Right Step");

try {

TimeUnit.SECONDS.sleep(10);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

};

Runnable leftStep = () -> {

while (true) {

System.out.println("Make Left Step");

try {

TimeUnit.SECONDS.sleep(10);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

};

Thread rightLeg = new Thread(rightStep);

Thread leftLeg = new Thread(leftStep);

rightLeg.start();

TimeUnit.SECONDS.sleep(5);

leftLeg.start();

// === WAREHOUSE (3) ===

Store storage = new Store();

System.out.println("Initial amount: " + storage.get());

Runnable stock = () -> {

while (true) {

System.out.println("Manufacturer sent items to stock...");

storage.addRandom();

int inStorage = storage.get();

System.out.println("Current amount: " + inStorage);

try {

TimeUnit.SECONDS.sleep(5);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

};

Runnable buy = () -> {

while (true) {

System.out.println("Buyer tries to buy some items...");

storage.retrieveItems();

int inStorage = storage.get();

System.out.println("Current amount: " + inStorage);

try {

TimeUnit.SECONDS.sleep(3);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

};

Thread manufacturerThread = new Thread(stock);

Thread buyerThread = new Thread(buy);

manufacturerThread.start();

TimeUnit.SECONDS.sleep(2);

buyerThread.start();

}

private static class Store {

private int amount = 0;

public synchronized void addRandom() {

Random random = new Random();

int randomInt = random.nextInt(10 - 2) + 2;

amount += randomInt;

}

public synchronized void retrieveItems() {

if (amount >= 5) {

amount -= 5;

} else {

System.out.println("Items can't be retrieved. Not enough amount.");

}

}

public synchronized int get() {

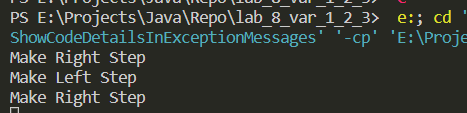
return amount;

}

}

}

Приведем результаты выполнения данного кода.



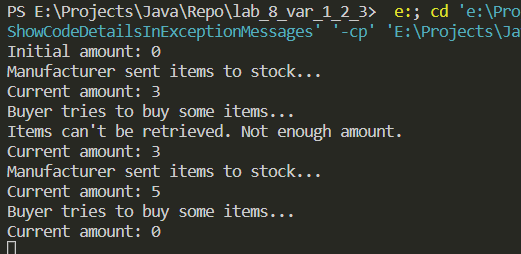


Рисунок 1 – Результат выполнения варианта задания 1

Для обработки доступа потоков к критической секции был использован такой механизм, как synchronized методы. Это ключевое слово, которое позволяет заблокировать доступ к методу или части кода, если его уже использует другой поток.

## Вывод

По итогам выполнения лабораторной работы были получены навыки программирования с использованием потоков на языке Java. Был получен опыт обработки доступа потоков к критическим секциям.