Министерство науки и образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

ОТЧЕТ

Вариант 16

к лабораторной работе № 3 по дисциплине:

|  |
| --- |
| Объектно–ориентированное программирование |
| Работа с потоками данных, обработка исключений |
|  |

наименование темы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | АСУб-19-1 |  |  |  | Гамаюнова А.С |
|  | шифр группы |  | подпись |  | Фамилия ИО |
| Проверил: | доцент |  |  |  | Маланова Т.В. |
|  | должность |  | подпись |  | Фамилия ИО |

Иркутск 2020 г.

**Содержание**

[1 Постановка задачи 3](#_Toc51082884)

[3 Описание структуры класса 5](#_Toc51082885)

[4 Описание методов классов и спецификации локальных переменных 6](#_Toc51082886)

[5 Таблица тестов 9](#_Toc51082887)

[6 Результаты тестирования 10](#_Toc51082888)

[7 Исходный код 12](#_Toc51082889)

[Список использованных источников 18](#_Toc51082890)

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. 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 в экземпляр класса решающего задачу должны быть разработаны соответствующие методы.

Индивидуальное задание:

1. Сгенерируйте числовой ряд длиной не более 50, представляющий собой числа Фибоначчи (каждый последующий элемент представляют суммы двух предыдущих, первые два элемента равны 1). Выведите на печать первые 10, отмечая четные числа каким-нибудь символом.
2. Подсчитайте количество всех знаков препинания в заданном тексте.

Удалите из строки, содержащей слова, разделенные пробелами, запятыми и точками, у которых первая буква не совпадает с последней.

2 Описание структуры пользовательского меню

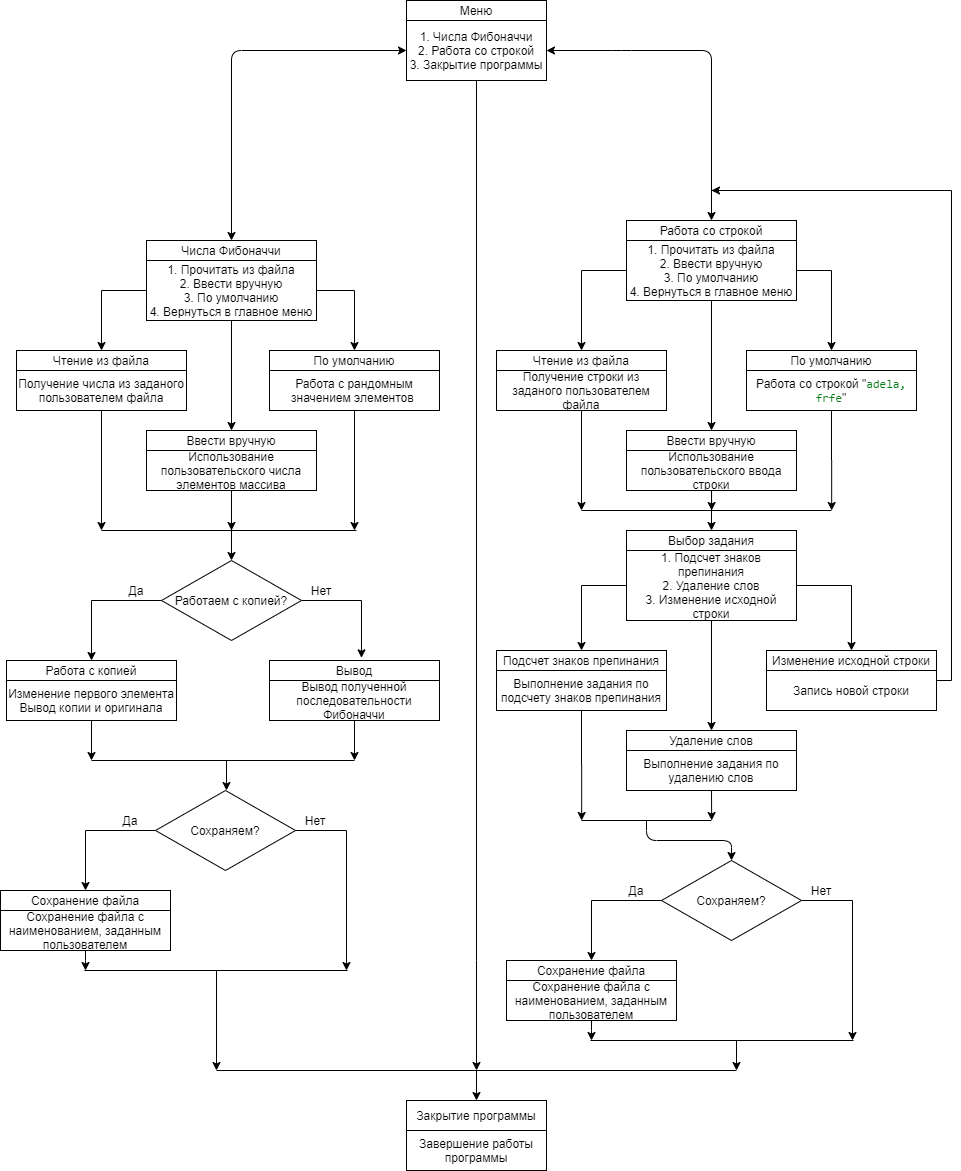


Рисунок 1 - Структура пользовательского меню

3 Описание структуры класса

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица 1 – Проектирование классов | | | |
| **№** | **Название** | **Назначение** | **Поля (атрибуты)** |
| 1 | public class L316 | Главный класс | - |
| 2 | public class Array | Класс для работы с массивом | private int amount; private int[] arr; |
| 3 | public class WorkString | Класс для работы со строкой | private String text; |
| 4 | public class Interface | Класс интерфейса для обработки пользовательского ввода | private static Array massiv; private static WorkString stroka; private static Exception exception; |
| 5 | Public class WorkFile | Класс для работы с файлами |  |

4 Описание методов классов и спецификации локальных переменных

Таблица 2 – Описание методов классов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Наименование метода | Описание метода | Тип данных | Тип доступа |
| Class WorkString | | | | |
| 1 | WorkString() | Конструктор | - | public |
| 2 | createNew\_str() | Создание строки без слов начинающихся и заканчивающихся на разные буквы | String | private |
| 3 | getNew\_Str | Вывод строки без слов начинающихся и заканчивающихся на разные буквы | void | public |
| 4 | countTotal() | Подсчет знаков препинания | int | private |
| 5 | getTotal() | Вывод знаков препинания | void | public |
| 6 | setStr | Добавление элемента для копирования | void | public |
| 7 | getStr | Вывод копирования | String | public |
| Class Array | | | | |
| 8 | Array() | Конструктор | - | public |
| 9 | createArr() | Заполнение массива числами Фибоначчи | int [] | private |
| 10 | getArr | Вывод массива | void | public |
| 11 | setArr | Изменение элемента для копирования | void | public |
| Class Interface | | | | |
| 12 | Interface() | Конструктор | - | public |
| 13 | choiceInput() | Главное меню | void | private |
| 14 | menuFib() | Меню для работы с последовательностью Фибоначчи | void | private |
| 15 | menuStr() | Меню для работы со строкой | void | private |
| Class WorkFile | | | | |
| 16 | loadMFile() | Чтение файла для работы с последовательностью Фибоначчи | int | public |
| 17 | loadSFile() | Чтение файла для работы со строкой | String | public |
| 18 | writeFile() | Запись строки в файл | void | public |
| 19 | writeByte() | Запись числа в файл | void | public |

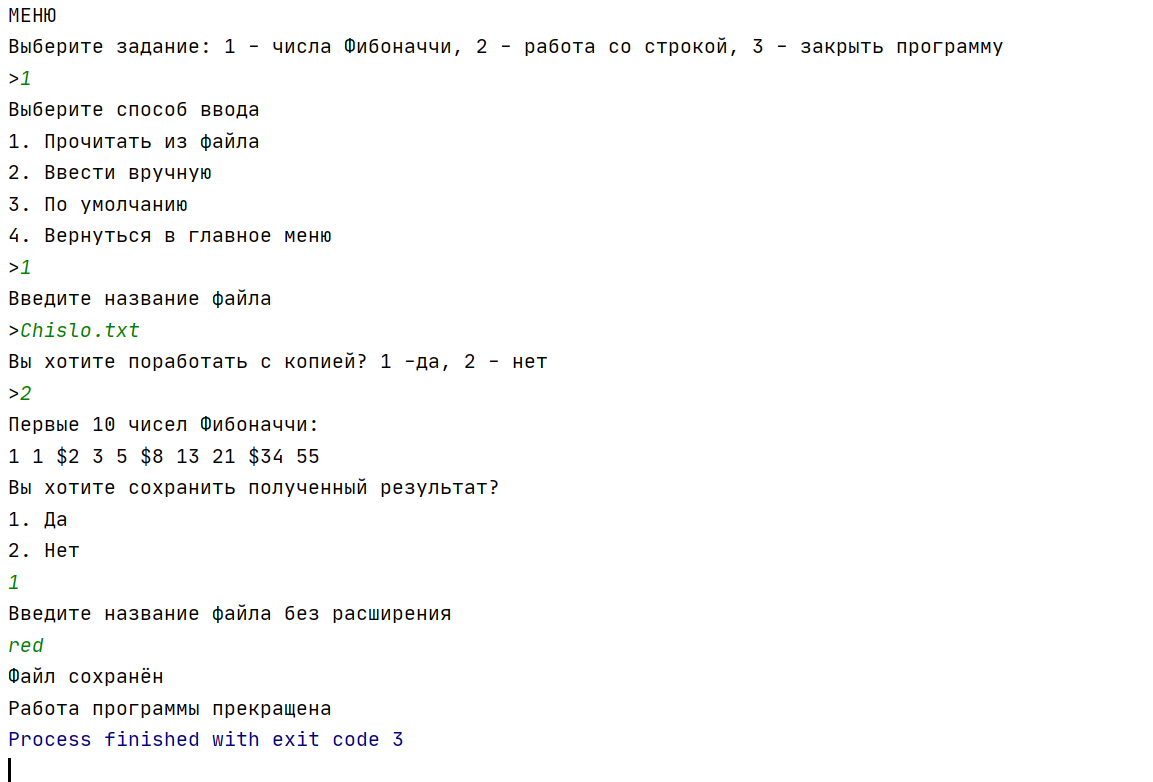
Таблица 3 – Описание локальных переменных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя переменной | Описание переменных | Тип данных | Тип доступа | ОДЗ |
| Class WorkString | | | | | |
| 1 | text | Исходная строка | String | private | [‘A’..’z’];  [‘А’..‘я’] |
| 2 | Before | Количество символов в исходной строке | int | private | [1..10000] |
| 3 | After | Количество символов в отформатированной строке | int | private | [1..10000] |
| 4 | word\_check | Массив слов | String | private | [‘A’..’z’];  [‘А’..‘я’] |
| Class Array | | | | | |
| 8 | Amount | Длина генерируемого ряда | String | private | [10;50] |
| 9 | arr | Массив с последовательностью Фибоначчи | int [] | private | [1;55] |
| 10 | i | Индекс элемента в массиве | int | private | [0;9] |
| 11 | element | Элемент массива | int | private | [1;55] |
| Class Interface | | | | | |
| 12 | choice | Выбор из главного меню | int | private | [1;3] |
| 13 | ch | Выбор из дополнительного меню | int | private | [1;4] |
| 14 | save | Строка для сохранения | String | private | [‘A’..’z’];  [‘А’..‘я’] |
| 15 | ch\_save | Число для сохранения | int | private | [1;10000] |
| Class WorkFile | | | | | |
| 16 | out | Число из файла | int | private | [1;10000] |
| 17 | output | Строка из файла | String | private | [‘A’..’z’];  [‘А’..‘я’] |
| 18 | fn | Имя файла с расширением | String | private | [‘A’..’z’];  [‘А’..‘я’, “.”] |

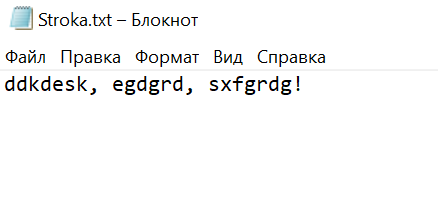
5 Таблица тестов

|  |  |
| --- | --- |
| № | Описание |
| 1 | Чтение из файла и запись в файл последовательности Фибоначчи |
| 2 | Чтение из файла строки и запись в файл количества пунктуационных знаков |
| 3 | Ошибка с вводом буквы, вместо числа |
| 4 | Ошибка работы с файлом, которого не существует |

6 Результаты тестирования



|  |
| --- |
| Рисунок 1 – Тест №1 |
|  |



|  |
| --- |
| Рисунок 2 – Данные для теста №2 |

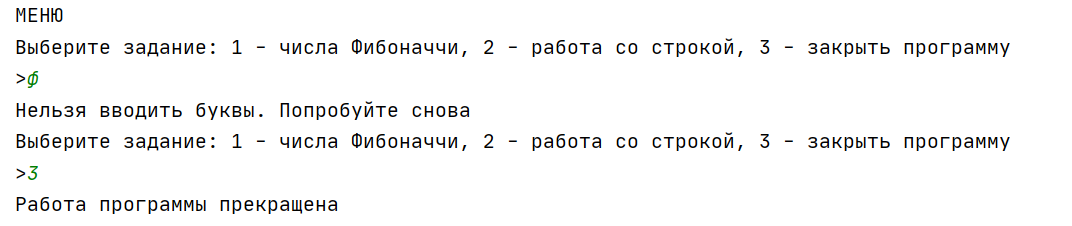


Рисунок 3 – Тест №4

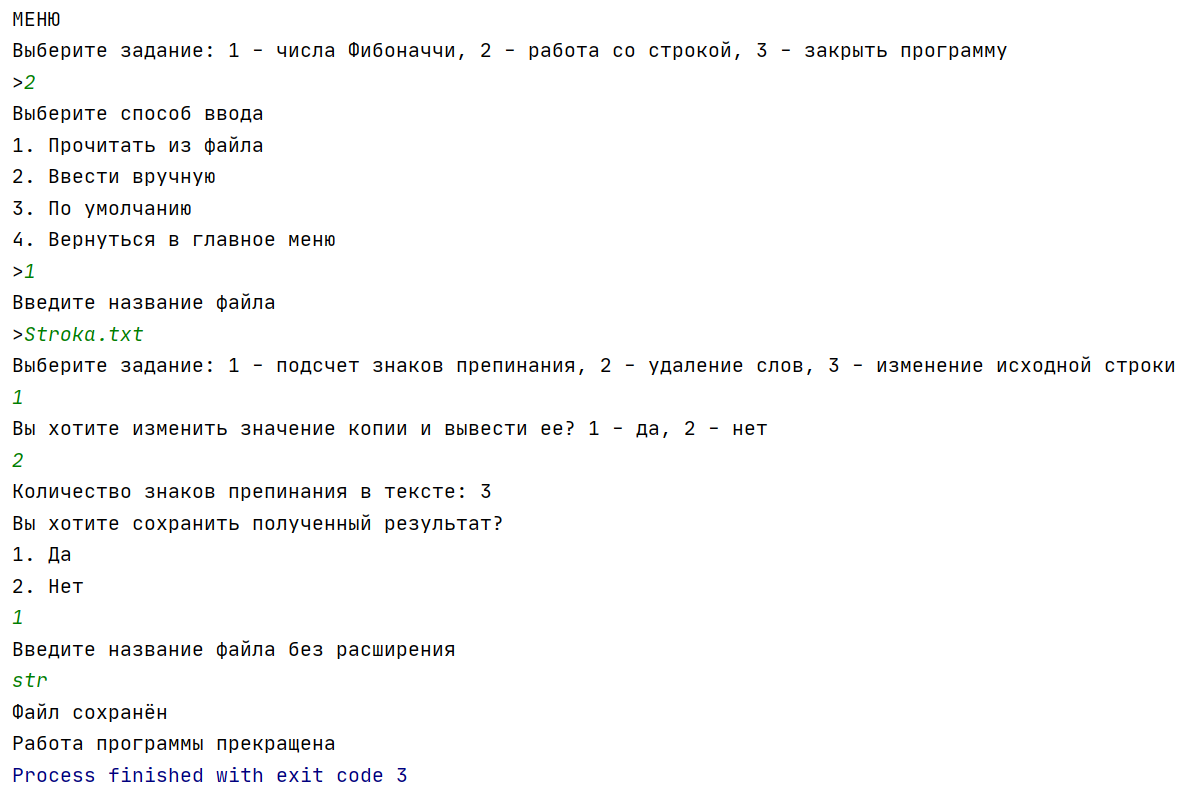
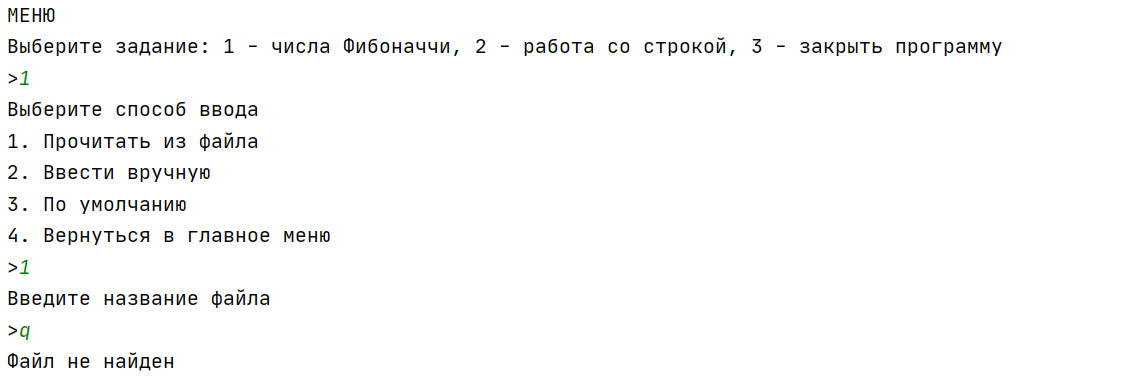


Рисунок 5 – Тест №5

Рисунок 4 – Тест №3

7 Исходный код

L316

package com.company;  
  
import java.sql.SQLOutput;  
import java.util.Arrays;  
import java.util.Scanner;  
  
public class L316 {  
 public static void main(String[] args) {  
 new Interface();  
 }  
}

Interface:  
package com.company;  
  
import java.util.InputMismatchException;  
import java.util.Scanner;  
import java.io.\*;  
  
public class Interface {  
 private static Scanner *in* = new Scanner(System.*in*);  
 private static Array *massiv*;  
 private static WorkString *stroka*;  
 private static Exception *exception*;  
  
 Interface() {  
 System.*out*.println("МЕНЮ");  
 choiceInput();  
 }  
  
 private void choiceInput() {  
 System.*out*.print("Выберите задание: 1 - числа Фибоначчи, 2 - работа со строкой, 3 - закрыть программу\n>");  
 try{  
 int choice = *in*.nextInt();  
 int ch;  
  
 switch (choice) {  
 case 1:  
 System.*out*.print("Выберите способ ввода\n1. Прочитать из файла\n2. Ввести вручную \n3. По умолчанию\n4. Вернуться в главное меню\n>");  
 ch = *in*.nextInt();  
 *in*.nextLine();  
 menuFib(ch);  
 break;  
 case 2:  
 System.*out*.print("Выберите способ ввода\n1. Прочитать из файла \n2. Ввести вручную \n3. По умолчанию\n4. Вернуться в главное меню\n>");  
 ch = *in*.nextInt();  
 *in*.nextLine();  
 menuStr(ch);  
 break;  
 case 3:  
 System.*out*.print("Работа программы прекращена");  
 System.*exit*(3);  
 break;  
 }  
 } catch (InputMismatchException e) {  
 System.*out*.println("Нельзя вводить буквы. Попробуйте снова");  
 *in*.next();  
 choiceInput();  
 }  
  
 }  
  
 private void menuFib ( int choice){  
 int ch;  
 switch (choice) {  
 case 1:  
 System.*out*.print("Введите название файла\n>");  
 String path = *in*.nextLine();  
 try {  
 int amount = WorkFile.*loadMFile*(path);  
 *massiv* = new Array(amount);  
 } catch (IOException e) {  
 System.*out*.println(e.getMessage());  
 System.*out*.println("Вы переведены в главное меню");  
 choiceInput();  
 }  
 break;  
 case 2:  
 System.*out*.print("Введите количество элементов массива\n>");  
 int amount = *in*.nextInt();  
 *massiv* = new Array(amount);  
 break;  
 case 3:  
 *massiv* = new Array();  
 break;  
 case 4:  
 choiceInput();  
 }  
 System.*out*.print("Вы хотите поработать с копией? 1 -да, 2 - нет\n>");  
 ch = *in*.nextInt();  
 String save;  
 switch (ch) {  
 case 1:  
 Array massiv1 = new Array(*massiv*);  
 System.*out*.println("Первые 10 чисел Фибоначчи: ");  
 *massiv*.getArr();  
 System.*out*.println();  
 System.*out*.println("Копия с переустановленным первым элементом");  
 massiv1.setArr(2);  
 massiv1.getArr();  
 System.*out*.println();  
 break;  
 case 2:  
 System.*out*.println("Первые 10 чисел Фибоначчи: ");  
 *massiv*.getArr();  
 System.*out*.println();  
 break;  
 }  
 save = *massiv*.getForSave();  
 int ch\_save;  
 System.*out*.println("Вы хотите сохранить полученный результат?\n1. Да \n2. Нет");  
 ch\_save = *in*.nextInt();  
 *in*.nextLine();  
 switch(ch\_save){  
 case 1:  
 System.*out*.println("Введите название файла без расширения");  
 String path = *in*.nextLine();  
 WorkFile.*writeByte*(save, path);  
 System.*out*.println("Файл сохранён");  
 break;  
 default:  
 System.*out*.print("Работа программы прекращена");  
 System.*exit*(3);  
 break;  
 }  
 System.*out*.print("Работа программы прекращена");  
 System.*exit*(3);  
 }  
  
 private void menuStr ( int choice){  
 int ch;  
 int ch\_save;  
 switch (choice) {  
 case 1:  
 System.*out*.print("Введите название файла\n>");  
 String path = *in*.nextLine();  
 try {  
 String text = WorkFile.*loadSFile*(path);  
 *stroka* = new WorkString(text);  
 } catch (IOException e) {  
 System.*out*.println(e.getMessage());  
 System.*out*.println("Вы переведены в главное меню");  
 choiceInput();  
 }  
 break;  
 case 2:  
 System.*out*.print("Введите любой текст:\n>");  
 String text = *in*.nextLine();  
 *stroka* = new WorkString(text);  
 break;  
 case 3:  
 *stroka* = new WorkString();  
 break;  
 case 4:  
 choiceInput();  
 case 5:  
 *in*.nextLine();  
 System.*out*.println("Введите новую строку");  
 String new\_str = *in*.nextLine();  
 *stroka*.setFStr(new\_str);  
 }  
 int k;  
 System.*out*.println("Выберите задание: 1 - подсчет знаков препинания, 2 - удаление слов, 3 - изменение исходной строки");  
 ch = *in*.nextInt();  
 k = ch;  
 String save;  
 switch (ch) {  
 case 1:  
 System.*out*.println("Вы хотите изменить значение копии и вывести ее? 1 - да, 2 - нет");  
 int cho = *in*.nextInt();  
 *in*.nextLine();  
 if (cho == 1) {  
 WorkString stroka1 = new WorkString(*stroka*);  
 System.*out*.println("Введите знак препинания");  
 String znak = *in*.nextLine();  
 stroka1.setStr(znak);  
 *//Работа с конструктором копирования* System.*out*.println("Копия");  
 stroka1.getTotal();  
 System.*out*.println("Оригинал");  
 *stroka*.getTotal();  
 } else {  
 *stroka*.getTotal();  
 }  
 break;  
 case 2:  
 *stroka*.getNew\_str();  
 break;  
 case 3:  
 menuStr(5);  
 break;  
 }  
 save = *stroka*.choiceOf(k);  
 System.*out*.println("Вы хотите сохранить полученный результат?\n1. Да \n2. Нет");  
 ch\_save = *in*.nextInt();  
 *in*.nextLine();  
 switch(ch\_save){  
 case 1:  
 System.*out*.println("Введите название файла без расширения");  
 String path = *in*.nextLine();  
 try {  
 WorkFile.*writeFile*(save, path);  
 System.*out*.println("Файл сохранён");  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 break;  
 default:  
 System.*out*.print("Работа программы прекращена");  
 System.*exit*(3);  
 break;  
 }  
 System.*out*.print("Работа программы прекращена");  
 System.*exit*(3);  
 }  
 }

Array

package com.company;  
  
import java.util.Random;  
  
public class Array {  
 private int amount;  
 private int[] arr;  
  
 public Array(){  
 this.amount = new Random().nextInt(41) + 10;  
 this.arr = createArr();  
 }  
  
 public Array(int amount){  
 this.amount = amount;  
 this.arr = createArr();  
 }  
  
 private int[] createArr(){  
 this.arr = new int[this.amount];  
  
 this.arr[0] = this.arr[1] = 1;  
  
 this.amount = this.amount - 2; *//так как 2 элемента задано* int i = 0;  
  
 while (i < this.amount) {  
 this.arr[i + 2] = this.arr[i + 1] + this.arr[i];  
  
 i++;  
 }  
 return this.arr;  
 }  
  
 public void getArr(){  
 int i = 0;  
  
 while (i < 10) {  
 int element = this.arr[i];  
  
 if (element % 2 == 0) {  
 System.*out*.print("$" + this.arr[i] + " ");  
 } else {  
 System.*out*.print(this.arr[i] + " ");  
 }  
  
 i++;  
 }  
 }  
 public Array(Array a){  
 this.amount = a.amount;  
 this.arr = a.arr;  
  
 }  
 public String getForSave(){  
 int i = 0;  
 String mass = "";  
  
 while (i < 10) {  
 int element = this.arr[i];  
  
 if (element % 2 == 0) {  
 mass = mass + "$" + Integer.*toString*(this.arr[i]) + " ";  
 } else {  
 mass = mass + Integer.*toString*(this.arr[i]) + " ";  
 }  
  
 i++;  
 }  
 return mass;  
 }  
 public void setArr(int chislo){  
 this.arr[0] = chislo;  
 }  
  
}

WorkString

package com.company;  
  
public class WorkString {  
 private String text;  
  
 WorkString() {  
 this.text = "adela, frfe";  
 }  
  
 WorkString(String text) {  
 this.text = text;  
 }  
  
 WorkString(WorkString s){  
 this(s.text);  
 }  
  
 private String createNew\_str() {  
 String[] word\_check = this.text.split("[,. ]+"); *//Разделяем строку на слова* for (int i = 0; i < word\_check.length; i++) {  
 *//Проверка первой и последней буквы* String wch = word\_check[i];  
 if (wch.toLowerCase().charAt(0) != wch.toLowerCase().charAt(wch.length() - 1))  
 this.text = this.text.replace(word\_check[i], "");  
 }  
 return this.text;  
 }  
  
 public void getNew\_str() {  
 System.*out*.println("После удаления слов,начинающихся и оканчивающихся на разные буквы, получилась такая строка: " + createNew\_str());  
 }  
  
 public String getForSaveNew(){  
 return createNew\_str();  
 }  
  
 private int countTotal() {  
 int before = this.text.length(); *//Подсчитываем длину текста* int after = this.text.replaceAll("[,.;!:?]", "").length(); *//Подсчитываем длину текста без знаков препинания* return before - after;  
 }  
  
 public void getTotal(){  
 if (countTotal() > 0)  
 System.*out*.println("Количество знаков препинания в тексте: " + countTotal());  
 else  
 System.*out*.println("Увы, в заданном тексте мне не удалось найти знаки препинания");  
 }  
 public int getForSaveTotal(){  
 return countTotal();  
 }  
  
 public String choiceOf(int ch){  
 String save = "";  
 String k = Integer.*toString*(getForSaveTotal());;  
 if (ch == 1)  
 save = k;  
 if (ch == 2)  
 save = getForSaveNew();  
 return save;  
 }  
  
 public void setStr(String znak){  
 if (znak.equals("!") || znak.equals(".") || znak.equals("?") || znak.equals(",") || znak.equals(";"))  
 this.text += znak;  
 }  
 public String getStr(){  
 return this.text;  
 }  
 public void setFStr(String new\_text){  
 this.text = new\_text;  
 }  
}

WorkFile

package com.company;  
  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.io.\*;  
import java.util.Scanner;  
  
public class WorkFile {  
  
 public WorkFile() {  
 }  
  
 public static int loadMFile (String name) throws IOException {  
 try{  
 File file = new File(name);  
 Scanner scanner = new Scanner(file);  
 int out = Integer.*parseInt*(scanner.nextLine());  
 scanner.close();  
 return out;  
 }  
 catch(FileNotFoundException e){  
 System.*out*.println("Файл не найден");  
 System.*exit*(3);  
 }  
 return 0;  
 }  
  
 public static String loadSFile (String name) throws IOException {  
 try {  
 File file = new File(name);  
 Scanner scanner = new Scanner(file);  
 String output = scanner.nextLine();  
 scanner.close();  
 return output;  
 }  
 catch(FileNotFoundException e){  
 System.*out*.println("Файл не найден");  
 System.*exit*(3);  
 }  
 return "";  
 }  
  
 public static void writeFile(String name, String filename) throws IOException {  
 String fn = filename + ".txt";  
 File file = new File(fn);  
 PrintWriter pw = new PrintWriter(file);  
 pw.println(name);  
 pw.close();  
 }  
 public static void writeByte(String name, String filename) {  
 String fn = filename + ".txt";  
 OutputStream os = null;  
 try {  
 os = new FileOutputStream(new File(fn));  
 os.write(name.getBytes(), 0, name.length());  
 } catch (IOException e) {  
 e.printStackTrace();  
 } finally {  
 try {  
 os.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

Список использованных источников

1. Аршинский В.Л. Объектно-ориентированное программирование. Метод. указания по выполнению лабораторных работ / В.Л. Аршинский. – Иркутск: Изд-во ИРНИТУ, 2015. – 23 c.