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 №2 по дисциплине «Объектно-ориентированное программирование» |
| 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](#_Toc53296258)

[2 Описание структуры классов 4](#_Toc53296259)

[3 Описание методов классов и спецификация локальных переменных 5](#_Toc53296260)

[4 Таблица тестов 8](#_Toc53296261)

[5 Результаты тестирования 9](#_Toc53296262)

[6 Исходный код 10](#_Toc53296263)

[Список использованных источников 15](#_Toc53296264)

*В тексте пустых строк не должно быть.*

*Никакие объекты не должны выходить за поля – ни текст, ни таблицы, ни рисунки.*

*Код лучше писать черным на белом фоне.*

*Литература должна быть оформлена по стандарту, даже ссылки на сайты.*

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 не обходимых данных и методы, обеспечивающие достаточную для решения задачи функциональность класса. Создание объекта класса решающего задание и вызов его методов должны осуществляться из главного класса. Ввод и вывод данных осуществляется как в предыдущей лабораторной работе, для передачи исходных данных экземпляру класса, решающему задачу, должны быть разработаны соответствующие методы.

2 Описание структуры классов

Проект разделён на 4 класса. Класс Main является точкой входа в программу и модулем управления объектами классов. Классы Matrix, Calculator и Words решают 1, 2 и 3 задачи соответственно.

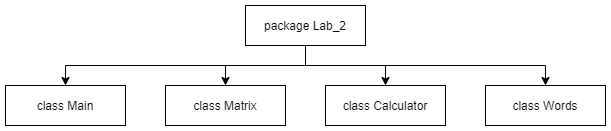


Рисунок 1 – Структура классов

*Обоснуйте выбор полей для классов. Какую роль играют поля в отношении объекта классов..? Как известно совокупность полей должна обладать свойством полноты, непротиворечивости и независимости. Зависимость полей может привести к противоречивости или несоответствию значений зависимых полей.*

3 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 ниже описанные методы. | - | public static | - |
| matrixTask | Создаёт объект класса Matrix и решает с его помощью задачу. | - | private static | - |
| calculateTask | Создаёт объект класса Calculator и решает с его помощью задачу. | - | private static | - |
| wordsTask | Создаёт объект класса Words и решает с его помощью задачу. | - | private static | - |

Таблица 2 – Описание полей класса Main

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Модификаторы |
| mObject | Объект класса Matrix | Matrix | private static |
| cObject | Объект класса Calculator | Calculator | private static |
| wObject | Объект класса Words | Words | private static |

Таблица 3 – Описание методов класса Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| Matrix | Конструктор по умолчанию. Устанавливает поля объекта значениями по умолчанию. | Matrix | public | - |
| Matrix | Конструктор по аргументам. Строит объект из переданной матрицы чисел. | Matrix | public | int[][] array |
| 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 |
| getMatrix | Функция, возвращающая матрицу значений объекта. | int[][] | public | - |

Продолжение таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| buildMatrix | Статическая функция построения объекта класса путём использования пользовательского интерфейса. | Matrix | public static | - |

Таблица 4 – Описание полей класса Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Модификаторы |
| matrix | Двумерный массив значений матрицы | int[][] | private |

Таблица 5 – Описание методов класса Calculator

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| Calculator | Конструктор по умолчанию. Устанавливает поля объекта значениями по умолчанию. | Calculator | public | - |
| Calculator | Конструктор по аргументам. Строит объект из переданных значений. | Calculator | public | Double operandA, double operandB, String operator |
| Calculator | Неполный конструктор по аргументам. Строит объект из переданных значений с частично пустыми полями. | Calculator | public | Double operandA, double operandB |
| Calculator | Конструктор копирования. Восстанавливает объект из файла по переданному имени. | Calculator | public | String name |
| getOperandA | Возвращает значение операнда A | double | public | - |
| getOperandB | Возвращает значение операнда B | double | public | - |
| getOperation | Возвращает значение оператора | String | public | - |
| setOperands | Устанавливает значение операндов | - | public | Double operandA, double operandB |
| setOperation | Устанавливает значение оператора | - | public | String operator |
| saveAs | Сохраняет объект в файл с переданным именем. | - | public | String name |

Продолжение таблицы 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| buildCalculator | Статическая функция построения объекта класса путём использования пользовательского интерфейса. | Calculator | public static | - |
| answer | Функция создания строки, включающей произведённые вычисления. | String | public | - |

Таблица 6 – Описание полей класса Calculator

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

Таблица 7 – Описание методов класса Words

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| Words | Конструктор по умолчанию. Создаёт объект класса путём использования пользовательского интерфейса. | Words | public | - |
| Words | Конструктор по аргументам. Строит объект по переданной строке. | Words | public | String line |
| Words | Конструктор копирования. Восстанавливает объект из файла по переданному имени. | Words | public | String name, boolean check |
| saveAs | Сохраняет объект в файл с переданным именем. | - | public | String name |
| getLine | Возвращает необработанную строку | String | public | - |
| getArray | Возвращает обработанную строку в виде массива слов | String[] | public | - |
| doTask | Обрабатывает строку в массив слов | - | public | - |
| print | Выводит обработанную строку | - | public | - |

Таблица 8 – Описание полей класса Words

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

4 Таблица тестов

Таблица 7 - Тесты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Задание | Вход | Выход | Примечания |
| 1 | Создание объекта класса Matrix через конструктор с аргументами и решение им задачи, сохранение объекта | 2 2 2 2 2  1 2 3 4 5  1 1 1 1 1  1 1 1 1 1  5 5 5 6 5  2  “Perfectly\_Named\_Matrix” | 2 2 2 2 2  2 4 6 8 10  1 1 1 1 1  1 1 1 1 1  5 5 5 6 5 | Задача решена, объект сохранён методом сериализации |
| 2 | Создание объекта класса Calculator через конструктор с аргументами и решение им задачи, сохранение объекта | 7 9  \*  “Perfectly\_Named\_Calculator” | “7.0 \* 9.0 = 63.0” | Задача решена, объект сохранёy методом сериализации |
| 3 | Создание объекта класса Words через конструктор с аргументами и решение им задачи, сохранение объекта | “n iuow bneb uy,gba,v,hiuo,ae nh,r gh d,k”  “Just\_Words” | n v r d k | Задача решена, объект сохранён методом сериализации |
| 4 | Создание новых объектов по данным, сохранённым ранее и вывод значений их полей | “Perfectly\_Named\_Matrix”  “Perfectly\_Named\_Calculator”  “Just\_Words” | 2 2 2 2 2  2 4 6 8 10  1 1 1 1 1  1 1 1 1 1  5 5 5 6 5  “7.0 \* 9.0 = 63.0”  “n iuow bneb uy,gba,v,hiuo,ae nh,r gh d,k” | Создание объектов произошло без ошибок, данные аналогичны сохраняемым |

5 Результаты тестирования

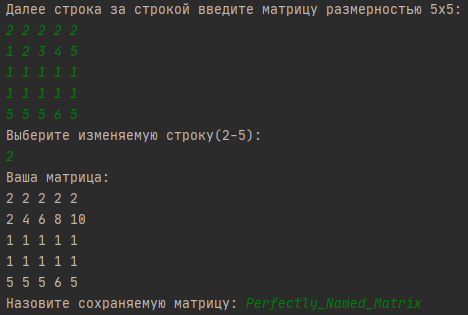


Рисунок 2 – Результат теста 1.1

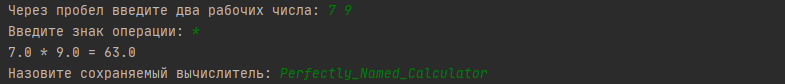


Рисунок 3 – Результат теста 1.2



Рисунок 4 – Результат теста 1.3

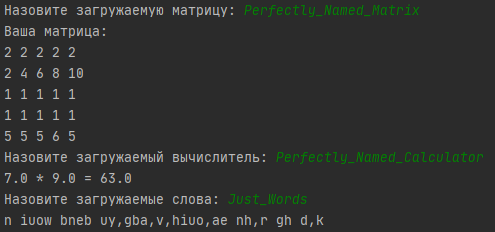


Рисунок 5 – Результат теста 1.4

6 Исходный код

1. Класс Main

package Lab\_2;  
import java.util.Scanner;  
public class Main  
{  
 private static Matrix *mObject*;  
 private static Calculator *cObject*;  
 private static Words *wObject*;  
 private static Scanner *input* = new Scanner(System.*in*);  
 public static void main(String[] args)  
 {  
 *matrixTask*();  
 System.*out*.print("Назовите сохраняемую матрицу: ");  
 *mObject*.saveAs(*input*.nextLine());  
 *calculateTask*();  
 System.*out*.print("Назовите сохраняемый вычислитель: ");  
 *cObject*.saveAs(*input*.nextLine());  
 *wordsTask*();  
 System.*out*.print("Назовите сохраняемые слова: ");  
 *wObject*.saveAs(*input*.nextLine());  
 System.*out*.print("Назовите загружаемую матрицу: ");  
 new Matrix(*input*.nextLine()).printMatrix();  
 System.*out*.print("Назовите загружаемый вычислитель: ");  
 System.*out*.print(new Calculator(*input*.nextLine()).answer());  
 System.*out*.print("Назовите загружаемые слова: ");  
 System.*out*.print(new Words(*input*.nextLine(), true).getLine());  
 }  
 private static void matrixTask()  
 {  
 *mObject* = Matrix.*buildMatrix*();  
 *mObject*.multiplyLine();  
 *mObject*.printMatrix();  
 }  
 private static void calculateTask()  
 {  
 *cObject* = Calculator.*buildCalculator*();  
 System.*out*.print(*cObject*.answer());  
 }  
 private static void wordsTask()  
 {  
 *wObject* = new Words();  
 *wObject*.doTask();  
 *wObject*.print();  
 }  
}

1. Класс Matrix

package Lab\_2;  
import java.io.\*;  
import java.util.Scanner;  
public class Matrix implements Serializable  
{  
 private static Scanner *input* = new Scanner(System.*in*);  
 private int[][] matrix;  
 public Matrix()  
 {  
 matrix = new int[5][5];  
 for (int i = 0; i < 5; i++)  
 for (int j = 0; j < 5; j++)  
 matrix[i][j] = j;  
 }  
 public Matrix(int[][] array)  
 {  
 matrix = array;  
 }  
 public Matrix(String name)  
 {  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name + ".mx")))  
 {  
 Matrix temp = (Matrix)ois.readObject();  
 matrix = temp.matrix;  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println("Объект с указанным именем не найден. Создаю базовый экземпляр.");  
 matrix = new int[5][5];  
 for (int i = 0; i < 5; i++)  
 for (int j = 0; j < 5; j++)  
 matrix[i][j] = j;  
 }  
 }  
 public void multiplyLine()  
 {  
 System.*out*.print("Выберите изменяемую строку(2-5):\n");  
 int line = *input*.nextInt() - 1;  
 for (int i = 0; i < 5; i++)  
 matrix[line][i] \*= matrix[0][i];  
 }  
 public void printMatrix()  
 {  
 System.*out*.print("Ваша матрица:\n");  
 for (int i = 0; i < 5; i++)  
 {  
 for (int j = 0; j < 5; 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 int[][] getMatrix()  
 {  
 return matrix;  
 }  
 public static Matrix buildMatrix()  
 {  
 int[][] mx = new int[5][5];  
 System.*out*.print("Далее строка за строкой введите матрицу размерностью 5x5:\n");  
 for (int i = 0; i < 5; i++)  
 for (int j = 0; j < 5; j++)  
 mx[i][j] = *input*.nextInt();  
 return new Matrix(mx);  
 }  
}

1. Класс Calculator

package Lab\_2;  
import java.io.\*;  
import java.util.Scanner;  
public class Calculator implements Serializable  
{  
 private double a;  
 private double b;  
 private String operation;  
 private static Scanner *input* = new Scanner(System.*in*);  
 public Calculator()  
 {  
 a = 5;  
 b = 6;  
 operation = "\*";  
 }  
 public Calculator(double operandA, double operandB, String operator)  
 {  
 a = operandA;  
 b = operandB;  
 operation = operator;  
 }  
 public Calculator(double operandA, double operandB)  
 {  
 a = operandA;  
 b = operandB;  
 }  
 public Calculator(String name)  
 {  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name + ".clc")))  
 {  
 Calculator temp = (Calculator)ois.readObject();  
 a = temp.getOperandA();  
 b = temp.getOperandB();  
 operation = temp.getOperation();  
 }  
 catch(Exception exception)  
 {  
 System.*out*.println("Объект с указанным именем не найден. Создаю базовый экземпляр.");  
 a = 5;  
 b = 6;  
 operation = "+";  
 }  
 }  
 public double getOperandA()  
 {  
 return a;  
 }  
 public double getOperandB()  
 {  
 return b;  
 }  
 public String getOperation()  
 {  
 return operation;  
 }  
 public void setOperands(double operandA, double operandB)  
 {  
 a = operandA;  
 b = operandB;  
 }  
 public void setOperation(String operator)  
 {  
 operation = operator;  
 }  
 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 static Calculator buildCalculator()  
 {  
 System.*out*.print("Через пробел введите два рабочих числа: ");  
 Calculator calculator = new Calculator(*input*.nextDouble(), Double.*parseDouble*(*input*.nextLine()));  
 System.*out*.print("Введите знак операции: ");  
 calculator.setOperation(*input*.nextLine());  
 return calculator;  
 }  
 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";  
 }  
 }  
}

1. Класс Calculator

package Lab\_2;  
import java.io.\*;  
import java.util.Scanner;  
public class Words implements Serializable  
{  
 private String line;  
 private String[] array;  
 private static Scanner *input* = new Scanner(System.*in*);  
 public Words()  
 {  
 System.*out*.print("Введите обрабатываемые слова через пробел или запятую: ");  
 line = *input*.nextLine();  
 }  
 public Words(String line)  
 {  
 this.line = line;  
 }  
 public Words(String name, boolean check)  
 {  
 try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream(name + ".ws")))  
 {  
 Words temp = (Words)ois.readObject();  
 line = temp.getLine();  
 }  
 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";  
 }  
 }  
 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;  
 }  
 public String[] getArray()  
 {  
 return array;  
 }  
 public 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();  
 }  
}

Список использованных источников

1. <https://metanit.com/java/tutorial/6.10.php>