ФЕДЕРАЛЬНОЕ Государственное АВТОНОМНОЕ образовательное УЧРЕЖДЕНИЕ ВЫСШЕГО образования

**БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ**

**ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»**

**(НИУ «БелГУ»)**

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРАИНФОРМАЦИОННЫХ СИСТЕМ

**Отчет**

по дисциплине «Разработка программных приложений на языках высокого уровня»

специальности 09.03.03 «Прикладная информатика»

**Лабораторная работа №5**

студентки очного отделения

3 курса 12002005 группы

Хариной Е. Д.

Проверил:

Лифиренко М.В.

БЕЛГОРОД 2023

Цель работы

Получить практические навыки создания многопоточных приложений.

## Содержание работы:

1 Разработать многопоточную программу для указанного варианта.

2 В приложении должен быть главный поток, порожденные потоки выполняющие

определенный процесс и поток, выводящий на экран результаты.

## Результаты выполнения работы

В директории лежат входные текстовые файлы, проименованные следующим

образом: in\_<N>.dat, где N - натуральное число. Каждый файл состоит из двух строк. В первой строке - число, обозначающее действие, а во второй - числа с плавающей точкой, разделенные пробелом. Действия могут быть следующими: 1 – сложение, 2 – умножение, 3 - сумма квадратов.

Необходимо написать многопоточное приложение, которое выполнит требуемые действия над числами и сумму результатов запишет в файл out.dat. Название рабочей директории передается в виде аргумента рабочей строки. В реализации приветствуется использование полиморфизма и паттернов проектирования.

## Листинг программы:

import java.io.FileNotFoundException;

import java.io.PrintWriter;

import java.io.UnsupportedEncodingException;

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

String pathIn;

ArrayList<Thread> threads = new ArrayList<>();

PrintWriter printWriter;

FileReadInNewThread fileReadInNewThread;

try {

printWriter = new PrintWriter("out.bat", "UTF-8");

} catch (FileNotFoundException e) {

throw new RuntimeException(e);

} catch (UnsupportedEncodingException e) {

throw new RuntimeException(e);

}

for (int i = 0; i < 4; i++) {

pathIn = "in\_%s.bat".formatted(i);

fileReadInNewThread = new FileReadInNewThread(pathIn, printWriter, i);

Thread thread = new Thread(fileReadInNewThread);

thread.start();

System.out.println("Поток" + i + " создался");

threads.add(thread);

}

for (Thread thread : threads) {

try {

thread.join();

} catch (InterruptedException e) {

throw new RuntimeException(e);

}

}

printWriter.close();

System.out.println("Завершение основного потока");

}

}

import java.io.File;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.Scanner;

public class FileReadInNewThread implements Runnable

{

File file;

public int number;

private final PrintWriter printWriter;

ArrayList<Double> values = new ArrayList<>();

Scanner scanner;

/\*\*Конструктор класса FileReadInNewThread инициализирует переменные которые передаются по параметрам

\* @param pathIn - путь к файлу с которого будет считываться информация типа (string)

\* @param printWriter - класс, производящий запись в файл\*/

public FileReadInNewThread(String pathIn, PrintWriter printWriter, int number){

file = new File(pathIn);

this.number = number;

try {

scanner = new Scanner(this.file);

this.printWriter = printWriter;

}

catch (Exception e)

{

throw new RuntimeException();

}

}

@Override

public void run() {

int operation;

try

{

operation = scanner.nextInt();

}

catch (Exception e)

{

throw new RuntimeException();

}

fillArayList(values,scanner);

switch (operation) {

case 1 -> printOnFile(printWriter, sumAll(values));

case 2 -> printOnFile(printWriter, multiplicationAll(values));

case 3 -> printOnFile(printWriter, sumOfSquares(values));

default -> {

}

}

}

/\*\*Функция производит заполнение ранее инициализированного ArrayList с ранее инициализированного Scaner

\* @param values - ArayList типа double, который будет заполняться значениями из файла(должен быть ранее инициализирован)

\* @param scanner - сканер, который хранит ссылку на файл, с которого производится считывание\*/

private void fillArayList(ArrayList<Double> values, Scanner scanner){

while (scanner.hasNextDouble()){

values.add(scanner.nextDouble());

}

}

/\*\*Функция суммирует все элементы коллекции ArrayList типа double

\* @param values - коллекция ArrayList типа double

\* @return возвращает сумму коллекции типа double\*/

private double sumAll(ArrayList <Double> values)

{

double sum = 0;

for (Double value : values) {

sum += value;

}

return sum;

}

/\*\*Функция умножает все элементы коллекции ArrayList типа double

\* @param values - коллекция ArrayList типа double

\* @return возвращает произведение коллекции тип данных double\*/

private double multiplicationAll(ArrayList <Double> values)

{

double multiplication = 1;

for (Double value : values) {

multiplication \*= value;

}

return multiplication;

}

/\*\*Функция суммирует квадраты всех элементов коллекции ArrayList типа double

\* @param values - коллекция ArrayList типа double

\* @return возвращает сумму квадратов коллекции тип данных double\*/

private double sumOfSquares(ArrayList <Double> values){

double sum = 0;

for (Double value : values) {

sum += Math.pow(value, 2);

}

return sum;

}

/\*\*Функция выполняет запись в файл полученных данных

\* @param printWriter - ранее инициализированный объект типа PrintWriter, хронящий ссылку на файл для вывода

\* @param res - результат который необходимо вывести\*/

private void printOnFile(PrintWriter printWriter, double res)

{

printWriter.println("Поток "+ number+" вывел " + res);

}

}

Результат выполнения программы представлен на рисунках 1 и 2.

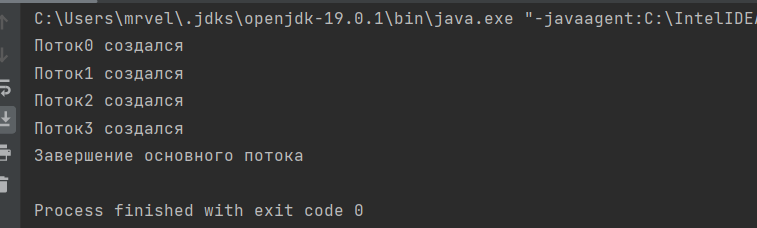


Рисунок 1 – Вывод в консоль

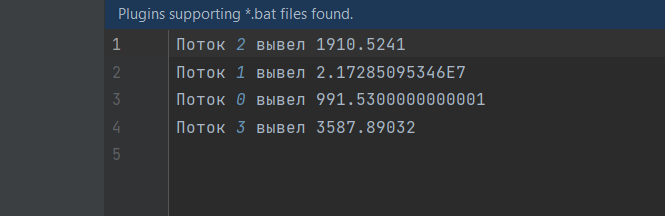


Рисунок 2 - Файл

## Вывод

В ходе выполнения лабораторной работы были получены практические навыки создания многопоточных приложений.