Министерство науки и образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт информационных технологий и анализа данных

РАЗРАБОТКА КЛАССОВ И ИХ ИСПОЛЬЗОВАНИЕ

Отсчёт по лабораторной работе №2

Вариант 14

По дисциплине: «Объектно-ориентированное программирование»

Выполнил студент

группы АСУб-19-1 \_\_\_\_\_\_\_\_\_\_\_ Д. А. Вологдин

Принял

Доцент \_\_\_\_\_\_\_\_\_\_\_  Т.В. Маланова

г. Иркутск 2020

**Оглавление**

[1 Постановка задачи 3](#_Toc32754747)

[2 Описание](#_Toc32754748) структуры классов [4](#_Toc32754748)

[3 Описание](#_Toc32754749) методов классов и спецификации локальных переменных [7](#_Toc32754749)

[4 Таблица тестов 8](#_Toc32754750)

[5 Результаты тестирования 9](#_Toc32754751)

[6 Исходный код 10](#_Toc32754752)

[Список использованных источников 11](#_Toc32754753)

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 экземпляру класса, решающему задачу, должны быть разработаны соответствующие методы.

Задание варианта:

1. Отсортируйте слова в массиве по их длине.
2. Подсчитайте количество предложений в заданном тексте.
3. Дана строка, содержащая слова, разделённые пробелами, запятыми, восклицательными знаками, вопросительными знаками. Отсортируйте слова по их длине.

Математическая модель:

1. Разделить строку на отдельные слова, составить из полученных слов массив и отсортировать полученный массив по длине слов
2. С помощью цикла проверить каждый символ текста, если будет найден символ заканчивающий предложение, прибавить к количеству предложений 1.
3. С помощью цикла проверить строку, и при нахождении разделяющего символа убрать этот символ путём конкатенации части строки до символа и части строки после символа. Разделить строку на слова, составить из слов массив и отсортировать его по длине слов.

2 Описание структуры классов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название | Назначение | Поля (атрибуты) |
| 1 | Public class L214 | Главный класс | - |
| 2 | Public class A\_task | Класс, выполняющий первое задание | Private String [][] enter\_arr - массив строк |
| 3 | Public class B\_task | Класс, выполняющий второе задание | Private String text – Введённый текст |
| 4 | Public class C\_task | Класс, выполняющий третье задание | Private String str – Введённая строка |

3 Описание методов классов и спецификации локальных переменных

Таблица 1 – Описание методов и классов

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название метода/поля | Название | Тип |
| Public class L214 | | | |
| 1 | public static void main(String[] args) | Главный метод | Процедура |
| Publick class A\_task | | | |
| 1 | Public A\_task | Конструктор по умолчанию | Процедура |
| 2 | Public A\_task(String enter) | Конструктор с исходной строкой |
| 3 | Public A\_task(A\_task cop) | Конструктор копирования |
| 4 | Public void setArr(String [][] str) | Устанавливающий метолд |
| 5 | Public void sort() | Метод выполняющий задание |
| 6 | Public String [][] getArr() | Функция чтения | Функция |
| Public class B\_task | | | |
| 1 | Public B\_task | Конструктор по умолчанию | Процедура |
| 2 | Public B\_task(String enter) | Конструктор с исходной строкой |
| 3 | Public B\_task(A\_task cop) | Конструктор копирования |
| 4 | Public void setStr(String [][] str) | Устанавливающий метолд |
| 5 | Public void counter() | Метод выполняющий задание |
| 6 | Public String [][] getStr() | Функция чтения | Функция |
| Public class C\_task | | | |
| 1 | Public C\_task | Конструктор по умолчанию | Процедура |
| 2 | Public C\_task(String enter) | Конструктор с исходной строкой |
| 3 | Public C\_task(A\_task cop) | Конструктор копирования |
| 4 | Public void setStr(String [][] str) | Устанавливающий метолд |
| 5 | Public void sort\_c() | Метод выполняющий задание |
| 6 | Public String [][] getStr() | Функция чтения | Функция |

Таблица 2 – Таблица спецификаций

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Имя переменной | Описание переменных | Тип данных | ОДЗ | Методы |
| 1 | Enter\_str | Исходная строка | String | [‘A’..’z’] | L214 A\_task |
| 2 | str\_arr | Массив слов исходной строки | String[] | [‘A’..’z’] | C\_task |
| 3 | text | Исходный текст, или строка | String | [‘A’..’z’] | L214 B\_task  C\_task |
| 4 | Enter\_pre\_arr | Массив слов исходной строки или текста | String | [‘A’..’z’] | A\_task |
| 5 | Prosh\_1 | Временная строка для перестановки в массиве строк | String | [‘A’..’z’] | A\_task |
| 6 | Prosh\_2 | Временная строка для перестановки в массиве строк | String | [‘A’..’z’] | A\_task |
| 7 | Number\_str | Количество предложений в тексте | int | [-2147483647.. -2147483647] | B\_task |
| 8 | i | Индекс символа в строке | int | [1..str.length()] | A\_task  B\_task  C\_task |

4 Таблица тестов

Таблица 2 – Таблица тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Описание | Входные данные | | Выходные данные |
| 1 | Сортировка слов в массиве  (a\_task) | Sentence | «Я слов на ветер не бросаю. Это мой путь ниндзя» | «Слова отсортированные по их длине: Я на не Это мой слов путь ветер ниндзя бросаю.» |
| 2 | Подсчёт кол-ва предложений в тексте  (b\_task) | Sentence | «Ничего не понимаю! Ладно, притворюсь, что понял.» | Кол-во предложений в тексте: 2 |
| 3 | Сортировка слов в строке  (c\_task) | Sentence | “На!ошибках.учатся,после?ошибок.лечатся.” | «Отсортированная строка: На после учатся ошибок ошибках лечатся» |
| 4 | Сортировка слов в массиве  (a\_task) | Sentence | «Только начнешь работать, обязательно кто-нибудь разбудит.» | «Слова отсортированные по их длине: Только начнешь работать, разбудит. кто-нибудь обязательно» |
| 5 | Подсчёт кол-ва предложений в тексте  (b\_task) | Sentence | «Мало сказать глупость, надо чтобы в нее поверили. Боль — это боль, как ее не назови. Разве она становится слабее от того, что ты думаешь, что она нереальна?» | Кол-во предложений в тексте: 3 |
| 6 | Сортировка слов в строке  (c\_task) | Sentence | «Люди?становятся?сильнее,благодаря?воспоминаниям,которые,не,могут,забыть.Это!и!есть.взросление.» | «Отсортированная строка: и не Это Люди есть могут забыть сильнее которые благодаря становятся взросление воспоминаниям» |

5 Результаты тестирования

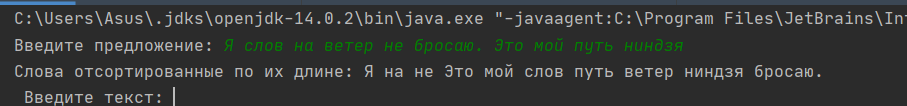


Рисунок 2.1 – Тест 1

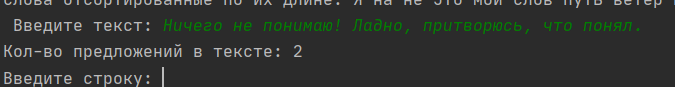


Рисунок 2.2 – Тест 1

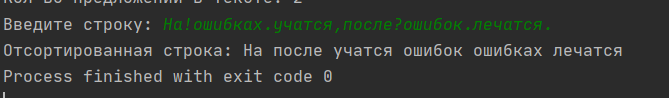


Рисунок 2.3 – Тест 1

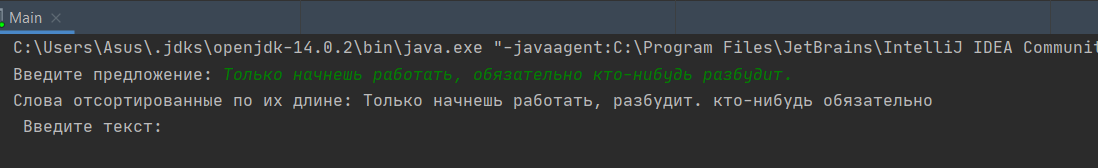


Рисунок2.4 – Тест 2

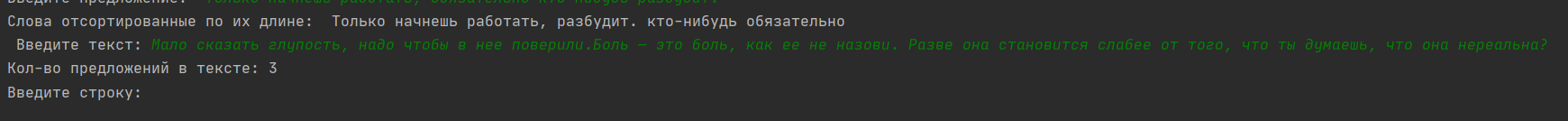


Рисунок 2.5 – Тест 2

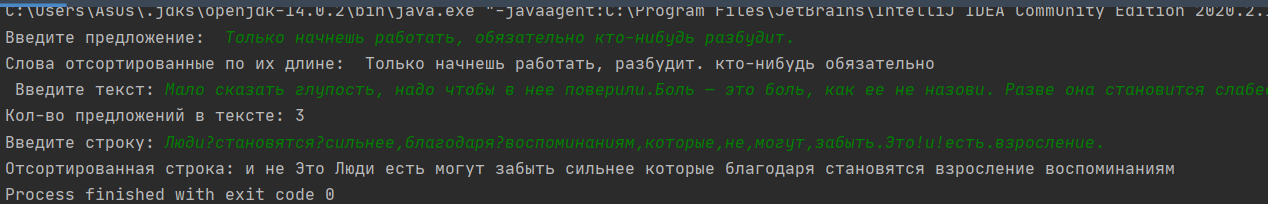


Рисунок 2.6 – Тест 2

6 Исходный код

L214

package L214;  
import java.util.Scanner;  
  
public class L214 {  
  
 public static void main(String[] args){  
 //Создаём объект класса Scanner для считывания строк  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.print("Введите предложение: ");  
 String enter\_str = in.nextLine();  
 //Преобразуем строку в массив  
 A\_task a = new A\_task(enter\_str);  
 //Сортируем массив по длине слов  
 a.sort();  
 System.*out*.print("\n Введите текст: ");  
 String text = in.nextLine();  
 B\_task b = new B\_task(text);  
 //Проверяем каждый элемент строки  
 b.counter();  
 System.*out*.print("Введите строку: ");  
 text = in.nextLine();  
 //Поиск ненужных симовлов в строке и их удоление  
 C\_task c = new C\_task(text);  
 //Из готовой строки делаем массив и сортируем слова в нём  
 c.sort\_c();  
 in.close();  
 }  
}

A\_task

package L214;  
  
public class A\_task {  
  
  
 private String [][] enter\_arr;  
  
 public A\_task(String enter\_str){  
 String[] enter\_pre\_arr = enter\_str.split(" ");  
 this.enter\_arr = new String[2][enter\_pre\_arr.length];  
 for (byte i = 0; i < enter\_pre\_arr.length; i ++){  
 this.enter\_arr[0][i] = enter\_pre\_arr[i];  
 this.enter\_arr[1][i] = Integer.*toString*(enter\_pre\_arr[i].length());  
 }  
  
 }  
  
 public A\_task(A\_task cop){  
 this.enter\_arr = cop.enter\_arr;  
 }  
  
 public A\_task(){  
 this.enter\_arr = new String[2][10];  
 this.enter\_arr[0][1] = "Test";  
 this.enter\_arr[0][2] = "string";  
 this.enter\_arr[1][1] = "4";  
 this.enter\_arr[1][2] = "6";  
 }  
  
 public void setArr(String [][] str){  
 this.enter\_arr = str;  
 }  
  
 public String [][] getArr(){  
 return this.enter\_arr;  
 }  
  
 public void sort(){  
 for (int i = 0; i < this.enter\_arr[0].length; i++){  
 for(int j = 0; j < this.enter\_arr[0].length - 1; j++){  
 if (Integer.*parseInt*(this.enter\_arr[1][j]) > Integer.*parseInt*(this.enter\_arr[1][j + 1])){  
 String prosh = this.enter\_arr[0][j + 1];  
 String prosh2 = this.enter\_arr[1][j + 1];  
 this.enter\_arr[0][j + 1] = this.enter\_arr[0][j];  
 this.enter\_arr[1][j + 1] = this.enter\_arr[1][j];  
 this.enter\_arr[0][j] = prosh;  
 this.enter\_arr[1][j] = prosh2;  
 }  
 }  
 }  
 System.*out*.print("Слова отсортированные по их длине: ");  
 for (byte i = 0; i < this.enter\_arr[0].length; i++) {  
 System.*out*.print(this.enter\_arr[0][i] + " ");  
 }  
 }  
  
}

B\_task

package L214;  
  
public class B\_task {  
  
 private String text;  
  
 public B\_task(){  
 this.text = "Test. Test? Test!";  
 }  
  
 public B\_task(B\_task te){  
 this.text = te.text;  
 }  
  
 public B\_task(String text){  
 this.text = text;  
 }  
  
 public void setStr(String text){  
 this.text = text;  
 }  
  
 public String getStr(){  
 return this.text;  
 }  
  
 public void counter(){  
 byte number\_str = 0;  
 for (int i = 0; i < this.text.length(); i++){  
 if (this.text.charAt(i) == '.' || this.text.charAt(i) == '!' || this.text.charAt(i) == '?') {  
 number\_str++;  
 }  
 }  
 if (number\_str != 0) System.*out*.print("Кол-во предложений в тексте: " + number\_str + "\n");  
 else System.*out*.print("Предложений нет \n");  
 }  
}

C\_task

package L214;  
import java.util.Arrays;  
import java.util.Comparator;  
  
public class C\_task {  
  
 private String str;  
  
 public C\_task(){  
 this.str = "Test,string.for!testing?program";  
 }  
  
 public C\_task(C\_task str){  
 this.str = str.str;  
 }  
  
 public C\_task(String text){  
 this.str = text;  
 }  
  
 public void setStr(String text){  
 this.str = text;  
 }  
  
 public String getStr(){  
 return this.str;  
 }  
  
 public void sort\_c(){  
 this.str = this.str.replaceAll(",", " ");  
 this.str = this.str.replaceAll("\\.", " ");  
 this.str = this.str.replaceAll("!", " ");  
 this.str = this.str.replaceAll("\\?", " ");  
 String[] str\_arr = this.str.split(" ");  
 Arrays.*sort*(str\_arr, Comparator.*comparing*(String::length));  
 System.*out*.print("Отсортированная строка: ");  
 for (byte i = 0; i < str\_arr.length; i++){  
 System.*out*.print(str\_arr[i] + " ");  
 }  
 }  
}

Список использованных источников

1. Объектно-ориентированное программирование. Метод. указания по

выполнению лабораторных работ / сост.: В.Л. Аршинский. – Иркутск : Изд-во

ИРНИТУ, 2017. – 24 c.