Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Институт Информационных Технологий и Анализа Данных

Кафедра вычислительной техники

**Название работы** – Работа с потоками данных и обработка исключений

Отчет по лабораторной работе № 3

Вариант 2

по дисциплине Объектно-ориентированное программирование

Выполнил

Студент, номер группы ИСМб-19-1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. Д. Солопов

Дата: \_\_\_\_\_\_\_\_\_\_

Принял

Должность \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Т.В. Маланова

Дата: \_\_\_\_\_\_\_\_\_\_

Иркутск 2020

Содержание

[1 Постановка задачи 3](#_Toc53041476)

[2 Описание структуры пользовательского меню 4](#_Toc53041477)

[3 Проектирование классов 6](#_Toc53041478)

[4 Описание методов класса и спецификация локальных переменных 9](#_Toc53041479)

[5 Таблица тестов 12](#_Toc53041480)

[6 Листинг исходного кода 19](#_Toc53041481)

[7 Заключение 31](#_Toc53041482)

[8 Список использованных источников 32](#_Toc53041483)

1 Постановка задачи

Провести объектную декомпозицию задачи из индивидуального задания и разработать класс, содержащий соответствующие поля для хранения необхо-димых данных и методы, обеспечивающие достаточную для решения задачи функциональность класса. Разработать класс, реализующий текстовое меню, позволяющее осуществлять выбор вариантов выполнения индивидуального за-дания. Создание объекта класса решающего задание и вызов его методов долж-ны осуществляться из класса реализующего меню. При выполнении задания предусмотреть методы для организации ввода и выводы данных, как на кон-соль, так и в текстовые файлы. Организация работы с файлами может быть ор-ганизована методами класса решающего индивидуальное задание, либо мето-дами отдельного специально разработанного класса. Для всех проверяемых ис-ключений должны быть реализованы обработчики, в которых выводится сообщение о возникновении исключительной ситуации и предпринимаются меры для продолжения работы программы в штатном режиме.

Индивидуальное задание выполняется на основе заданий к лабораторной работе 1.

**a.** Определите, является ли одна строка подстрокой другой.

**b.** Найти длину указанного слова в предложении.

**c.** Выведите из строки, содержащей слова, разделенные пробелами и запятыми, центральное слово (если в предложении два централь-ных слова, выведите оба).

2 Описание структуры пользовательского меню

Меню состоит из пунктов, каждый из которых является целым натуральным числом. Для начала, меню программы предоставляет пользователю интерфейс выбора решения определённого задания, согласно индивидуальному варианту. При вводе не корректного числа или числа не принадлежащего диапазону чисел, составляющих пункты, пользователю предлагается ввести ещё раз свой выбор. Данный способ ввода применяется везде, где необходимо ввести число.

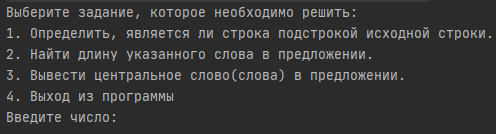


Рисунок 1 – Пользовательское меню

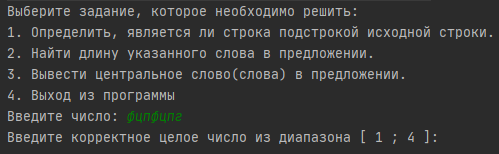


Рисунок 2 – Результат ввода не корректного числа

Далее, после выбора задания для решения, пользователю предлагается выбрать способ чтения данных, всего таких способа два: из файла или из консоли. Также при выборе варианта чтения данных следует учесть требование программы для данных, считываемых из файла: исходная строка и дополнительные данные (подстрока / порядковый номер слова) должны быть разделены символом \n, иначе Enter.

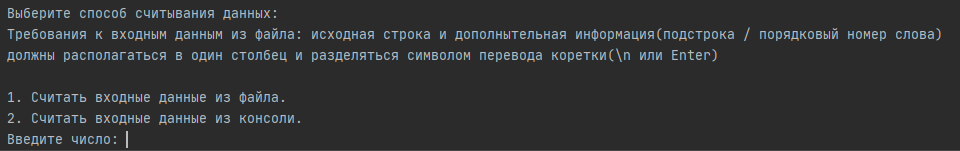


Рисунок 3 – Выбор чтения входных данных

Далее, пользователю предлагается выбрать способ вывода результата решения задания. Всего таких способа два: в файл или в консоль.

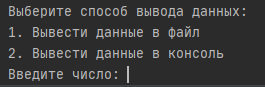


Рисунок 4 – Выбор вывода результата решения задания

Далее, согласно пользовательскому решению ввода/вывода данных осуществляется работа программы. На рисунке 5 будет приведён пример работы программы, при решении задания под номером 1, с чтением данных из консоли и ввод данных в консоль.

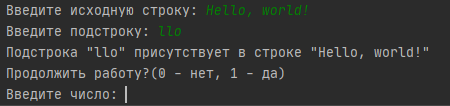


Рисунок 5 – Результат выполнения индивидуального задания под номером 1 с полным взаимодействием с консолью

После решения задания пользователю предлагается на выбор следующее развитие событий, связанное с эксплуатацией программы: не продолжать работу или продолжать работу. Пользователю нужно ввести число 0, в случае, если он решает прекратить работу программы и 0 в том случае, если он работу программы не прекращает. В случае продолжения работы программы пользователю выведется первоначальное меню, представленное на рисунке 1.

3 Проектирование классов

Программа содержит 4 класса, один из которых – Main, является точкой входа в программу.

Описание классов:

1. Main – точка входа в программу
2. StringClass – класс, решающий задания согласно индивидуальному варианту
3. FileIO – класс, реализующий логику ввода/вывода данных в файл / из файла.
4. MenuClass – класс, предоставляющий пользователю консольный интерфейс.

Класс Main не обрабатывает исключения и работает только с классом MenuClass.

Класс StringClass не содержит функций, из которых можно поймать исключение (throws). В классе StringClass выбрасывать исключения не имеет смысла, поскольку методы класса только принимают входные данные, которые в свою очередь вводит пользователь(т.е. используя определённый интерфейс), и возвращает результат вычислений.

Класс FileIO имеет два метода возвращающих общее исключение Exception, содержащая определённое сообщение об ошибке и указании на факт перевода работы программы в штатный режим. Методы класса FileIO возвращающие исключения: inputDataInFile и outputDataFromFile.

Класс MainClass реализует обработку исключений, которые возвращаются из методов класса FileIO и исключений, при определённых ситуациях связанных с вводом данных или их преобразованием.

Таблица 1 – Методы классов выбрасывающие исключения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Имя метода** | **Назначение** | **Класс** | **Исключение** | **Назначение исключения** |
| 1 | inputDataInFile | Запись данных в файл | FileIO | Exception | Выбрасывается при указании не корректного пути к файлу |
| 2 | outputDataFromFile | Чтение данных из файла | FileIO | Exception | Выбрасывается при указании не корректного пути к файлу, при не существовании файла. |

Таблица 2 – Методы классов обрабатывающие исключения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Имя метода** | **Назначение** | **Класс** | **Исключение** | **Назначение обработки** |
| 1 | outputMenu | Вывод пользовательского интерфейса | MenuClass | Exception | Перевод программы в штатный режим, в случае не корректного указания пути к файлу для ввода/вывода |
| 2 | outputMenu | Вывод пользовательского интерфейса | MenuClass | NumberFormatException | Перевод программы в штатный режим, в случае не корректного чтения порядкового номера слова из файла |

**Словесное описание класса Main:**

Класс Main содержит в себе один статический метод main(), который является точкой входа в программу.

**Словесное описание класса StringClass:**

Класс содержит одно поле типа String – text, которое является исходной строкой, с которой методы класса осуществляют операции с целью решить поставленную задачу согласно индивидуальному варианту, пять методов: substringInString(string) – решает задачу a, lengthDefineWord(number) – решает задачу b, centralWords() – решает задачу c, setText(string) – метод set для text(записывает данные в поле text), getText() – метод get для text(возвращает данные text) и 3 конструктора: StringClass() – конструктор по умолчанию, StringClass(string) – конструктор с параметром, StringClass(StringClass) - конструктор копирования.

**Словесное описание класса FileIO:**

Класс FileIO содержит в себе 3 статических метода, 1 из которых имеет спецификацию доступа private: (private)checkFilePath() – проверяет путь к файлу на корректность, (public)inputDataInFile() – записывает данные в файл, (public)outputDataFromFile() – считывает данные из файла.

**Словесное описание класса MenuClass:**

Класс MenuClass содержит в себе 2 статических и 2 не статических метода. Все метода класса имеют спецификатор доступа public: (static)isNumber() – проверяет является ли входная строка числом, (static)getCorrectValue() – осуществляет корректный ввод числовых данных(данные вводятся до тех пор, пока не будут введены корректные данные), (non static)outputMenu() – выводит пользовательский интерфейс, (non static)startMenu() – запускает цикл многопериодичного вывода пользовательского интерфейса.

4 Описание методов класса и спецификация локальных переменных

Таблица 3 – Описание методов класса и спецификация локальных переменных

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя** | **Спецификатор** | **Входные данные** | **Назначение** | **Тип** | **Выходные данные** | **Назначение** | **Тип** |
| **Класс Main** | | | | | | | | |
| 1 | main | public | args | Аргументы командной строки | String[] | answers | Результат выполнения работы программы | String[] |
| **Класс StringClass** | | | | | | | | |
| 1 | StringClass | public | - | - | - | - | - | - |
| 2 | StringClass | public | \_text | Строка, для копирование в поле класса text | String | text | Строка, для работы объекта класса | String |
| 3 | StringClass | public | obj | Объект, для копирования | StringClass | text | Строка, для работы объекта класса | String |
| 4 | setText | public | \_text | Строка, для копирование в поле класса text | String | text | Строка, для работы объекта класса | String |

Продолжение таблицы 3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | getText | public | - | - | - | text | Строка, для работы объекта класса | String |
| 6 | substringInString | public | substring | Строка для поиска | String | flag | Результат поиска | boolean |
| 7 | lengthDefineWord | public | number | Порядковый номер слова | int | lengthWord | Длина искомого слова | int |
| 8 | centerWords | public | - | - | - | results | Результат поиска | String |
| **Класс FileIO** | | | | | | | | |
| 1 | checkFilePath | private | filePath | Полный путь к файлу | String | check | Результат проверки | boolean |
| 2 | inputDataInFile | public | filePath  data  create | Полный путь к файлу  Данные для записи  Флаг для создания файла | String  String  boolean | - | - | - |
| 3 | outputDataFromFile | public | filePath | Полный путь к файлу | String | data | Результат считывания данных из файла | String |

Методы класса StringClass() используют встроенный метод trim(), доступ к которому имеет любой объект класса String. Данный метод используется для удаления символов пробела в начале и в конце строки.

Продолжение таблицы 3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя** | **Спецификатор** | **Входные данные** | **Назначение** | **Тип** | **Выходные данные** | **Назначение** | **Тип** |
| **Класс MenuMain** | | | | | | | | |
| 1 | isNumber | public | text | Текст содержащий число | String | flag | Результат выполнения проверки | boolean |
| 2 | getCorrectValue | public | min  max | Минимальное возможное число  Максимальное возможное число | int  int | value | Результат выполнения консольного ввода | int |
| 3 | outputMenu | public | - | - | - | - | - | - |
| 4 | startMenu | public | - | - | - | - | - | - |

5 Таблица тестов

**Задания:**

**a.** Определите, является ли одна строка подстрокой другой.

**b.** Найти длину указанного слова в предложении.

**c.** Выведите из строки, содержащей слова, разделенные пробелами и запятыми, центральное слово (если в предложении два централь-ных слова, выведите оба).

Таблица 4 – Таблица тестов для решения задания a

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Ввод с консоли, вывод в консоль | text = “Hello, world!”  substring = “Hello” | Подстрока "Hello" присутствует в строке "Hello, world!" (Рис. 6) |
| 2 | Чтение с файла, вывод на консоль | text = “Высказывание, предложение, местоимение”  substring = “предложение”(Рис. 7) | Подстрока "предложение" присутствует в строке "﻿Высказывание, предложение, местоимение" (Рис. 8) |

Продолжение таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | Ввод с консоли, вывод в файл | text = “Hello, world!”  substring = “автомобиль” | (Рис. 9 - 10) |
| 4 | Чтение с файла, вывод в файл | text = “Высказывание, предложение, местоимение”  substring = “предложение”  (Рис. 7) | (Рис. 11 - 12) |

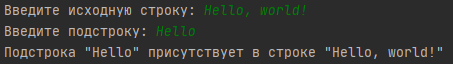


Рисунок 6 – Результат теста №1 из таблицы тестов №4

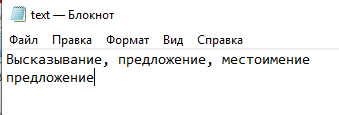


Рисунок 7 – Содержание файла C:\TESTS\text.txt для чтения данных

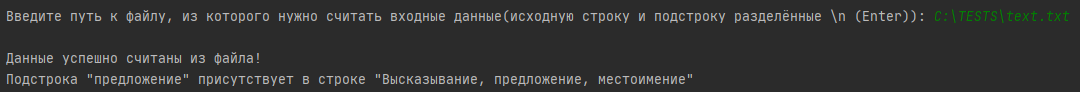


Рисунок 8 – Результат теста №2 из таблицы тестов №4

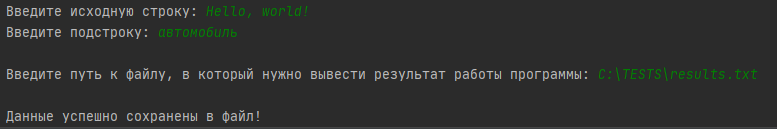


Рисунок 9 – Результат теста №3 из таблицы тестов №4

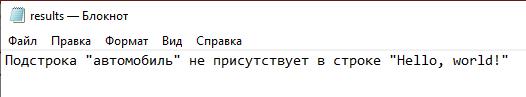


Рисунок 10 – Содержание файла C:\TESTS\results.txt выполнения теста №3 из таблицы тестов №4

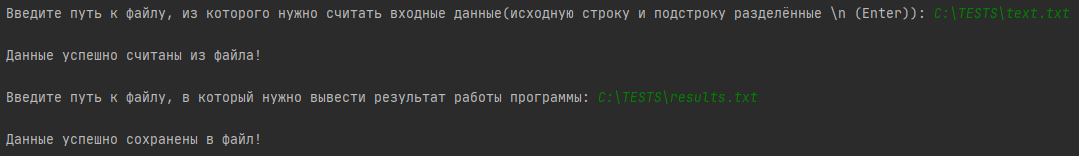


Рисунок 11 – Результат теста №4 из таблицы тестов №4

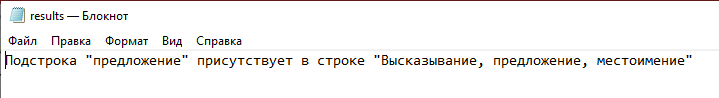


Рисунок 12 – Содержание файла C:\TESTS\results.txt выполнения теста №4 из таблицы тестов №4

Таблица 5 – Таблица тестов для решения задания b

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Ввод с консоли, вывод в консоль | text = “abc, abcd, abcde, a, b”  number = 5 | Слово с порядковым номером 5 имеет длину 1 (Рис. 13) |
| 2 | Чтение с файла, вывод на консоль | text = “Hello, world! Hello friend!”  number = 4  (Рис. 14) | Слово с порядковым номером 4 имеет длину 7 (Рис. 15) |
| 3 | Ввод с консоли, вывод в файл | text = “Высказывание – повествовательное предложение”  number = 3 | (Рис. 16 – 17) |
| 4 | Чтение с файла, вывод в файл | (Рис. 14) | (Рис. 18-19) |

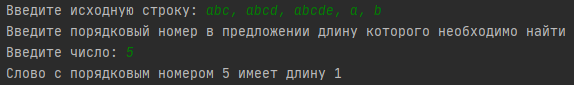


Рисунок 13 – Результат теста №1 из таблицы тестов №5

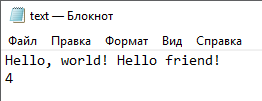


Рисунок 14 - Содержание файла C:\TESTS\text.txt для чтения данных

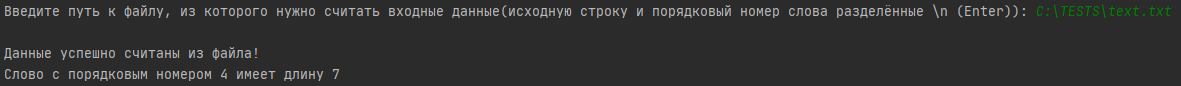


Рисунок 15 – Результат теста №2 из таблицы тестов №5

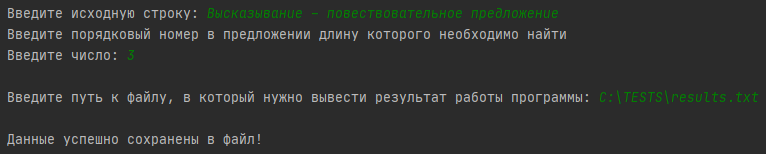


Рисунок 16 – Результат теста №3 из таблицы тестов №5

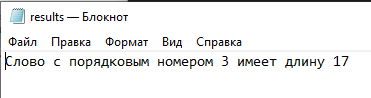


Рисунок 17 – Содержание файла C:\TESTS\results.txt после выполнения теста №3 из таблицы тестов №5

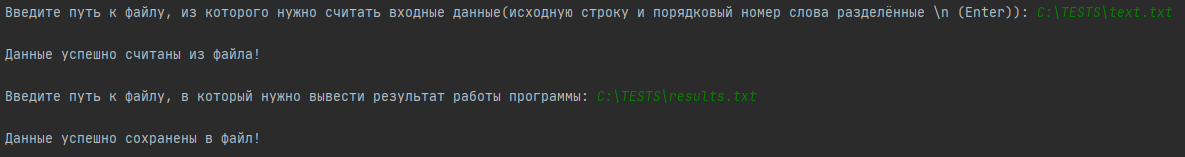


Рисунок 18 – Результат теста №4 из таблицы тестов №5

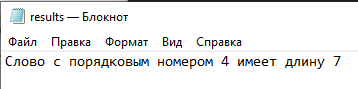


Рисунок 19 – Содержание файла C:\TESTS\results.txt после выполнения теста №4 из таблицы тестов №5

Таблица 6 – Таблица тестов для решения задания c

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Ввод с консоли, вывод в консоль | text = “abc, abcd, abcde, a, b” | Центральное слово/слова в предложении: abcde (Рис. 20) |
| 2 | Чтение с файла, вывод на консоль | text = “Hello, world! Hello friend! Tower of country. It is good tower”  (Рис. 21) | Центральное слово/слова в предложении: of (Рис. 22) |
| 3 | Ввод с консоли, вывод в файл | text = “Здравствуйте коллеги! Спешу представить вашему вниманию новую разработку нашей компании” | (Рис. 23 – 24) |
| 4 | Чтение с файла, вывод в файл | (Рис. 21) | (Рис. 25-26) |



Рисунок 20 - Результат теста №1 из таблицы тестов №6

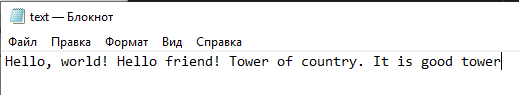


Рисунок 21 - Содержание файла C:\TESTS\text.txt для чтения данных

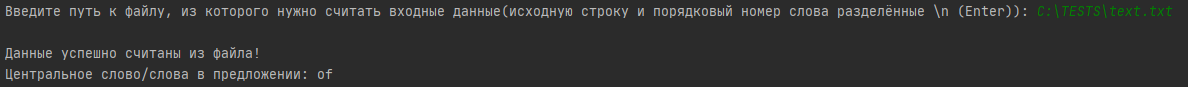


Рисунок 22 - Результат теста №2 из таблицы тестов №6

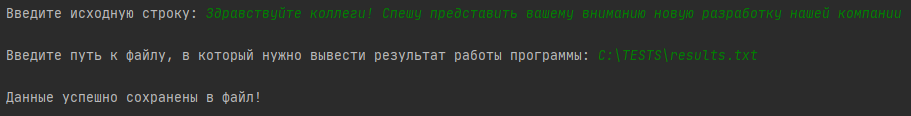


Рисунок 23 - Результат теста №3 из таблицы тестов №6

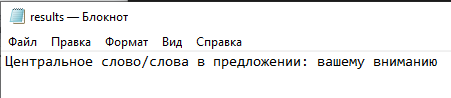


Рисунок 24 – Содержание файла C:\TESTS\results.txt после выполнения теста №3 из таблицы тестов №6

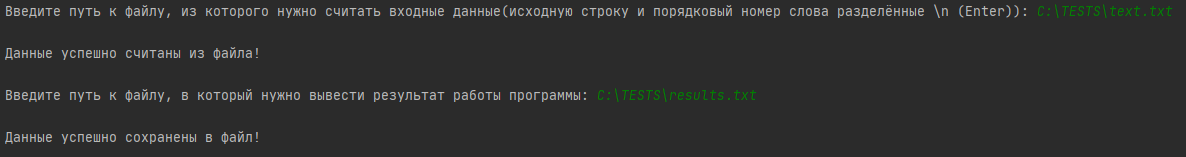


Рисунок 25 - Результат теста №4 из таблицы тестов №6

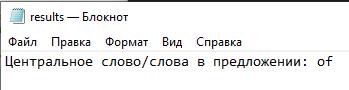


Рисунок 26 - Содержание файла C:\TESTS\results.txt после выполнения теста №4 из таблицы тестов №6

Таблица 7 – Таблица тестов обработки исключительных ситуаций

|  |  |  |  |
| --- | --- | --- | --- |
| **N теста** | **Назначение теста** | **Входные данные для теста** | **Выходные данные для теста** |
| 1 | Не корректное чтение из файла(файла по данному пути не существует), вывод осуществляется в консоль | (обработка исключительной ситуации для задания a)  text = “abc d e f”  substr = “d” | Подстрока “d” присутствует в строке “abc d e f” (Рис. 27) |
| 2 | Не корректное чтение из файла(в файле были не корректные данные (рис. 28)) | (обработка исключительной ситуации для задания b)  text = “abc, def, ghi, jklf”  number = 4 | Слово с порядковым номером 4 имеет длину 4 (Рис. 29) |
| 3 | Не корректный путь к файлу вывода, данные вводятся из консоли | (обработка исключительной ситуации для задания c)  text = “abcd efgh ijkl mnop rsty” | Центральное слово/слова в предложении: ijkl (Рис. 30) |
| 4 | Не корректный путь к файлу записи и вывода | (обработка исключительной ситуации для задания b)  text = “hello, world! It is my car”  number = 3 | Слово с порядковым номером 3 имеет длину 2 (Рис. 31) |

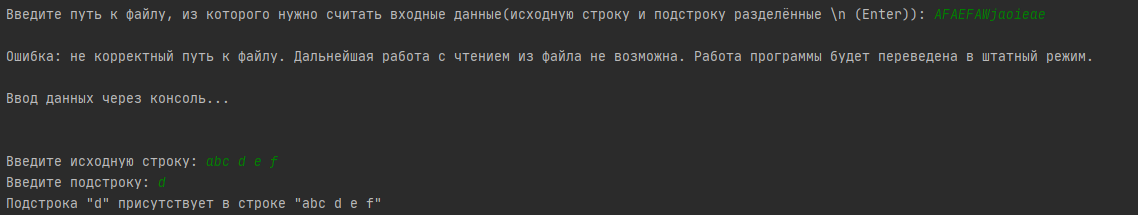


Рисунок 27 – Результат теста №1 из таблицы тестов №7

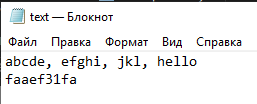


Рисунок 28 – Данные содержащиеся в файле C:\TESTS\text.txt до начала выполнения теста №2 из таблицы тестов №7

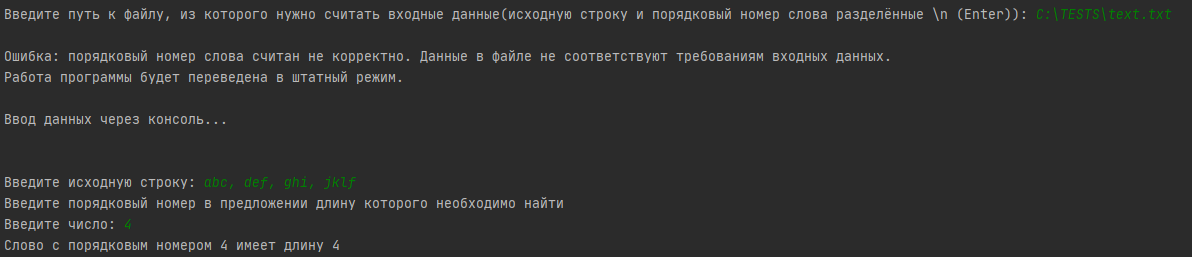


Рисунок 29 – Результат теста №2 из таблицы тестов №7

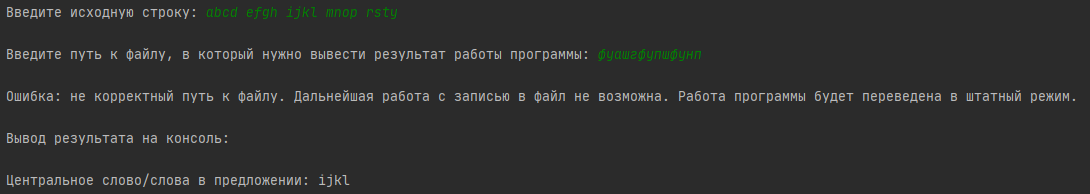


Рисунок 30 – Результат теста №3 из таблицы тестов №7

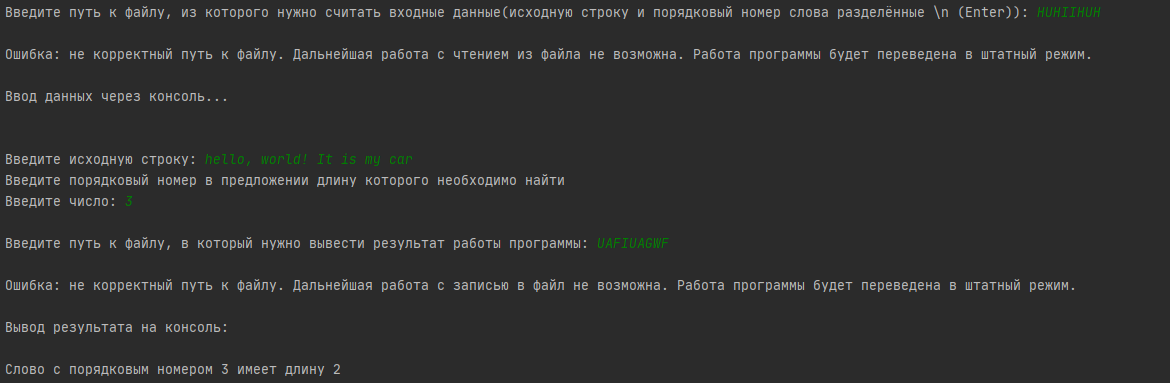


Рисунок 31 – Результат теста №4 из таблицы тестов №7

6 Листинг исходного кода

**Класс StringClass:**

package com.company;

public class StringClass {

private String text;

public StringClass(){ //Конструктор по умолчанию

text = "";

}

public StringClass(String \_text){ //Конструктор с параметром

\_text = \_text.trim();

text = \_text;

}

public StringClass(StringClass obj){ //Конструктор копирования

text = obj.text;

}

public void setText(String \_text){ //Метод для задания данных

\_text = \_text.trim();

text = \_text;

}

public String getText(){ //Метод для взятия данных

return text;

}

public boolean substringInString(String substring) {

/\*

a. Определите, является ли одна строка подстрокой другой.

\*/

substring = substring.trim();

boolean flag = false;

for(int i = 0; (i < text.length()) && (!flag); i++)

if(text.charAt(i) == substring.charAt(0)){

int count = 0;

for(int k = 0; k < substring.length(); k++)

if(substring.charAt(k) == text.charAt(k + i))

count++;

if(count == substring.length())

flag = true;

}

return flag;

}

public int lengthDefineWord(int number) {

/\*

b. Найти длину указанного слова в предложении.

\*/

int counter = 1;

int lengthWord = 0;

for(int i = 0; i < text.length(); i++){

if((i >= 1) && ((text.charAt(i-1) == ' ') || (text.charAt(i-1) == ','))

&& ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

counter++;

if((counter == number) && ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

lengthWord++;

}

return lengthWord;

}

public String centerWords () {

//c. Выведите из строки, содержащей слова, разделенные пробелами и запятыми, центральное слово

// (если в предложении два централь-ных слова, выведите оба).

String results = "";

int countWords = 1;

for(int i = 1; i < text.length(); i++)

if(((text.charAt(i - 1) == ' ') || (text.charAt(i-1) == ','))

&& ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

countWords++;

int number = (int)(countWords / 2), counter = 1;

if((countWords % 2) != 0)

number++;

for(int i = 0; i < text.length(); i++){

if((i >= 1) && ((text.charAt(i-1) == ' ') || (text.charAt(i-1) == ','))

&& ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

counter++;

if((counter == number) && ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

results += text.charAt(i);

}

if((countWords % 2) == 0){

results += ' ';

counter = 1;

number++;

for(int i = 0; i < text.length(); i++){

if((i >= 1) && ((text.charAt(i-1) == ' ') || (text.charAt(i-1) == ','))

&& ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

counter++;

if((counter == number) && ((text.charAt(i) != ' ') && (text.charAt(i) != ',')))

results += text.charAt(i);

}

}

return results;

}

}

**Класс Main:**

package com.company;

public class Main{

public static void main(String[] args) {

MenuClass menu = new MenuClass();

menu.startMenu();

}

}

**Класс FileIO:**

package com.company;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

public class FileIO {

private static boolean checkFilePath(String filePath){

boolean check = false;

if((filePath.length() == 0)

|| (filePath.charAt(0) == ':')

|| (filePath.indexOf(':') != filePath.lastIndexOf(':'))

|| (filePath.indexOf(':') == (-1))

|| ((filePath.indexOf('\\') != (-1)) && (filePath.indexOf('/') != (-1)))

|| ((filePath.indexOf('\\') == (-1)) && (filePath.indexOf('/') == (-1)))

|| ((filePath.indexOf('\\') != (-1)) && filePath.indexOf('\\') < filePath.indexOf(':'))

|| ((filePath.indexOf('/') != (-1)) && filePath.indexOf('/') < filePath.indexOf(':'))

|| (filePath.indexOf('.') != filePath.lastIndexOf('.'))

)

return check;

char[] symbols = {'|', '?', '\*', '<', '>'};

for(char symbol : symbols)

if(filePath.indexOf(symbol) != (-1))

return check;

check = true;

return check;

}

public static void inputDataInFile(String filePath, String data, boolean create) throws Exception{

if((!checkFilePath(filePath)))

throw new Exception("\nОшибка: не корректный путь к файлу. Дальнейшая работа с записью в файл не возможна." +

" Работа программы будет переведена в штатный режим. \n");

if(create == true){

String[] cellFilePath = filePath.replace('\\', '/').split("\\/");

String directory = "";

for(int i = 0; i < (cellFilePath.length-1); i++)

directory += cellFilePath[i] + "\\";

File file = new File(filePath);

if((!file.exists()) && (!(new File(directory).mkdir())) && (!file.createNewFile()))

throw new Exception("\nОшибка: файла не существует и файл нельзя создать. Дальнейшая работа с записью в файл не возможна." +

" Работа программы будет переведена в штатный режим. \n");

}

try(FileWriter fileWriter = new FileWriter(filePath, false)){

fileWriter.write(data);

}catch(Exception ex){

throw new Exception("\nОшибка: файла не существует. Дальнейшая работа с записью в файл не возможна." +

" Работа программы будет переведена в штатный режим. \n");

}

}

public static String outputDataFromFile(String filePath) throws Exception{

if((!checkFilePath(filePath)))

throw new Exception("\nОшибка: не корректный путь к файлу. Дальнейшая работа с чтением из файла не возможна." +

" Работа программы будет переведена в штатный режим. \n");

String data = "";

try(FileReader fileReader = new FileReader(filePath)){

int c = (-1);

while((c = fileReader.read()) != (-1))

data += (char)c;

}catch(Exception ex){

throw new Exception("\nОшибка: не удалось считать данные из файл. Дальнейшая работа с чтением из файла не возможна." +

" Работа программы будет переведена в штатный режим. \n");

}

return data;

}

}

**Класс MenuClass:**

package com.company;

import java.util.Scanner;

public class MenuClass {

public static boolean isNumber(String text){

if(text.length() == 0)

return false;

int countMinus = 0;

boolean flag = true;

for(int i = 0; (i < text.length()) && (flag); i++){

CharSequence s = Character.toString(text.charAt(i));

if((!("0123456789".contains((CharSequence)Character.toString(text.charAt(i)))))

&& (text.charAt(i) != '-'))

flag = false;

else if(text.charAt(i) == '-')

countMinus++;

}

if(!flag)

return false;

flag = (countMinus <= 1);

return flag;

}

public static int getCorrectValue(int min, int max){

Scanner sc = new Scanner(System.in);

String number = "";

System.out.print("Введите число: ");

number = sc.nextLine();

while((!isNumber(number)) || (!(((Integer.valueOf(number)) >= min) && ((Integer.valueOf(number)) <= max)))){

System.out.print("Введите корректное целое число из диапазона [ " + min + " ; " + max + " ]: ");

number = sc.nextLine();

}

int value = Integer.valueOf(number);

return value;

}

public void startMenu(){

do{

outputMenu();

System.out.print("\nПродолжить работу?(0 - нет, 1 - да)\n");

}while(getCorrectValue(0, 1) == 1);

}

public void outputMenu(){

System.out.println("Выберите задание, которое необходимо решить: ");

System.out.println("1. Определить, является ли строка подстрокой исходной строки.");

System.out.println("2. Найти длину указанного слова в предложении.");

System.out.println("3. Вывести центральное слово(слова) в предложении.");

System.out.println("4. Выход из программы");

int numberAnswer = getCorrectValue(1, 4);

if(numberAnswer == 4)

return;

System.out.println("\n\nВыберите способ считывания данных: ");

System.out.print("Требования к входным данным из файла: исходная строка и дополнытельная информация(подстрока / порядковый номер слова)\n");

System.out.print("должны располагаться в один столбец и разделяться символом перевода коретки(\\n или Enter)\n\n");

System.out.println("1. Считать входные данные из файла.");

System.out.println("2. Считать входные данные из консоли.");

int numberInput = getCorrectValue(1, 2);

System.out.println("\n\nВыберите способ вывода данных: ");

System.out.println("1. Вывести данные в файл");

System.out.println("2. Вывести данные в консоль");

int numberOutput = getCorrectValue(1, 2);

Scanner sc = new Scanner(System.in);

String text = "", answer = "";

StringClass obj = new StringClass();

switch(numberAnswer){

case 1:{

String substr = "";

if(numberInput == 1){

System.out.print("\nВведите путь к файлу, из которого нужно считать входные данные(исходную строку и подстроку разделённые \\n (Enter)): ");

String filePath = sc.nextLine();

try{

String[] str = FileIO.outputDataFromFile(filePath).split("\n");

text = str[0];

substr = str[1];

System.out.print("\nДанные успешно считаны из файла!\n");

}catch(Exception ex){

System.out.print(ex.getMessage());

System.out.print("\nВвод данных через консоль...\n\n");

System.out.print("\nВведите исходную строку: ");

text = sc.nextLine();

System.out.print("Введите подстроку: ");

substr = sc.nextLine();

}

}else{

System.out.print("\nВведите исходную строку: ");

text = sc.nextLine();

System.out.print("Введите подстроку: ");

substr = sc.nextLine();

}

obj.setText(text);

if(obj.substringInString(substr))

answer = "Подстрока \"" + substr + "\" присутствует в строке " + "\"" + text + "\"";

else

answer = "Подстрока \"" + substr + "\" не присутствует в строке " + "\"" + text + "\"";

if(numberOutput == 2){

System.out.print(answer);

break;

}

System.out.print("\nВведите путь к файлу, в который нужно вывести результат работы программы: ");

String filePath = sc.nextLine();

try{

FileIO.inputDataInFile(filePath, answer, true);

System.out.print("\nДанные успешно сохранены в файл!\n");

}catch (Exception ex){

System.out.print(ex.getMessage());

System.out.println("\nВывод результата на консоль: \n");

System.out.print(answer);

}

break;

}

case 2:{

int number = 0;

if(numberInput == 1){

System.out.print("\nВведите путь к файлу, из которого нужно считать входные данные(исходную строку и порядковый номер слова разделённые \\n (Enter)): ");

String filePath = sc.nextLine();

try{

String[] str = FileIO.outputDataFromFile(filePath).split("\n");

text = str[0];

number = Integer.valueOf(str[1]);

System.out.print("\nДанные успешно считаны из файла!\n");

}catch(NumberFormatException ex){

System.out.print("\nОшибка: порядковый номер слова считан не корректно. Данные в файле не соответствуют требованиям входных данных." +

" Работа программы будет переведена в штатный режим. \n");

System.out.print("\nВвод данных через консоль...\n\n");

System.out.print("\nВведите исходную строку: ");

text = sc.nextLine();

System.out.print("Введите порядковый номер в предложении длину которого необходимо найти\n");

number = getCorrectValue(1, text.trim().length());

}catch(Exception ex){

System.out.print(ex.getMessage());

System.out.print("\nВвод данных через консоль...\n\n");

System.out.print("\nВведите исходную строку: ");

text = sc.nextLine();

System.out.print("Введите порядковый номер в предложении длину которого необходимо найти\n");

number = getCorrectValue(1, text.trim().length());

}

}else{

System.out.print("\nВведите исходную строку: ");

text = sc.nextLine();

System.out.print("Введите порядковый номер в предложении длину которого необходимо найти\n");

number = getCorrectValue(1, text.trim().length());

}

obj.setText(text);

int lengthWord = obj.lengthDefineWord(number);

answer = "Слово с порядковым номером " + String.valueOf(number) + " имеет длину " + String.valueOf(lengthWord);

if(numberOutput == 2){

System.out.print(answer);

break;

}

System.out.print("\nВведите путь к файлу, в который нужно вывести результат работы программы: ");

String filePath = sc.nextLine();

try{

FileIO.inputDataInFile(filePath, answer, true);

System.out.print("\nДанные успешно сохранены в файл!\n");

}catch (Exception ex){

System.out.print(ex.getMessage());

System.out.println("\nВывод результата на консоль: \n");

System.out.print(answer);

}

break;

}

case 3:{

int number = 0;

if(numberInput == 1){

System.out.print("\nВведите путь к файлу, из которого нужно считать входные данные(исходную строку и порядковый номер слова разделённые \\n (Enter)): ");

String filePath = sc.nextLine();

try{

text = FileIO.outputDataFromFile(filePath);

System.out.print("\nДанные успешно считаны из файла!\n");

}catch(Exception ex){

System.out.print(ex.getMessage());

System.out.print("\nВвод данных через консоль...\n\n");

System.out.print("\nВведите исходную строку: ");

text = sc.nextLine();

}

}else{

System.out.print("\nВведите исходную строку: ");

text = sc.nextLine();

}

obj.setText(text);

answer = "Центральное слово/слова в предложении: " + obj.centerWords();

if(numberOutput == 2){

System.out.print(answer);

break;

}

System.out.print("\nВведите путь к файлу, в который нужно вывести результат работы программы: ");

String filePath = sc.nextLine();

try{

FileIO.inputDataInFile(filePath, answer, true);

System.out.print("\nДанные успешно сохранены в файл!\n");

}catch (Exception ex){

System.out.print(ex.getMessage());

System.out.println("\nВывод результата на консоль: \n");

System.out.print(answer);

}

break;

}

}

}

}

7 Заключение

Был применён на практике объектно-ориентированный подход для организации системы ввода/вывода, получен практический опыт описания и реализации классов, создания объектов, вызова методов, а также навыки работы с потоками данных и обработки исключений на базе платформы Java. Были созданы классы для пользовательского интерфейса и чтения/записи данных из файла / в файл. Обработка исключений корректна и при их обработке программа переходит в штатный режим, согласно заданию.

8 Список использованных источников

1. Java. Экспресс-курс [электронный ресурс] // Сайт Александра Климова [сайт], URL: <http://developer.alexanderklimov.ru/android/java/java.php> (дата обращения 12.09.20)

2. API Specification for the Java 7 SE. [официальный сайт] URL: <http://docs.oracle.com/javase/7/docs/api/> (дата обращения 12.09.20)

3. Гради Буч, Роберт А. Максимчук, Майкл У. Энгл, Бобби Дж. Янг, Джим Коналлен, Келли А. Хьюстон. Объектно-ориентированны анализ и проек-тирование с примерами приложений. Третье издание. М.: "Вильямс", 2010, -720 с.

4. Хабибуллин И.Ш. Java 7: для программистов / И. Ш. Хабибуллин. – Санкт-Петербург : БХВ–Петербург, 2014. – 768 с.

5. Васильев А. Н. Java. Объектно-ориентированное программирование: для магистров и бакалавров. Базовый курс по объектно-ориентированному программированию / А. Н. Васильев . – СПб.: Питер, 2012. – 395 с.