Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение   
высшего образования

**Иркутский национальный исследовательский технический университет**

|  |
| --- |
| Институт информационных технологий и анализа данных |
| наименование института |

|  |
| --- |
| **Отчет** |
| к лабораторной работе №3  по дисциплине «ООП»  **«**Работа с потоками данных и обработка исключений**»** |
| наименование темы  Вариант №1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ЭВМб-19-2 |  |  |  | А. А. Бодров |
|  |  | шифр |  | подпись |  | И. О. Фамилия |
| Проверил |  |  |  |  |  | Т. В. Маланова |
|  |  | должность |  | подпись |  | И. О. Фамилия |
| Работа защищена с оценкой | | | |  | | |

Иркутск 2020 г.

# 1 Постановка задачи

Провести объектную декомпозицию задачи из индивидуального задания и разработать класс, содержащий соответствующие поля для хранения необходимых данных и методы, обеспечивающие достаточную для решения задачи функциональность класса.

Разработать класс, реализующий текстовое меню, позволяющее осуществлять выбор вариантов выполнения индивидуального задания. Создание объекта класса решающего задание и вызов его методов должны осуществляться из класса реализующего меню.

При выполнении задания предусмотреть методы для организации ввода и вывода данных как на консоль, так и в текстовые файлы. Работа с файлами может быть организована методами класса реализующего пользовательское меню, либо в рамках специально разработанного класса.

Для всех проверяемых исключений должны быть реализованы обработчики, в которых выводится сообщение о возникновении исключительной ситуации, и предпринимаются меры для продолжения работы программы в штатном режиме.

*Индивидуальное задание:*

a) Вычислите сумму диагоналей квадратной матрицы.

b) Определите, является ли заданная строка символов числом.

c) Выполнить сложение двух матриц.

# 2 Структура меню

На рис. 1 представлена схема структуры пользовательского меню.

doTwoMatrix

doString

doMatrix

openTwoMatrix

openString

openMatrix

Pick == 4?

Pick == 3?

Pick == 2?

Pick == 1?

Рисунок 1 – Схема структуры пользовательского меню

За запуск пользовательского меню отвечает статический метод **start**, вызываемый из метода **main** (точка входа в программу) класса **Program**.

*Описание методов класса Menu:*

1. Метод **openMatrix –** статический метод, отвечающий за создание объекта класса **Matrix**. Пользователю доступны 2 способа заполнения матрицы: ручной ввод с клавиатуры и чтение матрицы из файла.

2. Метод **openString –** статический метод, отвечающий за создание объекта класса **StringNew**. Пользователю доступны 2 способа заполнения матрицы: ручной ввод с клавиатуры и чтение строки из файла.

3. Метод **doMatrix –** статический метод, отвечающий за работу над объектом класса **Matrix**. Предоставляет пользователю найти сумму главной и второстепенной диагонали.

4. Метод **doString –** статический метод, отвечающий за работу над объектом класса **StringNew**. Предоставляет пользователю проверить строку на число.

5. Метод **openTwoMatrix –** статический метод, отвечающий за создание объекта класса **Addition**. Пользователю доступны 2 способа заполнения матрицы: ручной ввод с клавиатуры и чтение матрицы из файла.

6. Метод **doTwoMatrix –** статический метод, отвечающий за работу над объектом класса **Addition**. Предоставляет пользователю найти сумму двух матриц.

# 3 Проектирование классов

На рис. 2 представлена схема структуры классов.

WriteReadFile

Addition

StringNew

Matrix

Control

Menu

Program\_processing\_java

Рисунок 2 – Схема структуры классов

*Описание классов:*

1. Класс **Program –** главный класс, метод **main** которого используется в качестве точки входа в программу.

2. Класс **Menu –** класс, реализующий текстовое меню, позволяющее осуществлять выбор пользователем вариантов выполнения индивидуального задания.

3. Класс **Matrix –** класс-обертка, предоставляющий дополнительные методы для работы с квадратной матрицей целых чисел и обеспечивающий решение 1-го индивидуального задания.

4. Класс **StringNew–** класс-обертка, предоставляющий дополнительные методы для работы с массивом строк и обеспечивающий решение 2-го индивидуального задания.

5. Класс **Addition –** класс-обертка, предоставляющий дополнительные методы для работы с квадратной матрицей целых чисел и обеспечивающий решение 3-го индивидуального задания.

6. Класс **WriteReadFile –** вспомогательныйкласс, отвечающий за работу с потоками ввода/вывода данных.

7. Класс **Control –** вспомогательный класс, единствееной задачей которого является контроль над вводом пользователем числовых значений с клавиатуры.

# 4 Спецификации

Таблица 1 – Описание полей классов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип данных | | Тип доступа | | ОДЗ |
| Класс Matrix | | | | | | | |
| 1 | matrix | Квадратная матрица | int [][] | | private | | [-231:231-1] |
| Класс StringNew | | | | | | | |
| 2 | S1 | Массив букв | String [] | private | | ‘A’ .. ‘z’,  ‘А’ .. ‘я’,  ‘0’ .. ‘9’, | |
| Класс Addition | | | | | | | |
| 3 | matrixA | Квадратная матрица | int [][] | private | | [-231:231-1] | |
| 4 | size | Размерность матрицы | int | private | | [-231:231-1] | |
| Класс Menu | | | | | | | |
| 5 | scanner | Объект класса Scanner, отвечающий за ввод данных с клавиатуры | Scanner | private | | - | |
| 6 | matrix | Объект класса Matrix | Matrix | private | | - | |
| 7 | matrixA | Объект класса Addition | Matrix | private | | - | |
| 8 | s1 | Объект класса StringNew | StringNew | private | | - | |

Таблица 2 – Описание методов класса Control

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип данных | Тип доступа | Параметры |
| 1 | input | Контроль над вводом числовых значений с клавиатуры (статический метод) | int | public | *int firstBorder*– левый предел значений  *int endBorder* – правый предел значений |

Таблица 3 – Описание методов класса Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип данных | Тип доступа | Параметры |
| 1 | getElement | Получаем элемент матрицы | int[][] | public | int i – *индекс строки* |
| int j – *индекс столбца* |
| 2 | setElement | Устанавливаем значение в элемент | int[][] | public | int i – *индекс строки* |
| int j – *индекс столбца* |
| value – *добавляемый новый элемент в матрицу* |
| 3 | Matrix | Конструктор по умолчанию | - | public | - |
| 4 | Matrix | Конструктор с параметрами | - | public | *int size* – размерность матрицы |
| 5 | Matrix | Конструктор копирования | - | public | *Matrix kopirovan –* объект копирования. |
| 6 | sum1 | Нахождение суммы диагоналей | int | public | *-* |
| 7 | sum2 | Нахождение суммы второй дигонали | int | public | *-* |
| 8 | getMatrix | Перевод из одномерного массива в двумерный | int[] | public | *-* |

Таблица 4 – Описание методов класса StringNew

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип данных | Тип доступа | Параметры |
| 1 | StringNew | Конструктор по умолчанию | - | public | - |
| 2 | StringNew | Конструктор с параметрами | - | public | String s1 – Данная строка |
| 3 | StringNew | Конструктор копирования | - | public | StringNew stringNew2 – объект копирования |
| 4 | isDigitalSrting | Проверка на символы | boolian | public | - |
| 5 | getString | Возврат s1 | - | public | - |

Таблица 5 – Описание методов класса Additition

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип данных | Тип доступа | Параметры |
| 1 | getElement | Получаем элемент матрицы | int[][] | public | int i – *индекс строки* |
| int j – *индекс столбца* |
| 2 | setElement | Устанавливаем значение в элемент | int[][] | public | int i – *индекс строки* |
| int j – *индекс столбца* |
| value – *добавляемый новый элемент в матрицу* |
| 3 | Addition | Конструктор по умолчанию | - | public | - |
| 4 | Addition | Конструктор с параметрами | int[][] | public | *int size* – размерность матрицы |
| 5 | Addition | Конструктор копирования | int[][] | public | *Addition addcopirov – копируемый объект* |
| 6 | sumTwoMatrix | Нахождение суммы матрицы | int[][] | public | *int [][] matrix – прибавляемая матрица* |
| 7 | getSquareMatrix | Возврат квадратной матрицы | int [][] | public | *-* |

Таблица 6 – Описание методов класса SaveLoader

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип данных | Тип доступа | Параметры |
| 1 | saveString | Запись строковых данных в файл | void | public | *String saved* – сохраняемая строка  *String fullname* – название файла |
| 2 | loadString | Чтение строковых данных из файла | String | public | *String fullname* – название файла |
| 3 | saveInt | Запись числовых данных в файл | void | public | *int [] saved* – сохраняемый массив  *String fullname* – название файла |
| 4 | loadInt | Чтение числовых данных из файла | int [] | public | *String fullname* – название файла |
| 5 | hasSaving | Возвращает информация о сохранении файла | boolean | public | *-* |

# 5 Проектирование тестов

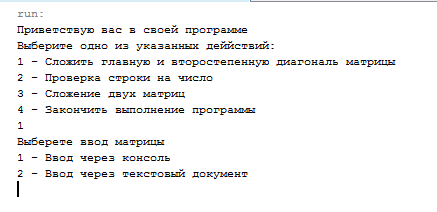
Таблица 6 – Тестирование программы

|  |  |  |
| --- | --- | --- |
| № | Действие | Результат действия |
| 1 | Первоначальный запуск программы | Выводится приветствие, затем запускается текстовое меню с выбором действий (рис. 3). |
| 2 | Выбор действия “Сложить главную и второстепенную диагональ матрицы ” | Вводим “1”, после чего появляется подменю с выбором способа ввода матрицы (рис. 4). |
| 3 | Ручной ввод значений элементов матрицы | Вводим “1”, после чего задаем размер матрицы. Вводим сами значения элементов. В случае неправильного ввода, требуется повторить ввод (рис. 5). |
| 4 | Сохранение матрицы в файл “matrix.txt” | После ввода матрицы пользователь может сохранить ее в файл. Для этого вводим “1” и указываем название файла (рис. 6). |
| 5 | Сумма двух диагоналей в квадратной матрице | После ввода и сохранения матрицы появляется результат суммирования двух дигоналей. Результат выводится на консоль (рис. 7). |
| 6 | Чтение матрицы из файла “matrix.txt” | При выборе способа ввода матрицы вводим “2” и название файла. Загруженная матрица выводится на консоль (рис. 8). |
| 7 | Выбор действия “Ввести предложение” | После завершения действий над матрицей снова появляется главное меню с выбором действий. Для создания строки вводим “2”, после чего появляется подменю с выбором способа ввода строки (рис. 9). |
| 8 | Вводим слово | Вводим “1”,. Для примера введем предложение: “qwe123” (рис. 10). |
| 9 | Сохранение строки в файл “words.txt” | После ввода строки пользователь может сохранить ее в файл. Для этого вводим “1” и указываем название файла (рис. 11).  Открыв файл, можно убедиться в правильности сохранения данных (рис. 12). |

Продолжение таблицы 6.

|  |  |  |
| --- | --- | --- |
| № | Действие | Результат действия |
| 11 | Чтение строки из файла “words.txt” | При выборе способа ввода строки вводим “2” и название файла. Загруженная строка выводится на консоль (рис. 13).  В случае отсутствия файла выводится соответствующее сообщение об ошибке, программа продолжает работу в штатном режиме (рис. 14). |
| 12 | Проверяем слово на число. | Далее результат о проверке выводится нам на консоль (рис. 15). |
| 13 | Выбор действия “Сложение двух матриц” | После выполнения проверки на число мы нажимаем цифру “3” (рис. 16). Далее мы нажимаем “1”, чтобы заполнить матрицу с помощью консоли (рис. 17). Далее вводим размерность, заполняем матрицы и выводим их (рис. 18). |
| 14 | Сохранение двух матриц в файл. | После ввода мы нажимаем “1”, чтобы сохранить матрицы в файл (рис. 19). После этого выводится суммирование матриц и спрашивают о сохранении суммы. Мы нажимаем “1” и пишем название файла (рис. 20). |
| 15 | Завершение работы программы | При выборе действия в главном меню вводим “4”, после чего выводится соответствующее сообщение, и программа завершает свою работу (рис. 21). |

# C:\Users\Админ\Desktop\Учеба\ООП\3 лабораторная\Program_processing_java\Рис 3.png6 Результаты тестирования

Рисунок 3 – Первоначальный запуск программы.

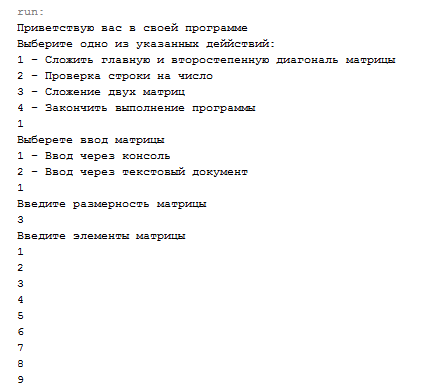
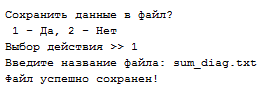
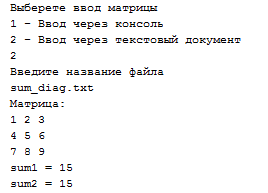
Рисунок 4 - Выбор действия “Сложить главную и второстепенную диагональ матрицы”.

Рисунок 5 - Ручной ввод значений элементов матрицы.

Рисунок 6 - Сохранение матрицы в файл “sum\_diag.txt”.

Рисунок 7 – Данная матрица и суммы их диагоналей.

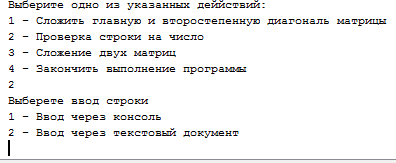


Рисунок 8 - Чтение матрицы из файла “sum\_diag.txt”.

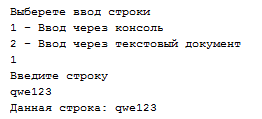
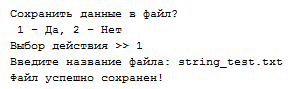
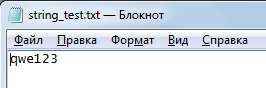
Рисунок 9 - Выбор действия “Проверка строки на число”.

Рисунок 10 – Вводим и выводим строку.



Рисунок 11 - Сохранение строки в файл “string\_text.txt”.

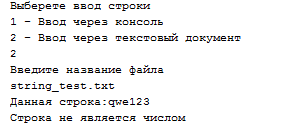
Рисунок 12 – Строка, сохраненная в файле “ string\_text.txt”

Рисунок 13 - Чтение строки из файла “ string\_text.txt”.

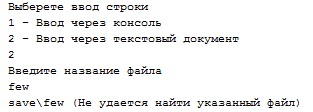
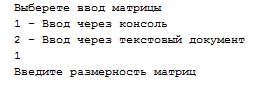
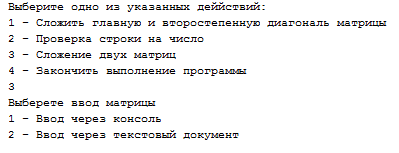
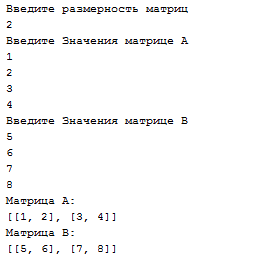


Рисунок 14 – Попытка обратиться к несуществующему файлу.

C:\Users\Админ\Desktop\Учеба\ООП\3 лабораторная\Program_processing_java\Рис 15.png

Рисунок 15 – Результат о проверке на число.

 Рисунок 16 - Выбор действия “Сложение двух матриц”.

 Рисунок 17 – Выбор заполнения ручным вводом.

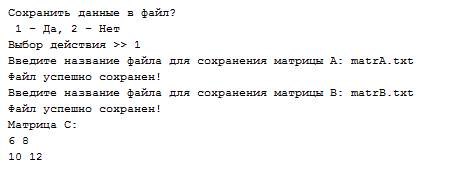
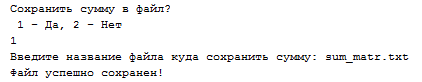
 Рисунок 18 – Заполнение и вывод матриц.

Рисунок 19 – Сохранение двух матриц и их сумма.



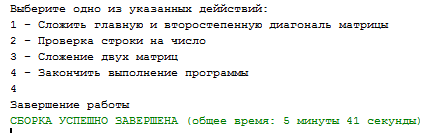
 Рисунок 20 – Сохранение суммы в файл.

Рисунок 21 – Выбор действия “Закончить выполнение программы”.

# 7 Листинг программы

## 7.1 Класс Menu

package program\_processing\_java;

import java.util.Scanner;

import java.util.Arrays;

class menu {

private static Scanner scanner = new Scanner(System.in,"CP1251");

private static Matrix matrix;

private static Addition matrixA;

private static StringNew s1;

private static int [][] matrixB;

public static void start(){

System.out.println("Приветствую вас в своей программе");

int pick;

do{

System.out.println("Выберите одно из указанных деййствий:");

System.out.println("1 - Сложить главную и второстепенную диагональ матрицы");

System.out.println("2 - Проверка строки на число");

System.out.println("3 - Сложение двух матриц");

System.out.println("4 - Закончить выполнение программы");

pick = Control.input(1,4);

switch (pick) {

case 1:

openMatrix();

doMatrix();

break;

case 2:

openString();

doString();

break;

case 3:

openTwoMatrix();

doTwoMatrix();

break;

}

}

while (pick != 4);

System.out.println("Завершение работы");

}

private static void openMatrix() {

System.out.println("Выберете ввод матрицы");

System.out.println("1 - Ввод через консоль");

System.out.println("2 - Ввод через текстовый документ");

switch (Control.input(1,2)){

case 1:

System.out.println("Введите размерность матрицы");

int size = scanner.nextInt();

matrix = new Matrix (size);

System.out.println("Введите элементы матрицы");

for (int i = 0; i<size;i++)

for (int j = 0; j<size;j++)

{

int value = scanner.nextInt();

matrix.setElement(i,j,value);

}

System.out.println("\nСохранить данные в файл?\n 1 - Да, 2 - Нет");

System.out.print("Выбор действия >> ");

if (Control.input(1,2) == 1) {

System.out.print("Введите название файла: ");

scanner = new Scanner(System.in, "CP1251");

WriteReadFile.saveInt(matrix.getMatrix(), "save\\" + scanner.nextLine());

if (WriteReadFile.hasSaving()) System.out.println("Файл успешно сохранен!");

else System.out.println("Ошибка сохранения!");

}

break;

case 2:

System.out.println("Введите название файла");

scanner = new Scanner(System.in, "CP1251");

int [] openFile = WriteReadFile.loadInt("save\\" + scanner.nextLine());

matrix = new Matrix((int) Math.sqrt(openFile.length));

for (int i = 0; i < openFile.length; i++) {

matrix.setElement(i/matrix.size(), i%matrix.size(), openFile[i]);

}

break;

}

}

private static void doMatrix() {

System.out.println("Матрица: ");

for (int i = 0; i < matrix.size(); i++) {

for (int j=0;j<matrix.size();j++) {

System.out.print(matrix.getElement(i, j) + " ");

}

System.out.println();

}

System.out.println("sum1 = " + matrix.sum1());

System.out.println("sum2 = " + matrix.sum2());

}

private static void openString() {

System.out.println("Выберете ввод строки");

System.out.println("1 - Ввод через консоль");

System.out.println("2 - Ввод через текстовый документ");

switch (Control.input(1,2)){

case 1:

System.out.println("Введите строку");

scanner = new Scanner(System.in, "CP1251");

s1 = new StringNew(scanner.nextLine());

System.out.println("Данная строка: " + s1.getString());

System.out.println("\nСохранить данные в файл?\n 1 - Да, 2 - Нет");

System.out.print("Выбор действия >> ");

if (Control.input(1,2) == 1) {

System.out.print("Введите название файла: ");

scanner = new Scanner(System.in, "CP1251");

WriteReadFile.saveString(s1.getString(), "save\\" + scanner.nextLine());

if (WriteReadFile.hasSaving()) System.out.println("Файл успешно сохранен!");

else System.out.println("Ошибка сохранения!");

}

break;

case 2:

System.out.println("Введите название файла");

scanner = new Scanner(System.in, "CP1251");

String openFile = WriteReadFile.loadString("save\\" + scanner.nextLine());

s1 = new StringNew(openFile);

break;

}

}

private static void doString() {

boolean number = s1.isDigitalSrting();

System.out.println("Данная строка:" + s1.getString());

System.out.println(number? "Строка является числом": "Строка не является числом");

}

private static void openTwoMatrix() {

System.out.println("Выберете ввод матрицы");

System.out.println("1 - Ввод через консоль");

System.out.println("2 - Ввод через текстовый документ");

switch (Control.input(1,2)){

case 1:

System.out.println("Введите размерность матриц");

int size = scanner.nextInt();

matrixA = new Addition(size);

System.out.println("Введите Значения матрице А");

for (int i = 0; i<size;i++)

for (int j = 0; j<size;j++)

{

int value = scanner.nextInt();

matrixA.setElement(i,j,value);

}

System.out.println("Введите Значения матрице B");

scanner = new Scanner(System.in, "CP1251");

matrixB = new int[size][size];

for (int i = 0; i<size;i++)

for (int j = 0; j<size;j++)

{

int value = scanner.nextInt();

matrixB[i][j]= value;

}

System.out.println("Матрица А:");

System.out.println(Arrays.deepToString(matrixA.getSquareMatrix()));

System.out.println("Матрица B:");

System.out.println(Arrays.deepToString(matrixB));

System.out.println("\nСохранить данные в файл?\n 1 - Да, 2 - Нет");

System.out.print("Выбор действия >> ");

if (Control.input(1,2) == 1) {

System.out.print("Введите название файла для сохранения матрицы А: ");

scanner = new Scanner(System.in, "CP1251");

WriteReadFile.saveInt(matrixA.getMatrix(), "save\\" + scanner.nextLine());

if (WriteReadFile.hasSaving()) System.out.println("Файл успешно сохранен!");

else System.out.println("Ошибка сохранения!");

System.out.print("Введите название файла для сохранения матрицы В: ");

scanner = new Scanner(System.in, "CP1251");

WriteReadFile.saveInt(new Addition(matrixB).getMatrix(), "save\\" + scanner.nextLine());

if (WriteReadFile.hasSaving()) System.out.println("Файл успешно сохранен!");

else System.out.println("Ошибка сохранения!");

}

break;

case 2:

System.out.println("Введите название файла c матрицей А");

scanner = new Scanner(System.in, "CP1251");

int [] openFile = WriteReadFile.loadInt("save\\" + scanner.nextLine());

matrixA = new Addition((int) Math.sqrt(openFile.length));

for (int i = 0; i < openFile.length; i++) {

matrixA.setElement(i/matrixA.size(), i%matrixA.size(), openFile[i]);

}

System.out.println("Вывод матрицы A:");

System.out.println(Arrays.deepToString(matrixA.getSquareMatrix()));

System.out.println("Введите название файла с Матрицей В ");

scanner = new Scanner(System.in, "CP1251");

int [] openFile2 = WriteReadFile.loadInt("save\\" + scanner.nextLine());

size = (int) Math.sqrt(openFile2.length);

matrixB = new int [size][size];

for (int i = 0; i < openFile2.length; i++) {

matrixB[i/matrixB.length][i%matrixB.length] = openFile2[i];

}

System.out.println("Вывод матрицы B:");

System.out.println(Arrays.deepToString(matrixB));

}

}

private static void doTwoMatrix() {

int [][] matrixC = matrixA.sumTwoMatrix(matrixB);

System.out.println("Матрица C:");

for (int i=0; i<matrixC.length;i++)

{

for (int j=0;j<matrixC.length;j++)

{

System.out.print(matrixC[i][j] + " ");

}

System.out.println();

}

System.out.print("\nСохранить сумму в файл?\n 1 - Да, 2 - Нет \n");

if (Control.input(1,2) == 1) {

System.out.print("Введите название файла куда сохранить сумму: ");

WriteReadFile.saveInt(new Addition(matrixC).getMatrix(), "save\\" + scanner.nextLine());

if (WriteReadFile.hasSaving()) System.out.println("Файл успешно сохранен!");

else System.out.println("Ошибка сохранения!");

}

}

}

7.2 Класс Prorgam\_processing\_java

package program\_processing\_java;

public class Program\_processing\_java {

public static void main(String[] args) {

menu.start();

}

}

7.3 Класс Matrix

package program\_processing\_java;

public class Matrix {

//Поля(Переменные класса)

// Создал внутреннюю переменную(Чтобы ее можно было видеть внутри класса)

private final int [][] matrix;

//получаем элемент матрицы по указанным индексам

public int getElement(int i, int j) {

return matrix[i][j];

}

//Добавляем новый элемент value в матрицу по указанным индексам i и j

public void setElement(int i, int j, int value) {

if (i < matrix.length && j < matrix.length)

matrix[i][j] = value;

}

//Конструктор по умолчанию, который создает новый обхект

public Matrix(){

this(2);// this - вызывает конструктор с параметрами

}

//Возврат длины

public int size() {

return matrix.length;

}

//Конструктор с параметрами

public Matrix(int size){

matrix = new int[size][size];

//this указывает на принадлежность данному классу,является его полем

// Внутренней переменной присвоил значение входного параметра

}

//Конструктор копирования - создает копию существующего объекта

//Присваиваю новому объекту значения из старого

public Matrix(Matrix kopirovan){

int size = kopirovan.matrix.length;

matrix = new int[size][size];

//Передает значения в матрицу нового объекта

for (int i = 0; i < matrix.length; i++) {

for (int j = 0; j < matrix.length; j++) {

matrix[i][j]= kopirovan.matrix[i][j];

}

}

}

//Метод нахождения суммы диагоналей

public int sum1()

{

int sum1 = 0;

for(int i = 0; i<matrix.length; i++)

for(int j = 0; j<matrix.length; j++)

if (i==j)

sum1+=matrix[i][j];

return sum1;

}

public int sum2(){

int sum2 = 0;

for(int i = 0; i<matrix.length; i++)

for(int j = 0; j<matrix.length; j++)

if(j==matrix.length-i-1)

sum2+=matrix[i][j];

return sum2;

}

public int [] getMatrix() {

int size = matrix.length;

int [] tempArray = new int [size \* size];

//пересчет индексов из одномерного массива в двумерный

for (int i = 0; i < size; i++)

for (int j = 0; j < size; j++)

//Копирование массивов из одного массива в другой

tempArray[i\*size+j] = matrix[i][j] ;

return tempArray;

}

}

7.4 Класс StringNew

package program\_processing\_java;

class StringNew {

//Поле

private String s1;

//Констроктор по умолчанию

public StringNew(){

this("-123");

}

//Конструктор с параметром

public StringNew(String s1) {

this.s1 = s1;

}

//Конструктор копирования

public StringNew (StringNew stringNew2){

//Вспомогательный класс StringBuilder - работает со строками

//sb - имя объекта

StringBuilder sb = new StringBuilder();

for (int i = 0; i < stringNew2.s1.length(); i++) {//Прохожусь по старой строке

sb.append(stringNew2.s1.charAt(i)); //Копирую в Sb

}

//С sb в новую строку

s1 = sb.toString();//Приведение к строке

}

public boolean isDigitalSrting(){

boolean number = true;

if((s1.charAt(0)<48 || s1.charAt(0)>57) && s1.charAt(0) != 45) //CharAt - берет символ по указанному индексу

number = false;

if (number)

{

for(int i=1;i<s1.length();i++)// length - длина строки

{

if (s1.charAt(i)<48 || s1.charAt(i)>57) //charAt - берем отдельный символ

{

number = false;

break;

}

}

}

return number;

}

public String getString() {

return s1;

}

}

7.5 Класс Addition

package program\_processing\_java;

public class Addition {

//Поле

private int [][] matrixA;

private final int size;// Создал внутреннюю переменную(Чтобы ее можно было видеть внутри класса)

//Получаем элемент матрицы по указанным индексам

public int getElement(int i, int j){

return matrixA[i][j];

}

//Добавляем новый элемент value в матрицу по указанным индексам i и j

public void setElement(int i,int j, int value){

if (i<size && j <size)

matrixA[i][j]=value;

}

//Конструктор по умолчанию

public Addition (){

this(3);// this - вызывает конструктор с параметрами

}

public Addition(int [][] matrix) {

this(matrix.length);

for (int i = 0; i<size;i++)

for (int j = 0; j<size;j++)

matrixA[i][j]= matrix[i][j];

}

//Конструктор с параметрами

public Addition (int size){

matrixA = new int [size][size];

this.size = size;//this указывает на принадлежность данному классу,является его полем

// Внутренней переменной присвоил значение входного параметра

}

//Конструктор копирования

public Addition(Addition addcopirov){

size=addcopirov.size;

matrixA = new int[size][size];

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

for (int j = 0; j < size; j++) {

matrixA[i][j] = addcopirov.matrixA[i][j];

}

}

}

public int size() {

return size;

}

public int [][] sumTwoMatrix(int [][] matrixB) {

int [][] matrixC = new int [matrixA.length][matrixA.length]; //Беру у любой матрицы ее размер(В данном случае матрицы А)

if (matrixA.length== matrixB.length)

{

int count=matrixA.length;//Cound - размер матрицы

for (int i=0; i<count;i++)

for (int j=0;j<count;j++)

matrixC[i][j]=matrixA[i][j]+matrixB[i][j];

}

else

System.out.println("Ошибка");

return matrixC;

}

public int [] getMatrix() {

int size = matrixA.length;

int [] tempArray = new int [size \* size];

//пересчет индексов из одномерного массива в двумерный

for (int i = 0; i < size; i++)

for (int j = 0; j < size; j++)

//Копирование массивов из одного массива в другой

tempArray[i\*size+j] = matrixA[i][j] ;

return tempArray; }

public int [][] getSquareMatrix() {

return matrixA; }

}

7.6 Класс WriteReadFale.

package program\_processing\_java;

import java.io.\*;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

public class WriteReadFile {

private static boolean isSaving;

public static void saveInt( int [] array, String fullname ){

//Обрабатываемый ресурс

try(DataOutputStream dos = new DataOutputStream(new FileOutputStream(fullname))){

for (int i=0; i<array.length;i++)

dos.writeInt(array[i]);

isSaving = true;

}

//Улавливаемое исключение

catch(IOException exc){

System.out.println("Не удалось сохранить данные в файл");

System.out.println(exc.getLocalizedMessage());

isSaving = false;

}

}

public static boolean hasSaving(){

return isSaving;

}

public static int [] loadInt(String fullname){

try(DataInputStream dis = new DataInputStream(new FileInputStream(fullname))){

int [] tempArray = new int [dis.available() / 4];

for(int i = 0; i < tempArray.length; i++)

tempArray[i] = dis.readInt();

return tempArray;

}

catch (IOException exc) {

System.out.println(exc.getLocalizedMessage());

System.out.println("Не удалось загрузить данные");

}

return null;

}

public static void saveString(String saved, String fullname) {

try (OutputStreamWriter osw = new OutputStreamWriter(new FileOutputStream(fullname))) {

osw.write(saved);

isSaving = true;

} catch (IOException exception) {

System.out.println(exception.getLocalizedMessage());

isSaving = false;

}

}

public static String loadString(String fullname) {

StringBuilder builder = new StringBuilder();

try (InputStreamReader reader = new InputStreamReader(new FileInputStream(fullname))) {

while (reader.ready()) builder.append((char) reader.read());

} catch (IOException exception) {

System.out.println(exception.getLocalizedMessage());

} return builder.toString();

}

}

7.7 Класс Control.

package program\_processing\_java;

import java.util.Scanner;

class Control {

public static int input(int firstBorder, int endBorder){

int put = firstBorder;

do {

Scanner scanner = new Scanner (System.in, "CP1251");

if (!scanner.hasNextInt())

System.out.print("Неправильный ввод. Введите значение еще раз");

else {

put = scanner.nextInt();

if (put<firstBorder || endBorder<put)

System.out.print("Неправильный ввод. Введите значение еще раз");

}

}

while (put<firstBorder || endBorder < put);

return put;

}

}

# Список литературы

1. API Specification for the Java 7 SE. [официальный сайт] URL:

<http://docs.oracle.com/javase/7/docs/api> (дата обращения: 10.10.2020)

2. The Java Tutorials. SE [электронный ресурс], URL:

<http://docs.oracle.com/javase/tutorial/index.html> (дата обращения: 12.10.2020)

3. Файн Я. – Программирование на Java для детей, родителей, дедушек и бабушек, 2011.

4. Васильев А. Н. Java. Объектно-ориентированное программирование: для магистров и бакалавров. Базовый курс по объектно-ориентированному

программированию / А. Н. Васильев. – СПб.: Питер, 2012. – 395 с.

5. Отношения классов — от UML к коду [Электронный ресурс] // Хабра-хабр: [сайт]. [2006]. URL: <http://habrahabr.ru/post/150041> (дата обращения: 13.10.2020)