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 №13 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | ЭВМб-19-1 |  |  |  | К. И. Храмцов |
|  |  | шифр |  | подпись |  | И. О. Фамилия |
| Проверил |  |  |  |  |  | Т. В. Маланова |
|  |  | должность |  | подпись |  | И. О. Фамилия |
| Работа защищена с оценкой | | | |  | | |

Иркутск 2020 г.

Содержание

[1 Постановка задачи 3](#_Toc53618352)

[2 Структура меню 4](#_Toc53618353)

[3 Проектирование классов 6](#_Toc53618354)

[4 Спецификации 7](#_Toc53618355)

[5 Проектирование тестов 11](#_Toc53618356)

[6 Результаты тестирования 13](#_Toc53618357)

[7 Листинг программы 18](#_Toc53618358)

[7.1 Класс ConsoleMenu 18](#_Toc53618359)

[7.2 Класс Program 20](#_Toc53618360)

[7.3 Класс Matrix 21](#_Toc53618361)

[7.4 Класс Sentence 23](#_Toc53618362)

[7.5 Класс SaveLoader 26](#_Toc53618363)

[7.6 Класс InputController 27](#_Toc53618364)

[Список литературы 28](#_Toc53618365)

# 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) Дана квадратная матрица. Необходимо найти минимальный элемент данной квадратной матрицы.

b) Дана строка, содержащая слова, разделенные пробелами и запятыми. Необходимо развернуть указанное слово справа налево.

c) Дано предложение (строка, разделенная пробелами). Необходимо удалить в предложении каждое третье слово.

# 2 Структура меню

На рис. 1 представлена схема структуры пользовательского меню.

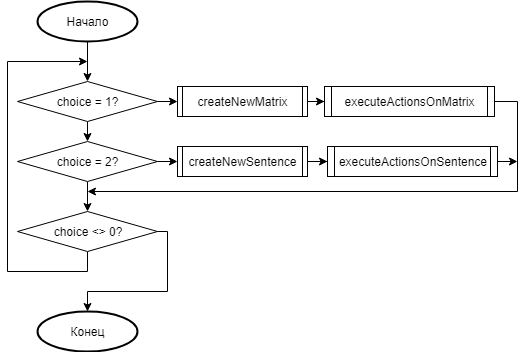


Рисунок 1 – Схема структуры пользовательского меню

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

*Описание методов класса ConsoleMenu:*

1. Метод **createNewMatrix –** статический метод, отвечающий за создание объекта класса **Matrix**. Пользователю доступны 3 способа заполнения матрицы: ручной ввод с клавиатуры, заполнение случайными числами и чтение матрицы из файла.

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

3. Метод **executeActionsOnMatrix –** статический метод, отвечающий за работу над объектом класса **Matrix**. Предоставляет пользователю на выбор 2 действия: поиск мин. и макс. элементов матрицы.

4. Метод **executeActionsOnSentence –** статический метод, отвечающий за работу над объектом класса **Sentence**. Предоставляет пользователю на выбор 2 действия: переворачивание слова по указанному номеру и удаление каждого 3-го слова в предложении.

5. Метод **saveIntToFile** – статический метод, отвечающий за побайтовое сохранение числовых данных в файл с помощью пользовательского класса **SaveLoader**.

6. Метод **saveStringToFile** – статический метод, отвечающий за посимвольное сохранение строковых данных в файл с помощью пользовательского класса **SaveLoader**.

# 3 Проектирование классов

На рис. 2 представлена схема структуры классов.

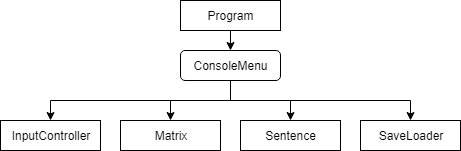


Рисунок 2 – Схема структуры классов

*Описание классов:*

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

2. Класс **ConsoleMenu –** 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 **Sentence –** класс-обертка, предоставляющий дополнительные методы для работы с массивом строк и обеспечивающий решение 2-го и 3-го индивидуальных заданий.

5. Класс **SaveLoader –** вспомогательныйкласс, отвечающий за работу с потоками ввода/вывода данных.

6. Класс **InputController –** 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 | [-1000; 1000] |
| 2 | size | Размер матрицы | int | private | [1; 10] |
| 3 | pointer | Указатель на текущий элемент | int | private | [0; 100) |
| Класс Sentence | | | | | |
| 4 | sentence | Массив слов | String [] | private | ‘A’ .. ‘z’,  ‘А’ .. ‘я’,  ‘0’ .. ‘9’ |
| 5 | count | Количество слов | int | private | [1; 100] |
| 6 | capacity | Вместимость массива слов | int | private | [1; 100] |
| Класс ConsoleMenu | | | | | |
| 7 | scanner | Объект класса Scanner, отвечающий за ввод данных с клавиатуры | Scanner | private | - |
| 8 | matrix | Объект класса Matrix | Matrix | private | - |
| 9 | sentence | Объект класса Sentence | Sentence | private | - |
| Класс SaveLoader | | | | | |
| 10 | UTF\_8 | Задает кодировку UTF-8 | Charset | private | UTF-8 |

Таблица 2 – Описание методов класса InputController

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

Таблица 3 – Описание методов класса Matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип данных | Тип доступа | Параметры |
| 1 | Matrix | Конструктор  по умолчанию | - | public | - |
| 2 | Matrix | Конструктор с параметрами | - | public | *int size* – размерность матрицы |
| 3 | Matrix | Конструктор копирования | - | public | *Matrix other* – копируемый объект |
| 4 | getMatrix | Возвращает матрицу (геттер) | int [] | public | *-* |
| 5 | getSize | Возвращает размер матрицы (геттер) | int | public | *-* |
| 6 | add | Добавление нового элемента | void | public | *int value* – значение нового элемента |
| 7 | elementAt | Возвращает элемент по указанным индексам | int | public | *int row* –  индекс строки  *int column –* индекс столбца |
| 8 | fillRandom | Заполняет массив случайными числами | void | public | *int bound* – пограничное значение |
| 9 | minElement | Возвращает мин. элемент в матрице | int | public | *-* |
| 10 | maxElement | Возвращает макс. элемент в матрице | int | public | *-* |
| 11 | findElement | Поиск нужного элемента, согласно бин. оператору | int | private | *BinaryOperator <Integer> opetator –* бин. оператор |
| 12 | toString | Возвращает объект класса в виде строки | String | public | *-* |

Таблица 4 – Описание методов класса Sentence

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип данных | Тип доступа | Параметры |
| 1 | Sentence | Конструктор по умолчанию | - | public | - |
| 2 | Sentence | Конструктор с параметрами | - | public | int capacity - вместимость |
| 3 | Sentence | Конструктор с параметрами | - | public | *String str* – строка слов, введенных через символ-разделитель  *String splitter -* разделитель |
| 4 | Sentence | Конструктор копирования | - | public | *Sentence other* – копируемый объект |
| 5 | getCount | Получить длину массива (геттер) | int | public | - |
| 6 | getSentence | Получить массив слов (геттер) | String [] | public | - |
| 7 | elementAt | Получить слово по указанному индексу в массиве | String | public | *int index -*  индекс слова |
| 8 | removeAt | Удалить слово по указанному индексу в массиве | boolean | public | *int index -*  индекс слова |
| 9 | deleteChar | Удаляет ненужные символы из строки слов, такие как повторяющиеся запятые и лишние отступы (пробелы) | void | private | *StringBuilder builder –* редактируемая строка  *String deleted* – удаляемая подстрока |

Продолжение таблицы 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип данных | Тип доступа | Параметры |
| 10 | addElement | Добавление новой строки в существующий массив слов | void | public | *String word* – добавляемое слово |
| 11 | format | Возвращает форматированную строку слов | String | public | *String splitter* - разделитель |
| 12 | reverseAt | Переворачивает строку по указанному номеру | void | public | *int number* – номер указанного слова |
| 13 | removeEach  ThirdWord | Удаление каждого 3-го слова из массива слов | void | public | *-* |

Таблица 5 – 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 path* – название файла |
| 2 | loadString | Чтение строковых данных из файла | String | public | *String path* – название файла |
| 3 | saveInt | Запись числовых данных в файл | void | public | *int [] saved* – сохраняемый массив  *String path* – название файла |
| 4 | loadInt | Чтение числовых данных из файла | int [] | public | *String path* – название файла |

# 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 “Ввести квадратную матрицу” | 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 | Поиск мин. элемента в квадратной матрице | После ввода и сохранения матрицы появляется подменю с выбором действий над матрицей. Для поиска мин. элемента вводим “1”. Результат выводится на консоль (рис. 7). |
| 6 | Чтение матрицы из файла “matrix.txt” | При выборе способа ввода матрицы вводим “3” 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 | Ввод слов через пробел | Вводим “2”, после чего вводим слова через пробел. Для примера введем предложение: “Квадрат гипотенузы равен сумме квадратов катетов” (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). |
| 10 | Удаление каждого 3-го слова в предложении | После ввода и сохранения матрицы появляется подменю с выбором действий над строкой. Для удаления каждого 3-го слова вводим “2”. Измененная строка выводится на консоль (рис. 13). |

Продолжение таблицы 6.

|  |  |  |
| --- | --- | --- |
| № | Действие | Результат действия |
| 11 | Чтение строки из файла “words.txt” | При выборе способа ввода строки вводим “3” и название файла. Загруженная строка выводится на консоль (рис. 14).  В случае отсутствия файла выводится соответствующее сообщение об ошибке, программа продолжает работу в штатном режиме (рис. 15). |
| 12 | Переворачивание слова по указанному номеру | Для переворачивания слова в строке при выборе действия над строкой вводим “1”. После чего вводим номер переворачиваемого слова. Измененная строка выводится на консоль (рис. 16). |
| 13 | Заполнение матрицы случайными числами | При выборе способа ввода матрицы вводим “2” и указываем размер матрицы. После ответа на вопрос: “Сохранить данные файл?” сгенерированная матрица выводится в консоль (рис. 17). |
| 14 | Завершение работы программы | При выборе действия в главном меню вводим “0”, после чего выводится соответствующее сообщение и программа завершает свою работу (рис. 18). |

# 6 Результаты тестирования

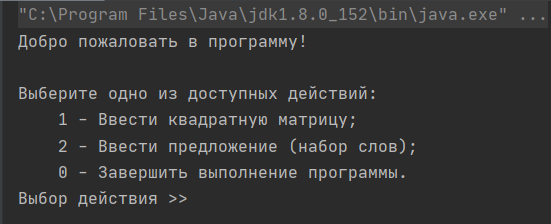


Рисунок 3 – Первоначальный запуск программы

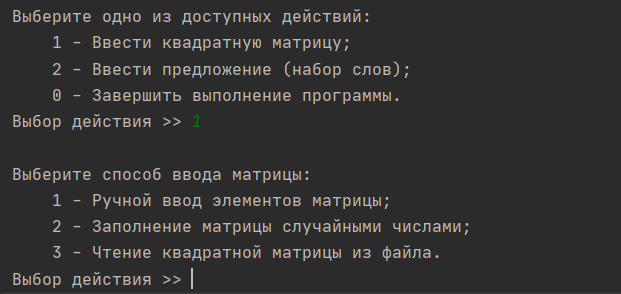


Рисунок 4 - Выбор действия “Ввести квадратную матрицу”

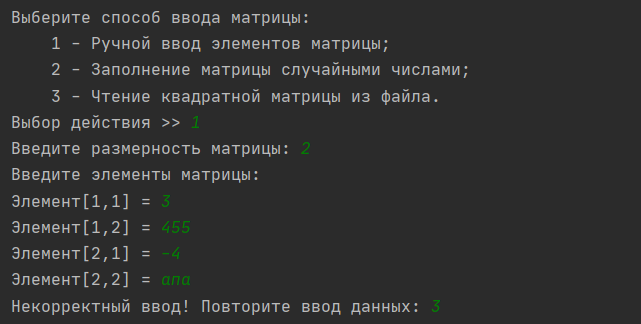


Рисунок 5 - Ручной ввод значений элементов матрицы

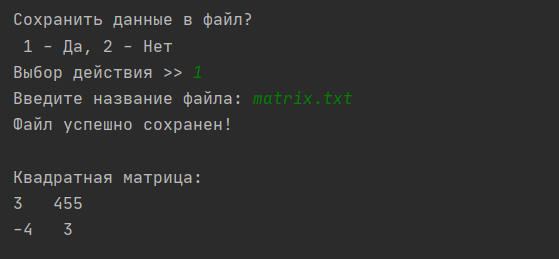


Рисунок 6 - Сохранение матрицы в файл “matrix.txt”

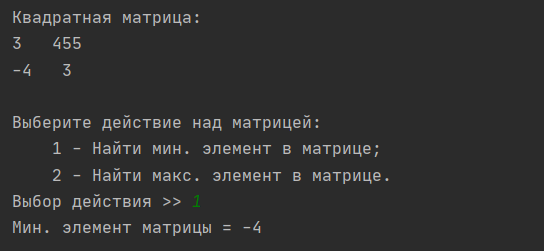


Рисунок 7 - Поиск мин. элемента в квадратной матрице

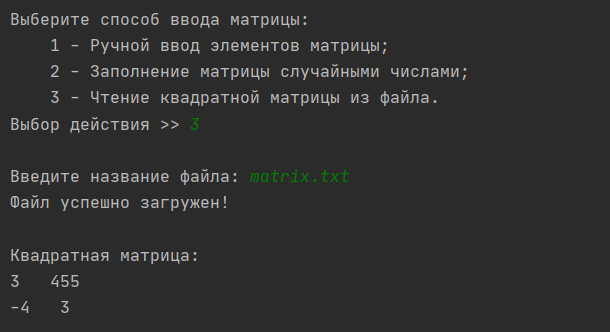


Рисунок 8 - Чтение матрицы из файла “matrix.txt”

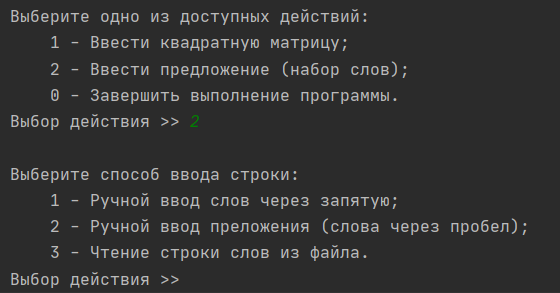


Рисунок 9 - Выбор действия “Ввести предложение”

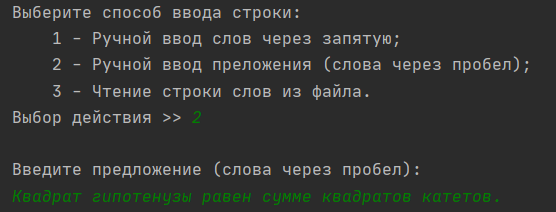


Рисунок 10 - Ввод слов через пробел

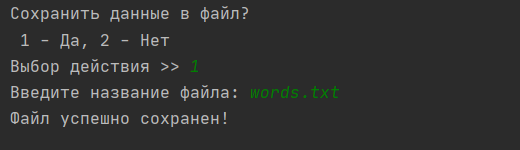


Рисунок 11 - Сохранение строки в файл “words.txt”

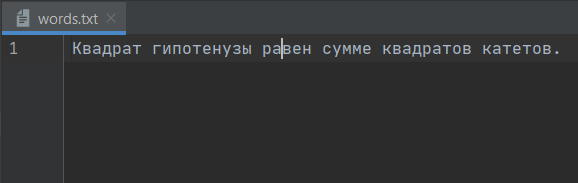


Рисунок 12 – Строка, сохраненная в файле “words.txt”

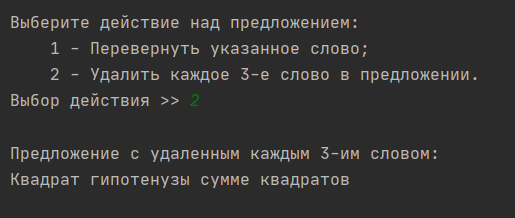


Рисунок 13 - Удаление каждого 3-го слова в предложении

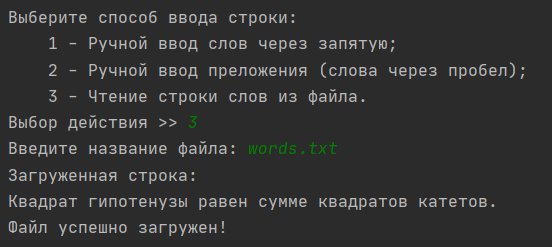


Рисунок 14 - Чтение строки из файла “words.txt”

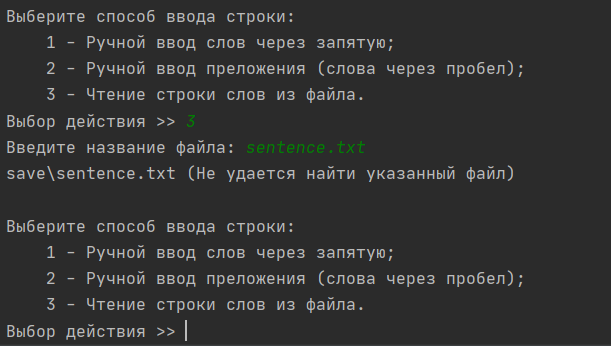


Рисунок 15 – Попытка обратиться к несуществующему файлу

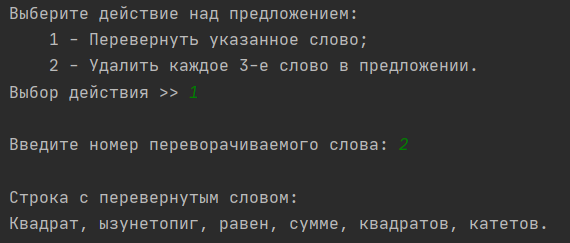


Рисунок 16 - Переворачивание слова по указанному номеру

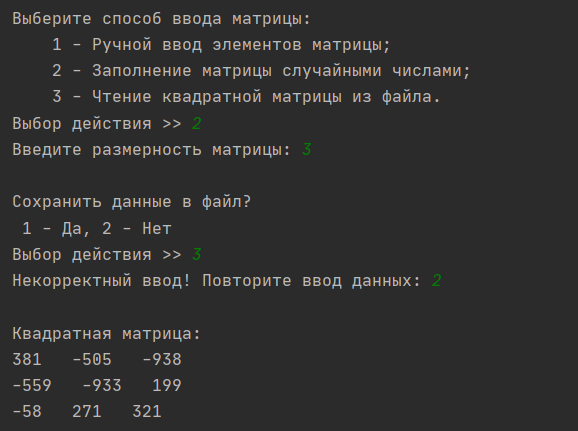


Рисунок 17 - Заполнение матрицы случайными числами

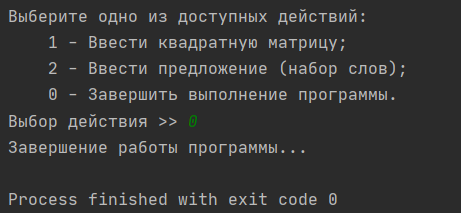


Рисунок 18 – Завершение работы программы

# 7 Листинг программы

## 7.1 Класс ConsoleMenu

import java.util.Scanner;

import lab.Matrix;

import lab.Sentence;

import io.InputController;

import io.SaveLoader;

public class ConsoleMenu {

private static final Scanner scanner = new Scanner(System.in);

private static Matrix matrix;

private static Sentence sentence;

public static void start() {

System.out.println("Добро пожаловать в программу!");

int choice;

do {

System.out.println("\nВыберите одно из доступных действий:");

System.out.println("\t1 - Ввести квадратную матрицу;");

System.out.println("\t2 - Ввести предложение (набор слов);");

System.out.println("\t0 - Завершить выполнение программы.");

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

choice = InputController.input(0, 2);

switch (choice) {

case 1:

createNewMatrix();

executeActionsOnMatrix();

break;

case 2:

createNewSentence();

executeActionsOnSentence();

break;

}

} while (choice != 0);

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

}

private static void createNewMatrix() {

System.out.println("\nВыберите способ ввода матрицы:");

System.out.println("\t1 - Ручной ввод элементов матрицы;");

System.out.println("\t2 - Заполнение матрицы случайными числами;");

System.out.println("\t3 - Чтение квадратной матрицы из файла.");

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

switch (InputController.input(1, 3)) {

case 1:

System.out.print("Введите размерность матрицы: ");

matrix = new Matrix(InputController.input(1, 10));

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

for (int row = 0; row < matrix.getSize(); row++) {

for (int column = 0; column < matrix.getSize(); column++) {

System.out.print("Элемент[" + (row + 1) + "," + (column + 1) + "] = " );

matrix.add(InputController.input(-1000, 1000));

}

} saveIntToFile(matrix.getMatrix());

break;

case 2:

System.out.print("Введите размерность матрицы: ");

matrix = new Matrix(InputController.input(1, 10));

matrix.fillRandom(1000);

saveIntToFile(matrix.getMatrix());

break;

case 3:

System.out.print("\nВведите название файла: ");

int [] tempArray = SaveLoader.loadInt("save\\" + scanner.nextLine());

if (tempArray != null) {

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

for (int element : tempArray) matrix.add(element);

System.out.println("Файл успешно загружен!");

} else createNewMatrix();

break;

} System.out.println("\nКвадратная матрица:\n" + matrix);

}

private static void createNewSentence() {

System.out.println("\nВыберите способ ввода строки:");

System.out.println("\t1 - Ручной ввод слов через запятую;");

System.out.println("\t2 - Ручной ввод преложения (слова через пробел);");

System.out.println("\t3 - Чтение строки слов из файла.");

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

switch (InputController.input(1, 3)) {

case 1:

System.out.println("\nВведите слова через запятую:");

sentence = new Sentence(scanner.nextLine(), ", ");

saveStringToFile(sentence.format(" "));

break;

case 2:

System.out.println("\nВведите предложение (слова через пробел):");

sentence = new Sentence(scanner.nextLine(), " ");

saveStringToFile(sentence.format(" "));

break;

case 3:

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

String tempString = SaveLoader.loadString("save\\" + scanner.nextLine());

if (!tempString.isEmpty()) {

sentence = new Sentence(tempString, " ");

System.out.println("Загруженная строка:\n" + sentence.format(" "));

System.out.println("Файл успешно загружен!");

} else createNewSentence();

break;

}

}

private static void executeActionsOnMatrix() {

System.out.println("\nВыберите действие над матрицей:");

System.out.println("\t1 - Найти мин. элемент в матрице;");

System.out.println("\t2 - Найти макс. элемент в матрице.");

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

if (InputController.input(1, 2) == 1)

System.out.println("Мин. элемент матрицы = " + matrix.minElement());

else System.out.println("Макс. элемент матрицы = " + matrix.maxElement());

}

private static void executeActionsOnSentence() {

System.out.println("\nВыберите действие над предложением:");

System.out.println("\t1 - Перевернуть указанное слово;");

System.out.println("\t2 - Удалить каждое 3-е слово в предложении.");

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

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

case 1:

System.out.print("\nВведите номер переворачиваемого слова: ");

sentence.reverseAt(InputController.input(1, sentence.getCount()));

System.out.println("\nСтрока с перевернутым словом:");

System.out.println(sentence.format(", "));

break;

case 2:

sentence.removeEachThirdWord();

System.out.println("\nПредложение с удаленным каждым 3-им словом:");

System.out.println(sentence.format(" "));

break;

}

}

private static void saveStringToFile(String sentence) {

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

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

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

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

SaveLoader.saveString(sentence, "save\\" + scanner.nextLine());

if (SaveLoader.hasSaving())

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

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

}

}

private static void saveIntToFile(int [] array) {

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

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

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

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

SaveLoader.saveInt(array, "save\\" + scanner.nextLine());

if (SaveLoader.hasSaving())

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

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

}

}

}

## 7.2 Класс Program

public class Program {

public static void main(String[] args) { ConsoleMenu.start(); }

}

## 7.3 Класс Matrix

package lab;

import java.util.Random;

import java.util.function.BinaryOperator;

public class Matrix {

private final int [][] matrix;

private final int size;

private int pointer;

// Конструктор по умолчанию

public Matrix() { this(10); }

// Конструктор с параметрами

public Matrix(int size) {

this.size = size;

pointer = 0;

matrix = new int[size][size];

}

// Конструктор копирования

public Matrix(Matrix other) {

size = other.size;

pointer = other.pointer;

matrix = new int[size][size];

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

System.arraycopy(other.matrix[i], 0, matrix[i], 0, size);

}

// Получить размерность матрицы

public int getSize() { return size; }

public String getMatrix() {

StringBuilder builder = new StringBuilder();

for (int [] array : matrix)

for (int element : array)

builder.append(element).append(" ");

return builder.toString();

}

// Добавление нового элемента

public void add(int value) {

if (pointer < size \* size) {

matrix[pointer / size][pointer % size] = value;

pointer++;

}

}

// Получение элемента по индексам

public int elementAt(int row, int column) {

if (row < 0 || row >= size || column < 0 || column >= size)

throw new IndexOutOfBoundsException("Выход за границы массива!");

return matrix[row][column];

}

// Заполнение матрицы случайними числами

public void fillRandom(int bound) {

Random random = new Random();

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

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

matrix[row][column] = random.nextInt() % bound;

}

// Поиск макс. элемента матрицы

public int maxElement() { return findElement(Math::max); }

// Поиск мин. элемента матрицы

public int minElement() { return findElement(Math::min); }

private int findElement(BinaryOperator<Integer> operator) {

int found = matrix[0][0];

for (int [] array : matrix)

for (int element : array)

found = operator.apply(element, found);

return found;

}

@Override

public String toString() {

StringBuilder builder = new StringBuilder();

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

for (int element : matrix[row])

builder.append(element).append(" ");

if (row != size - 1) builder.append('\n');

}

return builder.toString();

}

}

## 7.4 Класс Sentence

package lab;

import java.util.ArrayList;

import java.util.List;

public class Sentence {

private String [] sentence;

private int count;

private int capacity;

// Конструктор по умолчанию

public Sentence() { this(10); }

// Конструктор с параметрами

public Sentence(int capacity) {

if (capacity < 0 || capacity > 1000)

throw new IllegalArgumentException("Размер массива не может быть отрицательным!");

this.capacity = capacity;

sentence = new String[capacity];

count = 0;

}

// Конструктор с параметрами

public Sentence(String str, String splitter) {

StringBuilder builder = new StringBuilder(str);

for (String deleted : new String [] { " ", " ,", ",," })

deleteChar(builder, deleted);

String[] tmpSentence = builder.toString().split(splitter);

count = tmpSentence.length;

capacity = count \* 2;

sentence = new String[capacity];

System.arraycopy(tmpSentence, 0, this.sentence, 0, count);

}

// Конструктор копирования

public Sentence(Sentence other) {

sentence = new String [other.count];

this.count = other.count;

this.capacity = other.capacity;

System.arraycopy(other.sentence, 0, this.sentence, 0, other.count);

}

// Получить длину массива

public int getCount() { return count; }

// Получить массив слов

public String [] getSentence() {

String [] tempSentence = new String[count];

System.arraycopy(sentence, 0, tempSentence, 0, count);

return tempSentence;

}

// Получить элемент по указанному индексу

public String elementAt(int index) {

if (index < 0 || index >= count)

throw new IndexOutOfBoundsException("Выход за границы массива!");

return sentence[index];

}

public void addElement(String word) {

if (count >= capacity) {

String [] tmpSentence = new String[capacity \*= 2];

System.arraycopy(sentence, 0, tmpSentence, 0, sentence.length);

tmpSentence[count] = word;

sentence = tmpSentence;

} else sentence[count] = word;

count++;

}

public boolean removeAt(int index) {

if (index < 0 || index >= count) return false;

String [] tmpSentence = new String[--count];

System.arraycopy(sentence, 0, tmpSentence, 0, index);

System.arraycopy(sentence, index + 1, tmpSentence, index, count - index);

sentence = tmpSentence; return true;

}

// Удаление 1-го символа в указанной подстроке

private void deleteChar(StringBuilder builder, String deleted) {

int index = builder.indexOf(deleted);

while (index != -1) {

builder.deleteCharAt(index);

index = builder.indexOf(deleted);

}

}

// Форматирование строки

public String format(String splitter) {

StringBuilder builder = new StringBuilder();

for (int index = 0; index < getCount() - 1; index++)

builder.append(sentence[index]).append(splitter);

builder.append(sentence[getCount() - 1]);

return builder.toString();

}

// Переворачивание слова

public void reverseAt(int number) {

StringBuilder builder = new StringBuilder(sentence[number - 1]);

sentence[number - 1] = builder.reverse().toString();

}

// Удаление каждого 3-го слова из предложения

public void removeEachThirdWord() {

List<String> list = new ArrayList<>();

for (int index = 0; index < getCount(); index++)

if ((index + 1) % 3 != 0) list.add(sentence[index]);

sentence = list.toArray(new String[0]);

count = sentence.length;

}

}

## 7.5 Класс SaveLoader

package io;

import java.io.\*;

import java.nio.charset.Charset;

import java.nio.charset.StandardCharsets;

public class SaveLoader {

private static final Charset UTF\_8 = StandardCharsets.UTF\_8;

private static boolean isSaving;

public static boolean hasSaving() {

return isSaving;

}

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

try (OutputStreamWriter writer = new OutputStreamWriter(new FileOutputStream(path), UTF\_8)) {

writer.write(saved);

isSaving = true;

} catch (IOException exception) {

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

isSaving = false;

}

}

public static String loadString(String path) {

StringBuilder builder = new StringBuilder();

try (InputStreamReader reader = new InputStreamReader(new FileInputStream(path), UTF\_8)) {

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

} catch (IOException exception) {

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

} return builder.toString();

}

public static void saveInt(int [] saved, String path) {

try (DataOutputStream outputStream = new DataOutputStream(new FileOutputStream(path))) {

for (int element : saved) outputStream.writeInt(element);

isSaving = true;

} catch (IOException exception) {

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

isSaving = false;

}

}

public static int [] loadInt(String path) {

try (DataInputStream inputStream = new DataInputStream(new FileInputStream(path))) {

int [] tempArray = new int [inputStream.available() / Integer.BYTES];

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

tempArray[index] = inputStream.readInt();

return tempArray;

} catch (IOException exception) {

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

} return null;

}

}

## 7.6 Класс InputController

package io;

import java.util.Scanner;

public class InputController {

public static int input(int leftLimit, int rightLimit) {

int value = leftLimit - 1;

String error = "Некорректный ввод! Повторите ввод данных: ";

do {

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

if (!scanner.hasNextInt()) System.out.print(error);

else {

value = scanner.nextInt();

if (value < leftLimit || value > rightLimit)

System.out.print(error);

}

} while (value < leftLimit || value > rightLimit);

return value;

}

}

# 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)