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 |
| Кафедра вычислительной техники |
| наименование кафедры |

|  |
| --- |
| **Отчет** |
| по лабораторной работе №3 по дисциплине «Работа с потоками данных и обработка исключений» |
| наименование темы  Вариант №20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | АСУб-19-1 |  |  |  | Даниленко А.С. |
|  |  | шифр |  | подпись |  | И.О. Фамилия |
| Проверил |  |  |  |  |  | Маланова Т.В. |
|  |  |  |  | подпись |  | И.О. Фамилия |
| Работа защищена с оценкой | | | |  | | |

Иркутск 2020 г.

**Содержание**

[1 Постановка задачи 3](#_Toc55424417)

[2 Описание структуры пользовательского меню 4](#_Toc55424418)

[3 Проектирование классов 5](#_Toc55424419)

[4 Описание методов классов и спецификация локальных переменных 6](#_Toc55424420)

[5 Таблица тестов 9](#_Toc55424421)

[6 Результаты тестирования 10](#_Toc55424422)

[7 Исходный код 12](#_Toc55424423)

[Список использованных источников 22](#_Toc55424424)

1 Постановка задачи

1. Дана матрица. Необходимо сгенерировать новую матрицу, элементы не первой строки которой равны произведению соответствующих элементов исходной матрицы на первый элемент столбца.

2. Вычислите значение арифметического выражения. Два операнда и знак операции задаются в командной строке при запуске программы.

3. Выберите все слова из строки, содержащей слова, разделенные пробелами и запятыми, которые полностью состоят из одного символа.

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 выводы 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.

2 Описание структуры пользовательского меню

Меню хранит в себе 3 ссылки на объекты для каждой из задач. Из главного меню main по выбору осуществляется вход в меню подзадачи matrixField, calculatorField или wordsField. В меню подзадач доступны такие действия, как: ввод данных, решение задачи, сохранение данных, загрузка данных. Выбор действий реализуется посредством ввода номера, соответствующего определённому действию. Все поля и методы статичны.

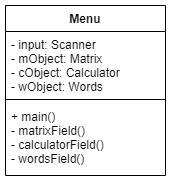


Рисунок 1 – Структура класса пользовательского меню

3 Проектирование классов

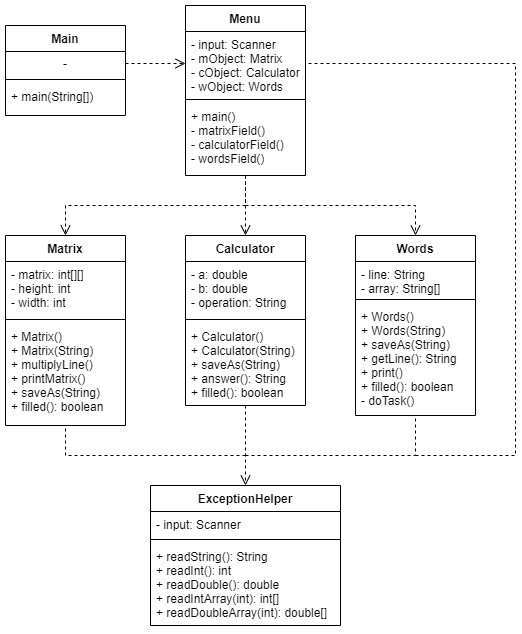


Рисунок 2 – Иерархия и структура классов

4 Описание методов классов и спецификация локальных переменных

Таблица 1 – Описание методов класса Main

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| main | Точка входа в программу. Вызывает использование класса меню. | - | public static | - |

Таблица 2 – Описание методов класса Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| Matrix | Конструктор по умолчанию. Запрашивает данные для полей у пользователя. | Matrix | public | - |
| Matrix | Конструктор копирования. Восстанавливает объект из файла по переданному имени. | Matrix | public | String name |
| multiplyLine | Перемножает указанную строку матрицы на первую. | - | public | - |
| printMatrix | Выводит матрицу в консоль. | - | public | - |
| saveAs | Сохраняет объект в файл с переданным именем. | - | public | String name |
| filled | Функция проверки заполненности полей объекта. | boolean | public | - |

Таблица 3 – Описание полей класса Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Модификаторы |
| matrix | Двумерный массив значений матрицы | int[][] | private |
| height | Высота матрицы | int | private |
| width | Ширина матрицы | int | private |

Таблица 4 – Описание методов класса Calculator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| Calculator | Конструктор по умолчанию. Запрашивает данные для полей у пользователя. | Calculator | public | - |
| Calculator | Конструктор копирования. Восстанавливает объект из файла по переданному имени. | Calculator | public | String name |
| saveAs | Сохраняет объект в файл с переданным именем. | - | public | String name |
| answer | Функция создания строки, включающей произведённые вычисления. | String | public | - |
| filled | Функция проверки заполненности полей объекта. | boolean | public | - |

Таблица 5 – Описание полей класса Calculator

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Модификаторы |
| a | Первый операнд | double | private |
| b | Второй операнд | double | private |
| operation | Оператор | String | private |

Таблица 6 – Описание методов класса Words

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| Words | Конструктор по умолчанию. Запрашивает данные для полей у пользователя. | Words | public | - |
| Words | Конструктор копирования. Восстанавливает объект из файла по переданному имени. | Words | public | String name |
| saveAs | Сохраняет объект в файл с переданным именем. | - | public | String name |
| getLine | Возвращает необработанную строку | String | public | - |
| doTask | Обрабатывает строку в массив слов | - | private | - |
| print | Выводит обработанную строку | - | public | - |
| filled | Функция проверки заполненности полей объекта. | boolean | public | - |

Таблица 7 – Описание полей класса Words

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Модификаторы |
| line | Необработанная строка | String | private |
| array | Обработанная строка | String[] | private |

Таблица 8 – Описание методов класса Menu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| main | Главное меню, осуществляет доступ в меню подзадач. | - | public static | - |
| matrixField | Меню первой подзадачи | - | public static | - |
| calculatorField | Меню второй подзадачи | - | public static | - |
| wordsField | Меню третьей подзадачи | - | public static | - |

Таблица 9 – Описание полей класса Menu

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Модификаторы |
| mObject | Объект, решающий 1 задачу | Matrix | private static |
| cObject | Объект, решающий 2 задачу | Calculator | private static |
| wObject | Объект, решающий 3 задачу | Words | private static |

Таблица 10 – Описание методов класса ExceptionHelper

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| readString | Запрашивает у пользователя строку. При вводе пустой строки повторяет запрос. | String | public static | - |
| readInt | Запрашивает у пользователя целое число. При неправильном вводе повторяет запрос. | int | public static | - |
| readDouble | Запрашивает у пользователя число. При неправильном вводе повторяет запрос. | double | public static | - |
| readIntArray | Запрашивает у пользователя набор целых чисел. При неправильном вводе повторяет запрос. | int[] | public static | int size |
| readDoubleArray | Запрашивает у пользователя набор чисел. При неправильном вводе повторяет запрос. | double[] | public static | int size |

5 Таблица тестов

Таблица 11 – Тесты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Задание | Вход | Выход | Примечания |
| 1 | Проверка меню и попытка сохранения пустой матрицы | 1  4 | “Матрица не определена” | Произошёл переход к меню первой задачи. Матрица не определена, а потому в её сохранении отказано. |
| 2 | Сохранение матрицы в файл | 1  4  foursix | Ваша матрица:  1 2 3 4 5 6  6 5 4 3 2 1  2 2 2 2 2 2  3 3 3 3 3 3 | Данные сохранены в файл foursix.mx |
| 3 | Обработка задачи | 5  3  1 | Ваша матрица:  1 2 3 4 5 6  6 5 4 3 2 1  2 4 6 8 10 12  3 3 3 3 3 3 | Задача была решена, что видно из изменившихся значений матрицы |
| 4 | Загрузка ранее сохранённых данных | 3  foursix  1 | Ваша матрица:  1 2 3 4 5 6  6 5 4 3 2 1  2 2 2 2 2 2  3 3 3 3 3 3 | Произошла загрузка ранее сохранённой матрицы, не изменённой ввиду того, что изменения не были сохранены |

6 Результаты тестирования

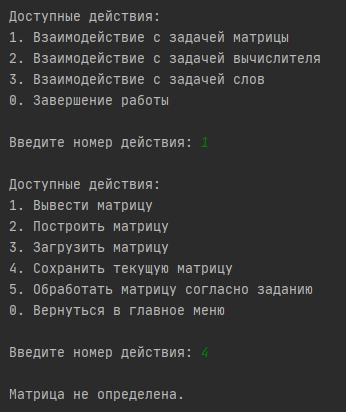


Рисунок 3 – Результат теста 1

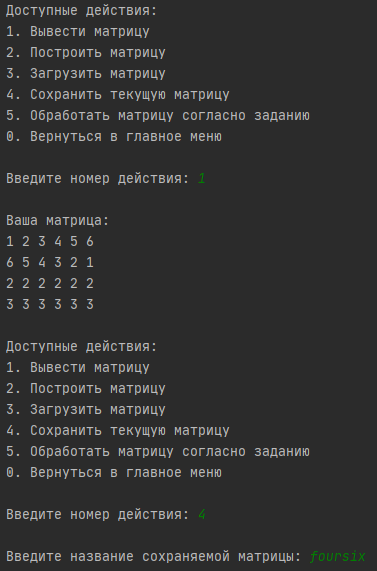


Рисунок 4 – Результат теста 2

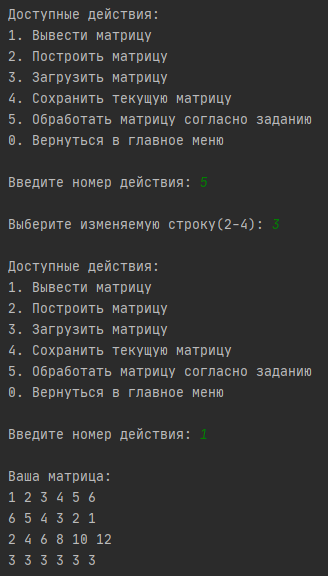


Рисунок 5 – Результат теста 3

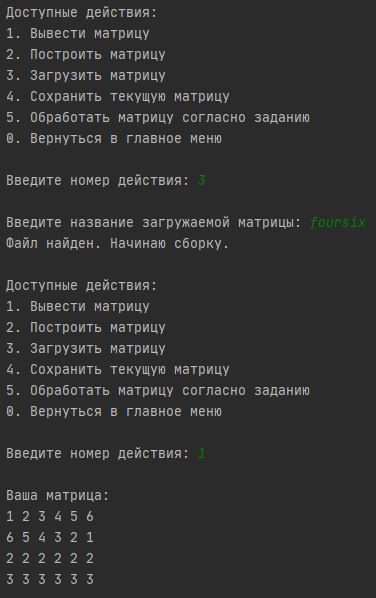


Рисунок 7 – Результат теста 4

7 Исходный код

Класс Main

package Lab\_3;  
public class Main  
{  
 public static void main(String[] args)  
 {  
 Menu.*main*();  
 }  
}

Класс Menu

package Lab\_3;  
import java.util.Scanner;  
public class Menu  
{  
 private static Scanner *input* = new Scanner(System.*in*);  
 public static void main()  
 {  
 System.*out*.print("\n");  
 System.*out*.print("Доступные действия:\n");  
 System.*out*.print("1. Взаимодействие с задачей матрицы\n");  
 System.*out*.print("2. Взаимодействие с задачей вычислителя\n");  
 System.*out*.print("3. Взаимодействие с задачей слов\n");  
 System.*out*.print("0. Завершение работы\n");  
 System.*out*.print("\nВведите номер действия: ");  
 switch (*input*.nextLine())  
 {  
 case "1":  
 *matrixField*();  
 *main*();  
 break;  
 case "2":  
 *calculatorField*();  
 *main*();  
 break;  
 case "3":  
 *wordsField*();  
 *main*();  
 break;  
 case "0":  
 System.*out*.print("Завершаю работу.\n");  
 break;  
 default:  
 System.*out*.print("Действие не распознано. Возвращаюсь к главному меню.\n\n");  
 *main*();  
 break;  
 }  
 }  
 //region Matrix  
 private static Matrix *mObject*;  
 private static void matrixField()  
 {  
 System.*out*.print("\n");  
 System.*out*.print("Доступные действия:\n");  
 System.*out*.print("1. Вывести матрицу\n");  
 System.*out*.print("2. Построить матрицу\n");  
 System.*out*.print("3. Загрузить матрицу\n");  
 System.*out*.print("4. Сохранить текущую матрицу\n");  
 System.*out*.print("5. Обработать матрицу согласно заданию\n");  
 System.*out*.print("0. Вернуться в главное меню\n");  
 System.*out*.print("\nВведите номер действия: ");  
 switch (*input*.nextLine())  
 {  
 case "1":  
 System.*out*.print("\n");  
 if (*mObject* != null)  
 {  
 if (*mObject*.filled())  
 *mObject*.printMatrix();  
 else  
 System.*out*.print("Матрица не определена.\n");  
 }  
 else  
 System.*out*.print("Матрица не определена.\n");  
 *matrixField*();  
 break;  
 case "2":  
 System.*out*.print("\n");  
 *mObject* = new Matrix();  
 *matrixField*();  
 break;  
 case "3":  
 System.*out*.print("\n");  
 System.*out*.print("Введите название загружаемой матрицы: ");  
 *mObject* = new Matrix(ExceptionHelper.*readString*());  
 *matrixField*();  
 break;  
 case "4":  
 System.*out*.print("\n");  
 if (*mObject* != null)  
 {  
 if (*mObject*.filled())  
 {  
 System.*out*.print("Введите название сохраняемой матрицы: ");  
 *mObject*.saveAs(ExceptionHelper.*readString*());  
 }  
 else  
 System.*out*.print("Матрица не определена.\n");  
 }  
 else  
 System.*out*.print("Матрица не определена.\n");  
 *matrixField*();  
 break;  
 case "5":  
 System.*out*.print("\n");  
 if (*mObject* != null)  
 {  
 if (*mObject*.filled())  
 *mObject*.multiplyLine();  
 else  
 System.*out*.print("Матрица не определена.\n");  
 }  
 else  
 System.*out*.print("Матрица не определена.\n");  
 *matrixField*();  
 break;  
 case "0":  
 System.*out*.print("\n");  
 System.*out*.print("Возвращаюсь к главному меню.\n");  
 break;  
 default:  
 System.*out*.print("\n");  
 System.*out*.print("Действие не распознано. Повторите ввод.\n");  
 *matrixField*();  
 break;  
 }  
 }  
 //endregion  
 //region Calculator  
 private static Calculator *cObject*;  
 private static void calculatorField()  
 {  
 System.*out*.print("\n");  
 System.*out*.print("Доступные действия:\n");  
 System.*out*.print("1. Вывести решение\n");  
 System.*out*.print("2. Ввести значения\n");  
 System.*out*.print("3. Загрузить значения\n");  
 System.*out*.print("4. Сохранить текущие значения\n");  
 System.*out*.print("0. Вернуться в главное меню\n");  
 System.*out*.print("\nВведите номер действия: ");  
 switch (*input*.nextLine())  
 {  
 case "1":  
 System.*out*.print("\n");  
 if (*cObject* != null)  
 {  
 if (*cObject*.filled())  
 System.*out*.print(*cObject*.answer());  
 else  
 System.*out*.print("Вычислитель не определён.\n");  
 }  
 else  
 System.*out*.print("Вычислитель не определён.\n");  
 *calculatorField*();  
 break;  
 case "2":  
 System.*out*.print("\n");  
 *cObject* = new Calculator();  
 *calculatorField*();  
 break;  
 case "3":  
 System.*out*.print("\n");  
 System.*out*.print("Введите название загружаемого вычислителя: ");  
 *cObject* = new Calculator(ExceptionHelper.*readString*());  
 *calculatorField*();  
 break;  
 case "4":  
 System.*out*.print("\n");  
 if (*cObject* != null)  
 {  
 if (*cObject*.filled())  
 {  
 System.*out*.print("Введите название сохраняемого вычислителя: ");  
 *cObject*.saveAs(ExceptionHelper.*readString*());  
 }  
 else  
 System.*out*.print("Вычислитель не определён.\n");  
 }  
 else  
 System.*out*.print("Вычислитель не определён.\n");  
 *calculatorField*();  
 break;  
 case "0":  
 System.*out*.print("\n");  
 System.*out*.print("Возвращаюсь к главному меню.\n");  
 break;  
 default:  
 System.*out*.print("\n");  
 System.*out*.print("Действие не распознано. Повторите ввод.\n");  
 *calculatorField*();  
 break;  
 }  
 }  
 //endregion  
 //region Words  
 private static Words *wObject*;  
 private static void wordsField()  
 {  
 System.*out*.print("\n");  
 System.*out*.print("Доступные действия:\n");  
 System.*out*.print("1. Вывести изначальную строку\n");  
 System.*out*.print("2. Вывести обработанную строку\n");  
 System.*out*.print("3. Ввести строку\n");  
 System.*out*.print("4. Загрузить строку\n");  
 System.*out*.print("5. Сохранить текущую строку\n");  
 System.*out*.print("0. Вернуться в главное меню\n");  
 System.*out*.print("\nВведите номер действия: ");  
 switch (*input*.nextLine())  
 {  
 case "1":  
 System.*out*.print("\n");  
 if (*wObject* != null)  
 {  
 if (*wObject*.filled())  
 System.*out*.print("Изначальная строка: " + *wObject*.getLine() + "\n");  
 else  
 System.*out*.print("Строка не определена.\n");  
 }  
 else  
 System.*out*.print("Строка не определена.\n");  
 *wordsField*();  
 break;  
 case "2":  
 System.*out*.print("\n");  
 if (*wObject* != null)  
 {  
 if (*wObject*.filled())  
 *wObject*.print();  
 else  
 System.*out*.print("Строка не определена.\n");  
 }  
 else  
 System.*out*.print("Строка не определена.\n");  
 *wordsField*();  
 break;  
 case "3":  
 System.*out*.print("\n");  
 *wObject* = new Words();  
 *wordsField*();  
 break;  
 case "4":  
 System.*out*.print("\n");  
 System.*out*.print("Введите название загружаемой строки: ");  
 *wObject* = new Words(ExceptionHelper.*readString*());  
 *wordsField*();  
 break;  
 case "5":  
 System.*out*.print("\n");  
 if (*wObject* != null)  
 {  
 if (*wObject*.filled())  
 {  
 System.*out*.print("Введите название сохраняемой строки: ");  
 *wObject*.saveAs(ExceptionHelper.*readString*());  
 }  
 else  
 System.*out*.print("Строка не определена.\n");  
 }  
 else  
 System.*out*.print("Строка не определена.\n");  
 *wordsField*();  
 break;  
 case "0":  
 System.*out*.print("\n");  
 System.*out*.print("Возвращаюсь к главному меню.\n");  
 break;  
 default:  
 System.*out*.print("\n");  
 System.*out*.print("Действие не распознано. Повторите ввод.\n");  
 *wordsField*();  
 break;  
 }  
 }  
 //endregion  
}

Класс Matrix

package Lab\_3;  
import java.io.\*;  
public class Matrix implements Serializable  
{  
 private int[][] matrix;  
 private int height = 0;  
 private int width = 0;  
 public Matrix()  
 {  
 System.*out*.print("Введите высоту матрицы: ");  
 height = ExceptionHelper.*readInt*();  
 while (height < 2)  
 {  
 System.*out*.print("Пожалуйста, введите число более 1: ");  
 height = ExceptionHelper.*readInt*();  
 }  
 System.*out*.print("Введите ширину матрицы: ");  
 width = ExceptionHelper.*readInt*();  
 while (width < 2)  
 {  
 System.*out*.print("Пожалуйста, введите число более 1: ");  
 width = ExceptionHelper.*readInt*();  
 }  
 System.*out*.println("Далее строка за строкой введите матрицу размерностью " + height + "x" + width);  
 matrix = new int[height][width];  
 for (int i = 0; i < height; i++)  
 {  
 int[] temp = ExceptionHelper.*readIntArray*(width);  
 for (int j = 0; j < width; j++)  
 matrix[i][j] = temp[j];  
 }  
 }  
 public Matrix(String name)  
 {  
 try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name + ".mx")))  
 {  
 Matrix temp = (Matrix)ois.readObject();  
 System.*out*.print("Файл найден. Начинаю сборку.\n");  
 matrix = temp.matrix;  
 height = temp.height;  
 width = temp.width;  
 }  
 catch (Exception exception)  
 {  
 System.*out*.print("Объект с указанным именем не найден.\n");  
 }  
 }  
 public void multiplyLine()  
 {  
 System.*out*.print("Выберите изменяемую строку(2-" + height + "): ");  
 int line = ExceptionHelper.*readInt*() - 1;  
 while (line < 1 || line + 1 > height)  
 {  
 System.*out*.print("Пожалуйста, введите число от 2 до " + height + ": ");  
 line = ExceptionHelper.*readInt*() - 1;  
 }  
 for (int i = 0; i < width; i++)  
 matrix[line][i] \*= matrix[0][i];  
 }  
 public void printMatrix()  
 {  
 System.*out*.print("Ваша матрица:\n");  
 for (int i = 0; i < height; i++)  
 {  
 for (int j = 0; j < width; j++)  
 System.*out*.print(matrix[i][j] + " ");  
 System.*out*.print('\n');  
 }  
 }  
 public void saveAs(String name)  
 {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(name + ".mx")))  
 {  
 oos.writeObject(this);  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println(exception.getMessage());  
 }  
 }  
 public boolean filled()  
 {  
 return matrix != null;  
 }  
}

Класс Calculator

package Lab\_3;  
import java.io.\*;  
public class Calculator implements Serializable  
{  
 private double a;  
 private double b;  
 private String operation;  
 public Calculator()  
 {  
 System.*out*.print("Через пробел введите два рабочих числа: ");  
 double[] temp = ExceptionHelper.*readDoubleArray*(2);  
 a = temp[0];  
 b = temp[1];  
 System.*out*.print("Введите знак операции: ");  
 operation = ExceptionHelper.*readString*();  
 }  
 public Calculator(String name)  
 {  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name + ".clc")))  
 {  
 Calculator temp = (Calculator)ois.readObject();  
 System.*out*.print("Файл найден. Начинаю сборку.\n");  
 a = temp.a;  
 b = temp.b;  
 operation = temp.operation;  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println("Объект с указанным именем не найден.");  
 }  
 }  
 public void saveAs(String name)  
 {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(name + ".clc")))  
 {  
 oos.writeObject(this);  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println(exception.getMessage());  
 }  
 }  
 public String answer()  
 {  
 switch (operation)  
 {  
 case "+":  
 return a + " " + operation + " " + b + " = " + (a + b) + '\n';  
 case "-":  
 return a + " " + operation + " " + b + " = " + (a - b) + '\n';  
 case "\*":  
 return a + " " + operation + " " + b + " = " + (a \* b) + '\n';  
 case "/":  
 return a + " " + operation + " " + b + " = " + (a / b) + '\n';  
 default:  
 return "Операция недействительна\n";  
 }  
 }  
 public boolean filled()  
 {  
 return operation != null;  
 }  
}

Класс Words

package Lab\_3;  
import java.io.\*;  
public class Words implements Serializable  
{  
 private String line;  
 private String[] array;  
 public Words()  
 {  
 System.*out*.print("Введите обрабатываемые слова через пробел или запятую: ");  
 line = ExceptionHelper.*readString*();  
 doTask();  
 }  
 public Words(String name)  
 {  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name + ".ws")))  
 {  
 Words temp = (Words)ois.readObject();  
 System.*out*.print("Файл найден. Начинаю сборку.\n");  
 line = temp.line;  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println("Объект с указанным именем не найден. Создаю базовый экземпляр.");  
 line = "o,iho n w,e,voj mqi v,bps gb,n";  
 }  
 finally  
 {  
 doTask();  
 }  
 }  
 public void saveAs(String name)  
 {  
 try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(name + ".ws")))  
 {  
 oos.writeObject(this);  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println(exception.getMessage());  
 }  
 }  
 public String getLine()  
 {  
 return line;  
 }  
 private void doTask()  
 {  
 int counter = 0;  
 String[] buffer = line.split("[, ]");  
 for (int i = 0; i < buffer.length; i++)  
 if (buffer[i].length() == 1)  
 {  
 buffer[counter] = buffer[i];  
 counter++;  
 }  
 array = new String[counter];  
 for (int i = 0; i < counter; i++)  
 array[i] = buffer[i];  
 }  
 public void print()  
 {  
 System.*out*.print("Все односимвольные слова: ");  
 for (int i = 0; i < array.length; i++)  
 System.*out*.print(array[i] + ' ');  
 System.*out*.println();  
 }  
 public boolean filled()  
 {  
 return line != null;  
 }  
}

Класс ExceptionHelper

package Lab\_3;  
import java.util.Scanner;  
public class ExceptionHelper  
{  
 private static Scanner *input* = new Scanner(System.*in*);  
 public static String readString()  
 {  
 String temp = *input*.nextLine();  
 if (temp != "")  
 return temp;  
 else  
 {  
 System.*out*.print("Пустой ввод. Повторите ввод строки.\n");  
 return *readString*();  
 }  
 }  
 public static int readInt()  
 {  
 try  
 {  
 int temp = Integer.*parseInt*(*readString*());  
 return temp;  
 }  
 catch (Exception e)  
 {  
 System.*out*.print("Ошибка при обработке числа. Повторите ввод целого числа.\n");  
 return *readInt*();  
 }  
 }  
 public static double readDouble()  
 {  
 try  
 {  
 double temp = Double.*parseDouble*(*readString*());  
 return temp;  
 }  
 catch (Exception e)  
 {  
 System.*out*.print("Ошибка при обработке числа. Повторите ввод числа.\n");  
 return *readDouble*();  
 }  
 }  
 public static int[] readIntArray(int size)  
 {  
 int counter = 0;  
 String[] buffer = *readString*().split(" ");  
 for (int i = 0; i < buffer.length; i++)  
 if (buffer[i].length() >= 1)  
 {  
 buffer[counter] = buffer[i];  
 counter++;  
 }  
 if (counter == size)  
 {  
 try  
 {  
 int[] temp = new int[counter];  
 for (int i = 0; i < counter; i++)  
 temp[i] = Integer.*parseInt*(buffer[i]);  
 return temp;  
 }  
 catch (Exception e)  
 {  
 System.*out*.print("Ошибка при обработке значений. Повторите ввод с [" + size + "] целочисленными аргументами.\n");  
 return *readIntArray*(size);  
 }  
 }  
 else  
 {  
 System.*out*.print("Неверное количество значений. Повторите ввод с [" + size + "] целочисленными аргументами.\n");  
 return *readIntArray*(size);  
 }  
 }  
 public static double[] readDoubleArray(int size)  
 {  
 int counter = 0;  
 String[] buffer = *readString*().split(" ");  
 for (int i = 0; i < buffer.length; i++)  
 if (buffer[i].length() >= 1)  
 {  
 buffer[counter] = buffer[i];  
 counter++;  
 }  
 if (counter == size)  
 {  
 try  
 {  
 double[] temp = new double[counter];  
 for (int i = 0; i < counter; i++)  
 temp[i] = Double.*parseDouble*(buffer[i]);  
 return temp;  
 }  
 catch (Exception e)  
 {  
 System.*out*.print("Ошибка при обработке значений. Повторите ввод с [" + size + "] численными аргументами.\n");  
 return *readDoubleArray*(size);  
 }  
 }  
 else  
 {  
 System.*out*.print("Неверное количество значений. Повторите ввод с [" + size + "] численными аргументами.\n");  
 return *readDoubleArray*(size);  
 }  
 }  
}

Список использованных источников