Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

|  |
| --- |
| Институт высоких технологий |
| наименование института  кафедра Автоматизированных систем |
| наименование кафедры |

ОТЧЁТ

к лабораторной работе № 3 по дисциплине

|  |
| --- |
| Объектно-ориентированное программирование |
| Работа с потоками данных и обработка исключений |
| наименование темы |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы: | ИСТб-16-1 |  |  |  | М.Е. Винзовский |
|  | Шифр |  | Подпись |  | И.О. Фамилия |
| Проверил: | доцент |  |  |  | В.Л. Аршинский |
|  | Должность |  | Подпись |  | И.О. Фамилия |
|  |  |  |  |  |  |

Иркутск 2018 г.

Оглавление

[1 Постановка задачи 3](#_Toc512623240)

[1.1 Цель работы 3](#_Toc512623241)

[1.2 Требования к выполнению лабораторной работы 3](#_Toc512623242)

[1.3 Содержание задания на выполнение лабораторной работы 3](#_Toc512623243)

[1.4 Вариант индивидуального задания 4](#_Toc512623244)

[2 Описание структуры пользовательского меню 5](#_Toc512623245)

[3 Проектирование классов 6](#_Toc512623246)

[4 Описание методов классов и спецификация локальных переменных 7](#_Toc512623247)

[5Таблица тестов 10](#_Toc512623248)

[6 Результаты тестирования 11](#_Toc512623249)

[7 Исходный код 12](#_Toc512623250)

[7.1 Исходный код класса Lab3 12](#_Toc512623251)

[7.2 Исходный код класса Menu 12](#_Toc512623252)

[7.3 Исходный код класса WFile 16](#_Toc512623253)

[7.4 Исходный код класса Chars 17](#_Toc512623254)

[Список использованных источников 23](#_Toc512623255)

# 1 Постановка задачи

## 1.1 Цель работы

1. Применение на практике объектно-ориентированного подхода для организации системы ввода/вывода.Применение элементов объектного подхода на практике: выполнение объектной декомпозиции, инкапсуляция механизмов реализации объектов.
2. Получение практического опыта описания и реализации классов, создания объектов, вызова методов, а также навыков работы с потоками данных и обработки исключений на базе платформы Java.

## 1.2 Требования к выполнению лабораторной работы

1. В ходе выполнения лабораторной работы студент должен на практике научиться проводить объектную декомпозицию задачи и на основе полученных результатов описывать и реализовывать соответствующие классы. Выполнение задания необходимо осуществлять с помощью интегрированной среды разработки приложений (IDE).
2. Выполнение лабораторной работы осуществляется с использованием интегрированной среды разработки приложений NetBeans IDE. Все поля и методы разработанных классов должны иметь соответствующие модификаторы доступа.
3. Студент должен написать программный код, решающий задачу в соответствии с индивидуальным заданием.
4. Помимо главного класса и класса, реализующего решение индивидуального задания, необходимо разработать класс для организации текстового меню, с помощью методов которого осуществляется управление выполнением приложения.
5. Поля и методы класса, реализующего меню программы, должны быть статическими (при их описании используется ключевое слово static).
6. Все поля и методы должны иметь соответствующие модификаторы доступа.
7. Программный код должен быть оформлен в соответствии с установленными правилами и содержать необходимые комментарии.
8. По результатам выполнения оформляется отчет в соответствии с установленными требованиями.

## 1.3 Содержание задания на выполнение лабораторной работы

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. Организация работы с файлами может быть организована методами класса, решающего индивидуальное задание, либо методами отдельного 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.

## 1.4 Вариант индивидуального задания

Индивидуальное задание выполняется на основе заданий к лабораторной работе №1. Т.е. конечная программа должна в итоге представлять собой четыре класса. Первый класс – класс, обрабатывающий файлы. Второй класс – класс, реализующий задачи консольного меню. Третий класс – главный класс, в котором и реализуется задача. Четвёртый класс занимающийся решением индивидуальных задач.

Программа должна выполнять следующие задачи:

* 1. Ввод строки, поиск символа встречающегося наибольшее количество раз.

Входные данные: строка.

Выходные данные: символ(-ы).

* 1. Ввод строки. Поиск самого длинного слова.

Входные данные: строка.

Выходные данные: слово(-а).

* 1. Удаление из строки каждого третьего слова.

Входные данные: строка.

Выходные данные: строка без удаленных слов.

# 2 Описание структуры пользовательского меню

Меню в данной работе представляет собой вывод пунктов меню, каждому из которых присвоен порядковый номер. Выбор пункта меню осуществляется вводом этого номера. При вводе неверного номера или вообще других данных, выводится сообщение об ошибке и предлагается ещё раз ввести номер пункта меню.

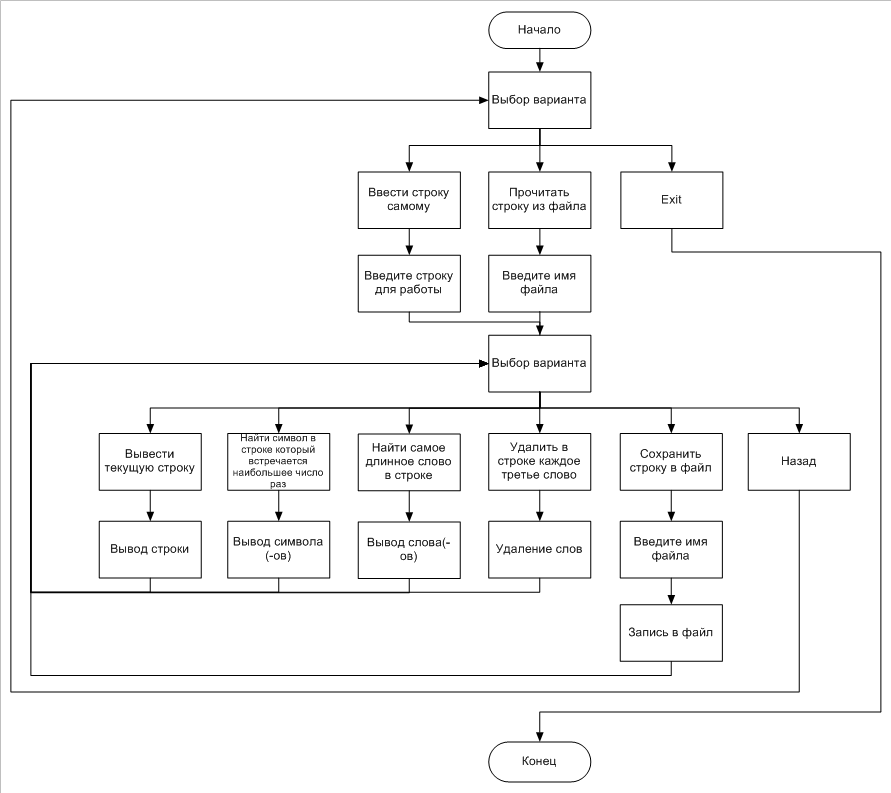


Рис. 1 – схема пользовательского меню программы

# 3 Проектирование классов

В процессе разработки программы были выделены 4 класса:

* класс Lab3 – основной класс
* класс Menu–класс, в котором реализуется работа с пользователем;
* класс Chars – класс, в котором выполняется работа со строкой;
* класс WFile – класс, в котором выполняется запись и чтения файла.

Таблица 1 - Поля класса Chars

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Имя** | **Тип** | **Назначение** | **ОДЗ** |
| ***Класс Chars*** | | | | |
| 1 | length | int | Длина строки | [0..2147483647] |
| 2 | chars | char[] | Массив символов | - |

# 4 Описание методов классов и спецификация локальных переменных

Таблица 2 – Методы классов и спецификация локальный переменных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Имя** | **Назначение** | **Тип** | | **ОДЗ** |
| **Класс lab3** | | | | | |
| ***public static void main(String[] args) – точка входа в программу*** | | | | | |
| **КлассMenu** | | | | | |
| ***public static void start() – Инициализация меню*** | | | | | |
| 1 | uslovie | Переменная для проверки данных | boolean | | {1;0} |
| 2 | stdin | Объект для ввода данных с клавиатуры | BufferedReader | | - |
| 3 | vibor | Переменная для выбора команды | String | | - |
| ***public static void choiceTask(String text)– Инициализация меню*** | | | | | |
| 4 | uslovie | Переменная для проверки данных | boolean | | {1;0} |
| 5 | stdin | Объект для ввода данных с клавиатуры | BufferedReader | | - |
| 6 | vibor | Переменная для выбора команды | String | | - |
| 7 | chars | Строка для обработки | Chars | | - |
| ***public static void writeFileChars(Chars chars) – Запись строки в файл*** | | | | | |
| 8 | uslovie | Переменная для проверки данных | boolean | | {1;0} |
| 9 | stdin | Объект для ввода данных с клавиатуры | BufferedReader | | - |
| 10 | fileName | Имя файла для записи | String | | - |
| ***public static void readFileChars() - Чтение строки из файла*** | | | | | |
| 11 | uslovie | Переменная для проверки данных | boolean | | {1;0} |
| 12 | stdin | Объект для ввода данных с клавиатуры | BufferedReader | | - |
| 13 | fileName | Имя файла для записи | String | | - |
| 14 | text | Считанная строка | String | | - |
| ***public static void readScanChars() - Ввод строки пользователем*** | | | | | |
| 15 | stdin | Объект для ввода данных с клавиатуры | BufferedReader | | - |
| 16 | text | Считанная строка | String | | - |
| **Класс WFile** | | | | | |
| ***public static boolean write(String fileName, String text) – Метод записи строки в файл*** | | | | | |
| 17 | usl | Проверка записи | boolean | | {0,1} |
| 18 | osw | Поток для записи символов в файл | OutputStreamWriter | | - |
| 19 | fos | Поток для записи в файл | FileOutputStream | | - |
| ***public static String read(String fileName) – Метод для чтения строки из файла*** | | | | | |
| 20 | text | Считанная строка | String | | - |
| 21 | fis | Поток для чтения символов из файла | FileInputStream | | - |
| 22 | isr | Поток для чтения из файла | InputStreamReader | | - |
| 23 | data | Числовое значение текущего считанного символа | int | | [0; 2147483647] |
| 24 | theChar | Текущий считанный символ | char | | - |
| **Класс Chars** | | | | | |
| ***Chars() – пустой конструктор*** | | | | | |
| ***Chars(char[] chars) – конструктор с вводом массива символов*** | | | | | |
| ***Chars(intlength) – конструктор с вводом кол-ва символов и заполнение его пробелами*** | | | | | |
| 25 | i | Счетчик | int | [0…2147483647] | |
| ***Chars(String string) – конструктор с вводом строки*** | | | | | |
| 26 | i | Счетчик | int | [0…2147483647] | |
| ***Chars(Chars clone) – конструктор копирования*** | | | | | |
| ***Chars(int n, char ch) – конструктор с вводом кол-ва символов и заполнение его определенном символом*** | | | | | |
| ***print() - метод для вывода строки*** | | | | | |
| 27 | i | Счетчик | int | [0…2147483647] | |
| 28 | text | Строка для вывода | String | - | |
| ***maxChar() –метод для нахождения кол-ва символов встречаемое наибольшее число раз*** | | | | | |
| 29 | i,j | Счетчики | int | [0…2147483647] | |
| 30 | count | Текущее кол-во одинаковых символов | int | [0…2147483647] | |
| 31 | max | Максимальное кол-во одинаковых символов | int | [0…2147483647] | |
| ***printCountChar(int count) –метод для вывода символов встречаемые определенное число раз*** | | | | | |
| 32 | i,j | Счетчики | int | [0…2147483647] | |
| 33 | c | Текущее кол-во одинаковых символов | int | [0…2147483647] | |
| 34 | count | Нужное число символов | int | [0…2147483647] | |
| 35 | text | Строка с символами | String | - | |
| ***maxWord() - метод для нахождения числа символов в самом длинном слове*** | | | | | |
| 36 | i,j | Счетчики | int | [0…2147483647] | |
| 37 | count | Текущая длина слова | int | [0…2147483647] | |
| 38 | max | Максимальная длина слова | int | [0…2147483647] | |
| ***printCountWord(int count) –метод для вывода слов определенной длины*** | | | | | |
| 39 | i,j | Счетчики | int | [0…2147483647] | |
| 40 | c | Текущая длина слова | int | [0…2147483647] | |
| 41 | count | Нужная длина слова | int | [0…2147483647] | |
| 42 | text | Строка со словами для вывода | String | - | |
| 43 | ch | Массив для записи текущего слова | Char[] | - | |
| ***deleteWords(int wordsDelete) - метод для удаления каждого wordsDelete слова*** | | | | | |
| 44 | wordsDelete | Удаление каждого wordsDelete слова | int | [0…2147483647] | |
| 45 | text | Строка без лишних слов | String | - | |
| 46 | ch | Новый массив словдля вывода | Chars | - | |
| 47 | three | Текущее слово | int | [0…2147483647] | |
| 48 | i,j | Счетчики | int | [0…2147483647] | |

# 5 Таблица тестов

Таблица 3 – Выделенные классы эквивалентности

|  |  |  |  |
| --- | --- | --- | --- |
| Область возможных ошибок | Входные условия | Правильные классы эквивалентности | Неправильные классы эквивалентности |
| Работа с меню | Введенная строка содержит только цифры | Содержит только цифры. (1) | Содержит другие символы.(2) |
| Число не в ОДЗ | Число от 0 до 2, целое(3) | Число меньше 0 (4)  Число больше 2 (5) |
| Ввод имени файла | Введенное имя файла корректно | Корректно (нет запрещенных символов) (6) | Некорректно (7) |

Таблица 4 – таблица тестов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Входные данные | Выходные данные |
| 1 | Введено целое правильное число  (классы эквивалентности 1,3) | 1 | Сообщение с запросом данных, соответствующих задаче. Или выход из программы (если введено 0) |
| 2 | Введено целое неподходящее число  (классы эквивалентности 1, 4 или 5) | 5 | "Нет такого варианта” |
| 3 | Введено не целое число  (классы эквивалентности 2,4) | 0.3 | " Нет такого варианта |
| 4 | Введено не число (классы эквивалентности 2) | as | " Нет такого варианта ” |
| 5 | Введено корректное имя файла  (класс эквивалентности 6) | name | Сообщение об успешности открытия или записи файла |
| 6 | Введено некорректное имя файла  (класс эквивалентности 7) | \*fa | Сообщение об ошибки открытия или записи файла |

# 6 Результаты тестирования

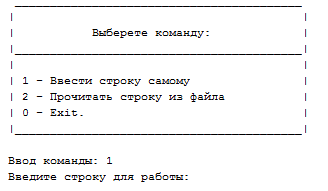


Рисунок 2 – результат теста 1

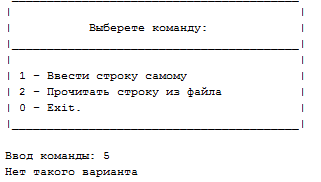


Рисунок 3 – результат теста 2

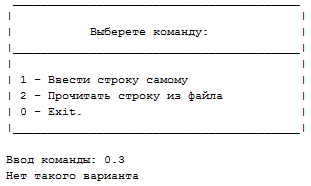


Рисунок 4 – результат теста 3

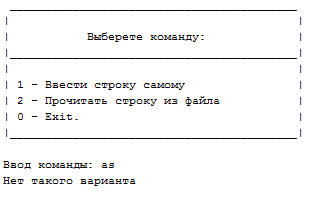


Рисунок 5 – результат теста 4

C:\Users\Akramont\Pictures\5.PNG

Рисунок 6 – результат теста 5

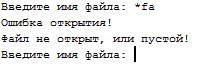


Рисунок 7 – результат теста 6

# 7 Исходный код

## 7.1 Исходный код класса Lab3

|  |
| --- |
| public class Lab3 {  public static void main(String [] args)  {  try{  Menu.vvod();  }  catch(IOException e){    }    }    } |

## 7.2 Исходный код класса Menu

|  |
| --- |
| public class Menu {      public static void vvod() throws IOException  {  boolean uslovie = true;  while (uslovie)  {  System.out.println(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ");  System.out.println("| |");  System.out.println("| Выберете команду: |");  System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");  System.out.println("| |");  System.out.println("| 1 - Ввести строку самому |");  System.out.println("| 2 - Прочитать строку из файла |");  System.out.println("| 0 - Exit. |");  System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");  System.out.print("\nВвод команды: ");    String vibor = "";  BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));  vibor = stdin.readLine();  switch (vibor)  {  case "1":  readScanChars();  break;  case "2":  readFileChars();  break;  case "0":  uslovie = false;  break;  default:  System.out.println("Нет такого варианта");  break;  }  }  }    public static void readFileChars() throws IOException  {  boolean uslovie = true;  while (uslovie)  {  System.out.print("Введите имя файла: ");  String fileName = "";  BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));  fileName = stdin.readLine();  String text = WFile.read(fileName);  if (!text.equals(""))  {  choiceTask(text);  uslovie = false;  }  else  {  System.out.println("Файл не открыт, или пустой!");  }  }    }    public static void writeFileChars(Chars chars) throws IOException  {  boolean uslovie = true;  while (uslovie)  {  System.out.print("Введите имя файла: ");  String fileName = "";  BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));  fileName = stdin.readLine();  uslovie = WFile.write(fileName, chars.print());  }  }    public static void readScanChars() throws IOException  {  System.out.println("Введите строку для работы: ");  String text = "";  BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));  text = stdin.readLine();  choiceTask(text);  }    public static void choiceTask(String text) throws IOException  {  Chars chars = new Chars(text);  boolean uslovie = true;  while (uslovie)  {  System.out.println(" \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ");  System.out.println("| |");  System.out.println("| Выбор варианта работы с текущей строкой |");  System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");  System.out.println("| |");  System.out.println("| 1 - Вывести текущую строку |");  System.out.println("| 2 - Найти символ в строке который встречается наибольшее число раз |");  System.out.println("| 3 - Найти самое длинное слово в строке |");  System.out.println("| 4 - Удалить в строке каждое третье слово |");  System.out.println("| 5 - Сохранить строку в файл |");  System.out.println("| 0 - Назад |");  System.out.println("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|");  System.out.print("\nВвод команды: ");    String str = "";  BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));  str = stdin.readLine();    switch (str)  {  case "1":  System.out.println("Текущая строка: ");  System.out.println(chars.print());  break;  case "2":  System.out.println("Количество одинаковых символов: " + chars.maxChar() + "\nСимвол(ы):");  System.out.println(chars.printCountChar(chars.maxChar()));  break;  case "3":  System.out.println("Длина слова: " + chars.maxWord() + "\nСловa:");  System.out.println(chars.printCountWord(chars.maxWord()));  break;  case "4":  chars = chars.deleteWords(3);  System.out.print("Удалено каждое " + 3 + " слово: ");  System.out.println(chars.print());  break;  case "5":  writeFileChars(chars);  break;  case "0":  uslovie = false;  break;  default:  System.out.println("Нет такого варианта");  break;  }  }  }  } |

## 7.3 Исходный код класса WFile

|  |
| --- |
| public class WFile{    public static boolean write(String fileName, String text) {  boolean usl = false;  try(FileOutputStream fos = new FileOutputStream(fileName)){  try (OutputStreamWriter osw = new OutputStreamWriter(fos)) {  osw.write(text);  osw.close();  System.out.println("Файл записан!");    }  catch(IOException ex)  {  System.out.println("Ошибка записи в файл!");  usl = true;  }  fos.close();  }  catch(IOException ex){  System.out.println("Ошибка!");  usl = true;  }  return usl;  }  public static String read(String fileName)  {  String text = "";  try(FileInputStream fis = new FileInputStream(fileName)){  try (InputStreamReader isr = new InputStreamReader(fis)) {  int data = isr.read();  while(data != -1){  char theChar = (char) data;  text += theChar;  data = isr.read();  }  isr.close();  System.out.println("Файл открыт!");  }  catch(IOException ex){  System.out.println("Ошибка чтения из файла!");  }  fis.close();  }  catch(IOException ex){  System.out.println("Ошибка открытия!");  }  return text;  }    } |

## 7.4 Исходный код класса Chars

|  |
| --- |
| public class Chars {    private int length = 0;  private char [] chars = new char[length];    public Chars()  {  this.chars = new char[] {'H','e','l','l','o'};  this.length = this.chars.length;  }    public Chars(char[] chars)  {  this.length = chars.length;  this.chars = chars;  }    public Chars(int length)  {  this.length = length;  this.chars = new char[this.length];  for (int i = 0; i<this.length; i++)  {  this.chars[i]=' ';  }  }    public Chars(String string)  {  this.length = string.length();  this.chars = new char[this.length];  for (int i = 0; i < this.length; i++)  {  this.chars[i] = string.charAt(i);  }  }    public Chars(Chars clone)  {  this.length = clone.length;  this.chars = clone.chars;  }  public Chars(int n, char ch)  {  this.length = n;  this.chars = new char[this.length];  for (int i = 0; i < this.length; i++)  {  this.chars[i] = ch;  }  }    public int getLength()  {  return length;  }    public char getChar(int i)  {  return this.chars[i];  }    public char[] getChars()  {  return this.chars;  }    public void setLength(int length)  {  this.length = length;  }    public void setChar(int i, char ch)  {  this.chars[i] = ch;  }    public void setChars(char[] ch)  {  this.chars = ch;  }    public String print()  {  String text = "";  for (int i=0; i<this.getLength(); i++)  {  text+=this.getChar(i);  }  return text;  }    public int maxChar()  {  int max = 0;  int count;  for (int i = 0; i<this.getLength(); i++)  {  count = 0;  for (int j = 0; j<this.getLength(); j++)  {  if (this.getChar(i) == this.getChar(j))  {  count++;  }  }  if (count > max)  {  max = count;  }  }  return max;  }    public String printCountChar(int count)  {  String text = "";  int c;  for (int i=0; i<this.getLength(); i++)  {  c = 0;  for (int j = i; j<this.getLength(); j++)  {  if (this.getChar(i) == this.getChar(j))  {  c++;  }  }  if (c == count)  {  text += "'" + this.getChar(i) + "'\n";  }  }  return text;  }    public int maxWord()  {  int max=0;  int count = 0;  for (int i=0; i<this.getLength(); i++)  {  count++;  if (this.getChar(i) == ',' || this.getChar(i) == ':' ||  this.getChar(i) == '.' || this.getChar(i) == ';' || this.getChar(i) == ' ')  {  count--;  if (count > max)  {  max = count;  }  count = 0;  }  if ( i == this.getLength()-1 && count > max)  {  max =count;  }  }  return max;  }    public String printCountWord(int count)  {  String text = "";  int c = 0;  char[] ch = new char[count+1];  for (int i=0; i<this.getLength(); i++)  {  c++;  ch[c-1] = this.getChar(i);  if (this.getChar(i) == ',' || this.getChar(i) == ':' ||  this.getChar(i) == '.' || this.getChar(i) == ';' || this.getChar(i) == ' ')  {  c--;  if (c == count)  {  for (int j=0; j<count; j++)  {  text+=ch[j];  }  text+='\n';  }  c = 0;  }  if ( i == this.getLength()-1 && c == count)  {  for (int j=0; j<count; j++)  {  text+=ch[j];  }  text+='\n';  }  }  return text;  }    public Chars deleteWords(int wordsDelete)  {  String text = "";  int three = 1;  for (int i=0; i<this.getLength(); i++)  {    if (this.getChar(i) == ' ' && three == wordsDelete)  {  three=0;  }  if (this.getChar(i) == ' ' && three != wordsDelete)  {  three++;  }  if (three != wordsDelete)  {  text+=this.getChar(i);  }  }  Chars ch = new Chars(text);  return ch;  }  } |

# Список использованных источников

1. Объектно-ориентированное программирование. Метод.указания по выполнению лабораторных работ / сост.: В.Л. Аршинский. – Иркутск : Изд-во ИРНИТУ, 2015. – 23 c.
2. Программирование на Java для детей, родителей, дедушек и бабушек/ Файн Я.