|  |  |
| --- | --- |
| 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 16** |

**Название:**

Потоки

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ6-23М | |  |  | В.И. Тарасов |
|  | | (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 Java.

**Ход работы**

**Задание 1:**

1. Реализовать многопоточное приложение “Банк”. Имеется банковский счет. Сделать синхронным пополнение и снятие денежных средств на счет/со счет случайной суммой. При каждой операции (пополнения или снятие) вывести текущий баланс счета. В том случае, если денежных средств недостаточно – вывести сообщение.
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.

Листинг 1 – Код класса Account

package src.lab8;

public class Account {

public long balance = 0;

public Account(long balance) {

this.balance = balance;

}

public long getBalance() {

return balance;

}

void add(long value) {

System.out.println("Пополнение " + value);

this.balance += value;

System.out.println("Остаток: " + this.balance);

}

void withdraw(long value) {

System.out.println("Списание: " + value);

if (value > this.balance) {

throw new IllegalArgumentException("Недостаточно средств");

}

this.balance -= value;

System.out.println("Остаток: " + this.balance);

}

}

Листинг 2 – Код класса AccountThread

package src.lab8;

import java.util.Objects;

public class AccountThread implements Runnable{

final Account res;

String action;

long amount;

AccountThread(Account res, String action, long amount){

this.res = res;

this.action = action;

this.amount = amount;

}

@Override

public void run() {

synchronized (res) {

System.out.println(Thread.currentThread().getName());

if (Objects.equals(action, "withdraw")) {

try {

res.withdraw(amount);

} catch (Exception e) {

System.out.println(e.getMessage());

}

}

else if (Objects.equals(action, "add")){

res.add(amount);

}

}

}

}

Листинг 3 – Код основной программы подзадачи 1

package src.lab8;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.Random;

public class Program2 {

public static void main(String[] args) {

// 1. Реализовать многопоточное приложение “Банк”. Имеется банковский счет.

// Сделать синхронным пополнение и снятие денежных средств на счет/со счет случайной суммой.

// При каждой операции (пополнения или снятие) вывести текущий баланс счета.

// В том случае, если денежных средств недостаточно – вывести сообщение.

Random rand = new Random();

List<String> actions = Arrays.asList("withdraw", "add");

Account account = new Account(2000);

for (int i = 0; i < 5; i++){

String action = actions.get(rand.nextInt(actions.size()));

long amount = rand.nextInt(3000);

Thread t;

t = new Thread(new AccountThread(account, action, amount));

t.setName("Поток " + i);

t.start();

}

}

}

Результаты выполнения задачи представлены на рисунке 1.

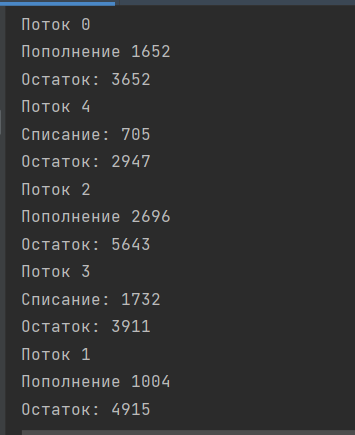


Рисунок 1 – Результат выполнения подзадачи 1

Листинг 4 – Код основной программы подзадания 2

package src.lab8;

public class Program {

volatile boolean currentLeg = true;

class Leg implements Runnable {

public String name;

public boolean leg;

Leg(String name, boolean leg) {

this.name = name;

this.leg = leg;

}

@Override

public void run() {

while(true) {

if (leg == currentLeg) {

step();

currentLeg = !leg;

}

}

}

private void step() {

System.out.println(name);

}

}

Leg left = new Leg("LEFT", false);

Leg right = new Leg("RIGHT", true);

void run() {

new Thread(left).start();

new Thread(right).start();

}

public static void main(String[] args) {

new Program().run();

}

}

Результаты выполнения подзадачи 2 представлены на рисунке 2.

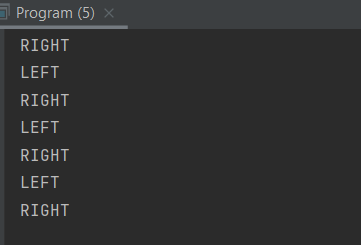


Рисунок 2 – Результаты выполнения подзадачи 2

**Местоположение проектных файлов** –файлы проекта расположены в репозитории GitHub по следующим адресам:

<https://github.com/Fregatty/BigDataLanguages/tree/main/src/lab8>

**Вывод** – в результате выполнения лабораторной работы были получены навыки работы с потоками в языке Java. Также получен опыт обработки доступа к критической секции.