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 дисциплине  Объектно-ориентированные языки |
| по лабораторной работе №3 «Работа с потоками данных и обработка исключений» |
| наименование темы  Вариант №29 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент |  | АСУб-19-1 |  | C:\Users\Kinzel\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1.png |  | Янхаев Д. В. |
|  |  | 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](#_Toc59114392)

[2 Описание структуры классов 4](#_Toc59114393)

[3 Проектирование классов 6](#_Toc59114394)

[4 Описание методов классов и спецификация локальных переменных 8](#_Toc59114395)

[5 Таблица тестов 11](#_Toc59114396)

[6 Тесты программы 12](#_Toc59114397)

[7 Листинг программы 21](#_Toc59114398)

[Список использованных источников 30](#_Toc59114399)

1 Постановка задачи

1. Отсортируйте слова в массиве по третьей букве, если слово меньше трех букв, то по последней.
2. Определите, входит ли строка, указанная в командной строке, при запуске программы, в предложение.
3. Удалите в предложении каждое третье слово.

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. Работа с файлами может быть организована методами класса реализующего пользовательское меню, либо в рамках 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.

2 Описание структуры классов

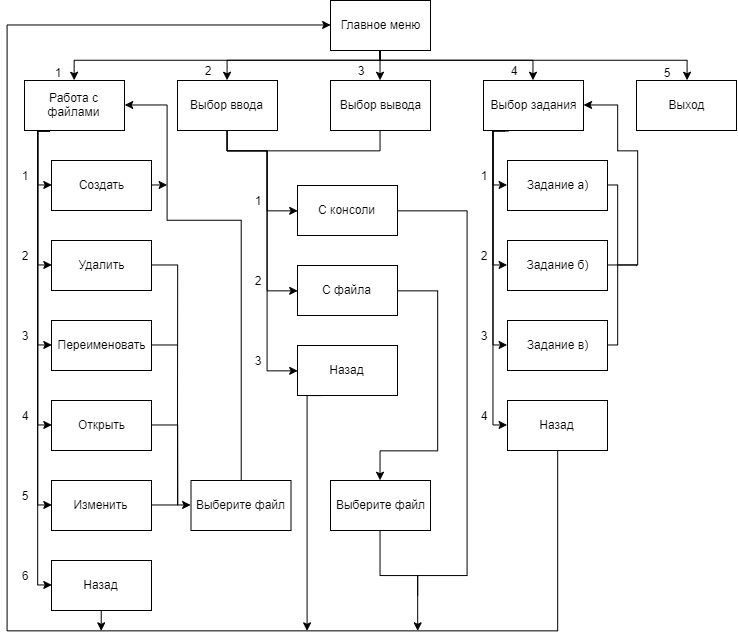


Рисунок 1 – Структура пользовательского меню

При запуске программы пользователь попадает в главное меню, которое состоит из пяти пунктов: «работа с файлами»(1), «выбор ввода»(2) «выбор вывода»(3) «выбор задания»(4) и «выход»(5). Сверху рамки расположен индикатор состояния ввода и вывода, работающий в реальном времени, по умолчанию ввод и вывод осуществляется с консоли.

При выборе «пункта работа с файлами»(1) появляются 6 вариантов: создать(1), удалить(2), переименовать(3) открыть(4) изменить(5) и назад(6). При создании файла(1) необходимо ввести имя файла. Для того, чтобы удалить(2), переименовать(3), открыть(4) или изменить(5) файл, его нужно выбрать из последующего выпадающего списка всех файлов формата .txt. После выполнения одного из пунктов (1)-(5), снова возвращается пунктам части «работа с файлами»(1). При выборе назад(6) возвращаемся в главное меню. Сверху рамки расположен индикатор состояния ввода и вывода, работающий в реальном времени, по умолчанию ввод и вывод осуществляется с консоли.

В пункте «выбор ввода»(2) и «выбор вывода»(3) пользователю предлагается 2 варианта ввода и вывода текста: с консоли(1) и с файла(2). При выборе с файла(2), выпадает список всех файлов формата .txt. При выборе назад(3) возвращаемся в главное меню.

При выборе пункта «выбор задания»(4), пользователю предлагается выбор из трех задач: задание а)(1), задание б)(2), задание в)(3), ввод и вывод которых зависит от строки состояния расположенный над рамкой данного меню. После выполнения одного из пунктов (1)-(3), снова возвращается пунктам части «выбор задания»(4). При выборе назад(4) возвращаемся в главное меню.

Для того, чтобы завершить работу программы, нужно при необходимости попасть в главное меню, затем выбрать пункт «выход»(5).

3 Проектирование классов

Решение задания состоит из четырех классов: Main,Menu,Arrays и Strings.

Класс Main реализует вызов метода mainMenu() класса Menu и содержит:

* Главный метод main(String[] args)

Класс Menu реализует меню программы, содержит: 18 методов(6 функции, 14 процедур) и 1 поле:

* Поле private static String[] modeIO – поле класса, содержащее значение режима ввода и вывода;
* Процедура public static void setModeIO(String mode, int num, int numOther) – изменяет одно из двух значений поля modeIO, вызывает функцию checkVoidFile();
* Функция public static String getModeIO(int num) – возвращает значение поля modeIO[num];
* Процедура public static void getModeIO() – возвращает состояние modeIO;
* Процедура public void mainMenu() – выводит главное меню, вызывает процедуры getModeIO(), fileMenu(), inputMode(), outputMode(); taskMenu();
* Процедура public static void frame(String menuString, String[] arrayMenuString, int count) – создает форму контекстного меню с названием menuString, с шириной в count, с вариантами выбора пунктов массива arrayMenuString;
* Функция public void checkVoidFile(String mode) – возвращает true если файл mode пуст иначе false;
* Функция public static String[] listFiles() – возвращает текстовый массив названий файлов имеющих расширение .txt;
* Функция public static int choiceFile() – создает список из массива файлов listFiles() с выбором варианта, возвращает номер выбранного элемента, вызывает функцию isDigit(String s);
* Функция private static boolean isDigit(String s) – возвращает true если текст s является целым, иначе false;
* Процедура public static void inputMode() – вызывает контекстное меню для выбора метода ввода: с консоли или с файла, вызывает процедуры getModeIO(), setModeIO(), frame(menuString, arrayMenuString, count), также при выборе метода ввода с файла вызывает функцию listFiles()[choiceFile()-1] который возвращает элемент строкового массива получаемого с помощью функции choiceFile();
* Процедура public static void outputMode()– вызывает контекстное меню для выбора метода вывода: с консоли или с файла, вызывает процедуры getModeIO(), setModeIO(), frame(menuString, arrayMenuString, count), также при выборе метода ввода с файла вызывает функцию listFiles()[choiceFile()-1] который возвращает элемент строкового массива получаемого с помощью функции choiceFile();
* Процедура public static void fileMenu() – выводит меню для работы с файлами, вызывает процедуры getModeIO(), frame(menuString, arrayMenuString, count), createFile(), removeFile(), renameFile(), openFile(), editFile();
* Процедура public static void taskMenu() – выводит меню заданий, вызывает процедуры getModeIO(), frame(menuString, arrayMenuString, count). Для выполнения заданий вызывает методы класса Arrays, Strings;
* Процедура public static void editFiles() – позволяет изменить файл, который выбирает пользователь, вызывается методом listFiles()[choiceFile()-1], пользователь вводит данные с консоли;
* Процедура public static void editFiles(String data) – выводит data в файл, который отображен в состоянии modeIO[1];
* Процедура public static String openFiles() – позволяет посмотреть содержимое файл, который выбирает пользователь, вызывается методом listFiles()[choiceFile()-1];
* Функция public static String openFiles(boolean flag) – позволяет вернуть содержимое из файла.
* Процедура public static void renameFiles() - позволяет изменить название файла, который выбирает пользователь, вызывается методом listFiles()[choiceFile()-1];
* Процедура public static void createFiles() – создает файл с названием вводимое пользователем с консоли, файл автоматически имеет расширение .txt;
* Процедура public static void removeFiles() - позволяет удалить файл, который выбирает пользователь, вызывается методом listFiles()[choiceFile()-1];

Класс Arrays – осуществляет работу с массивами в заданиях, содержит 4 метода (2 функции, 2 процедуры) и 1 поле:

* Поле arrayString – хранит массив, полученный из строк, который используется для решения заданий;
* Процедура public void setArrayString(String inputString) – создает массив из строки inputString и загружает в поле arrayString;
* Функция public String getArrayString() – возвращает строку из массива поля arrayString;
* Процедура public void compareArray(String[] arrayString, int num) – сравнивает элементы массива arrayString и производит сортировку по символу с номером num, для поиска символа вызывается метод класса String string.searchChar(String arrayString, int num);
* Функция public String deleteArray(int numElement) – создает строку без numElement элементов массива поля arrayString;

Класс Strings – осуществляет работу со строками в заданиях, который содержит 3 метода (2 функции, 1 процедура) и 1 поле:

* Поле private String inputString – хранит строку, которая используется в заданиях;
* Функция public String getInputString() – возвращает строку из поля inputString;
* Процедура public void setInputString() – загружает строку в поле inputString вводимую с консоли;
* Функция public static char searchChar(String elemArrayString, int n) – возвращает символ с индексом n-1 в строке elemArrayString;

4 Описание методов классов и спецификация локальных переменных

Таблица 1 – Описание методов класса mainMenu

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| setModeIO | Метод для установки значения поля modeIO | void | public static | String, int, int |
| getModeIO | Метод для получения поля modeIO[num] | String | public static | int |
| getModeIO | Метод для получения состояния поля modeIO | void | public static | - |
| mainMenu | Метод выводит главное меню | void | public | - |
| frame | Метод создает форму для меню | void | public static | String, String[], int |
| listFiles | Метод получает список файлов с расширением .txt | String[] | public static | - |
| choiceFile | создает список из массива файлов listFiles() с выбором варианта | int | public static | - |
| inputMode | вызывает контекстное меню для выбора метода ввода: с консоли или с файла | void | public static | - |
| outputMode | вызывает контекстное меню для выбора метода вывода | void | public static | - |
| fileMenu | выводит меню для работы с файлами | void | public static | - |
| taskMenu | выводит меню заданий | void | public static | - |
| editFiles | позволяет изменить файл, который выбирает пользователь | void | public static | - |
| editFiles | выводит data в файл, который отображен в состоянии modeIO[1] | void | public static | String |
| openFiles | позволяет посмотреть содержимое файл, который выбирает пользователь | void | public static | - |
| openFiles | позволяет вернуть содержимое из файла. | String | public static | boolean |
| renameFiles | позволяет изменить название файла, который выбирает пользователь | void | public static | - |
| createFiles | создает файл с названием вводимое пользователем с консоли | void | public static | - |
| removeFiles | позволяет удалить файл, который выбирает пользователь | void | public static | - |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| checkVoidFile | Метод для проверки пустоты файла | boolean | private static | String |
| isDigit | Метод для получения поля modeIO[num] | boolean | private static | String |

Таблица 2 – Описание методов класса Strings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| getImputString | Метод для получения поля inputString | String | public | - |
| setImputString | Метод для установки значения inputString | void | public | String |
| searchChar | Метод позволяющий найти букву по элементу в строке, если такой буквы нет возвращает последнюю | char | public static | String |

Таблица 3 – Описание методов класса Arrays

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| Arrays | Конструктор по умолчанию. Устанавливает поля объекта значениями по умолчанию | - | public | - |
| getArrayString | Метод, возвращающий строку, собранный из массива строк поля arrayString | String | public | - |
| setArrayString | Метод для загрузки массива из строк в поле arrayString | void | public | String |
| compareArray | Метод осуществляющий сортировку методом пузырька по номеру элементу num массива из строк поля arrayString | void | public | - |
| deleteArray | Метод удаляющий из массива строк поля arrayString строку с номером numElement | String | public | - |

Таблица 4 – Описание полей класса Strings

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Модификаторы |
| inputString | Хранит строку над которой будут производиться различные действия | String | private |

Таблица 5 – Описание полей класса Arrays

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Модификаторы |
| arrayString | Хранит массив строк над которой будут производиться различные действия | String[] | private |

Таблица 6 – Описание полей класса mainMenu

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Модификаторы |
| modeIO | Хранит значение режима ввода и вывода | String[] | private static |

Таблица 7 – Спецификация локальных переменных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | ОДЗ |
| Процедура setModeIO(String mode, int num, int numOther) | | | | |
| 1 | mode | Строка для записи в поле modeIO | String | [A;z, 0;9] |
| 2 | num | Индекс для состояния ввода/вывода modeIO | Int | [0; 1] |
| 3 | numOther | Индекс сравниваемого символа | Int | [0; 1] |
| Функция getModeIO(int num) | | | | |
| 4 | num | Индекс состояния ввода/вывода modeIO | Int | [0; 2\*109] |
| процедура mainMenu() | | | | |
| 5 | count | Длина рамки меню | Int | [0; 2\*109] |
| 6 | menuString | Название рамки меню | String | [A;z, 0;9] |
| 7 | ArraymenuString | Элементы меню | String | [A;z, 0;9] |
| 8 | num | Переменная для выбора элемента меню | Int | [0; 2\*109] |
| процедура frame(String menuString, String[] arrayMenuString, int count) | | | | |
| 10 | count | Длина рамки меню | Int | [0; 2\*109] |
| 11 | menuString | Название рамки меню | String | [A;z, 0;9] |
| 12 | ArraymenuString | Элементы меню | String | [A;z, 0;9] |
| Функция listFiles() | | | | |
| 13 | name | Имя файла | String | [A;z, 0;9] |
| 14 | filesNames | Массив из имен файлов | String | [A;z, 0;9] |
| процедура inputMode() | | | | |
| 15 | count | Длина рамки меню | Int | [0; 2\*109] |
| 16 | menuString | Название рамки меню | String | [A;z, 0;9] |
| 17 | ArraymenuString | Элементы меню | String | [A;z, 0;9] |
| процедура outputMode() | | | | |
| 18 | count | Длина рамки меню | Int | [0; 2\*109] |
| 19 | menuString | Название рамки меню | String | [A;z, 0;9] |
| 20 | ArraymenuString | Элементы меню | String | [A;z, 0;9] |
| процедура fileMenu() | | | | |
| 21 | count | Длина рамки меню | Int | [0; 2\*109] |
| 22 | menuString | Название рамки меню | String | [A;z, 0;9] |
| 23 | ArraymenuString | Элементы меню | String | [A;z, 0;9] |
| процедура taskMenu() | | | | |
| 24 | count | Длина рамки меню | Int | [0; 2\*109] |

Продолжение таблицы 7

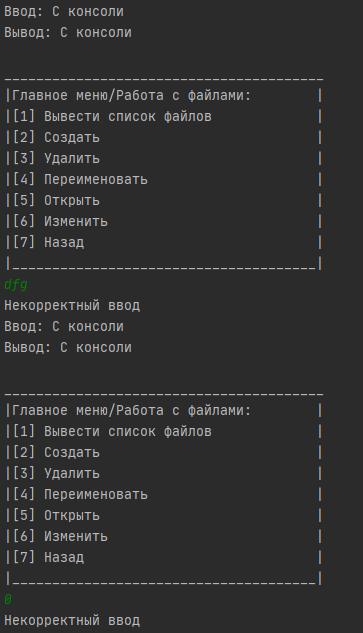
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | ОДЗ |
| 25 | menuString | Название рамки меню | String | [A;z, 0;9] |
| 26 | ArraymenuString | Элементы меню | String | [A;z, 0;9] |
| процедура openFiles() | | | | |
| 27 | c | Символ преобразованный в int в файле | Int | [0; 2\*109] |
| Функция openFiles() | | | | |
| 28 | c | Символ преобразованный в int в файле | Int | [0; 2\*109] |
| 29 | sb | Строка в которой хранится все содержимое файла | String | [A;z, 0;9] |

5 Таблица тестов

Таблица 8 – Таблица тестов

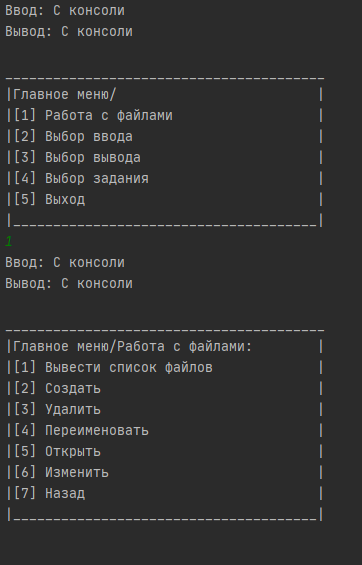
|  |  |  |
| --- | --- | --- |
| № | Описание теста | Результат работы программы |
| 1 | Ввод в меню числа, не соответствующего пунктам меню | Сообщение об ошибке ввода, «Некорректный ввод» |
| 2 | Ввод в меню некорректных символов | Сообщение об ошибке ввода, «Некорректный ввод» |
| 3 | Ввод в главном меню «1» | Переход в меню «работа с файлами» |
| 4 | Ввод в главном меню «2» | Переход в меню для выбора способа ввода текста |
| 5 | Ввод в главном меню «3» | Переход в меню для выбора вывода текста |
| 6 | Ввод в главном меню «4» | Переход в меню для выбора задания |
| 7 | Ввод в главном меню «5» | Выход их программы |
| 8 | Ввод в меню работы с файлами «2»(создать) и корректного имени файла | Предлагает ввести имя файла, создает файл, выдает сообщение об успешном создании файла, переходит снова в меню работы с файлами |
| 9 | Ввод в меню работы с файлами «2»(создать) существующего имени файла | Предлагает ввести имя файла, выдает сообщение об ошибке, переходит снова в меню работы с файлами |
| 10 | Ввод в меню работы с файлами «3»(удалить) | Удаляет файл, выводит сообщение об успешном удалении файла, переходит в меню работы с файлами |
| 11 | Ввод в меню работы с файлами «3»(удалить) и неверный файл | Выводит сообщение об ошибке и просит снова выбрать удаляемый файл |
| 12 | Ввод в меню работы с файлами «4»(Переименовать) | Предлагает ввести новое имя файла, сообщение о том, что файл переименован, переход снова в меню работы с файлами |
| 13 | Ввод в меню работы с файлами «4»(Переименовать) и неверный файл | Выводит сообщение об ошибке и переход снова в меню работы с файлами |
| 14 | Ввод в меню работы с файлами «5»(открыть) | Показывает содержимое файла, переходит снова в меню работы с файлами |
| 15 | Ввод в меню работы с файлами «5»(открыть) и неверный номер файла | Выводит сообщение об ошибке, переходит снова в меню работы с файлами |
| 16 | Ввод в меню работы с файлами «6»(изменить) | Предлагает ввести новое содержимое файла, переходит снова в меню работы с файлами |
| 17 | Ввод в меню работы с файлами «6»(изменить) и неверный номер файла | Выводит сообщение об ошибке, переходит снова в меню работы с файлами |
| 18 | Ввод в меню работы с файлами «6»(назад) | Переход в главное меню |
| 19 | Ввод в меню выбора способа ввода текста «1»(С файла) пустой файл | Сообщение об ошибке, переход в главное меню |
| 20 | Ввод в меню выбора способа ввода текста «2»(из файла) | Переход в меню выполнения заданий |
| 21 | Ввод в меню выбора способа ввода текста «2»(из файла) и неверный номер файл | Сообщение об ошибке, переход в меню выбора способа ввода |
| 22 | Ввод в меню выбора способа ввода или вывода текста «2,3»(из файла) файл уже записанный на вывод или ввод. | Сообщение об ошибке, переход в меню выбора способа ввода |
| 22 | Ввод в меню выбора способа ввода текста «3»(назад) | Переход в главное меню |
| 23 | Ввод в меню выполнения заданий «4»(назад) | Переход в меню выбора способа ввода текста |
| 24 | Ввод в меню выбора способа ввода текста «3»(в главное меню) | Переход в главное меню |

6 Тесты программы

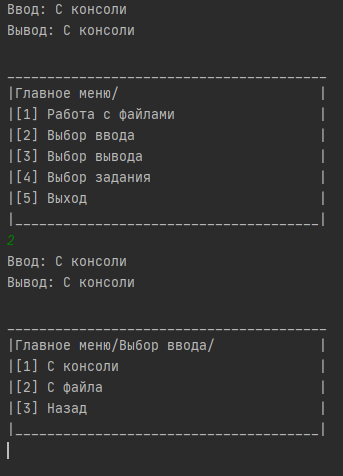


Тесты 1-2

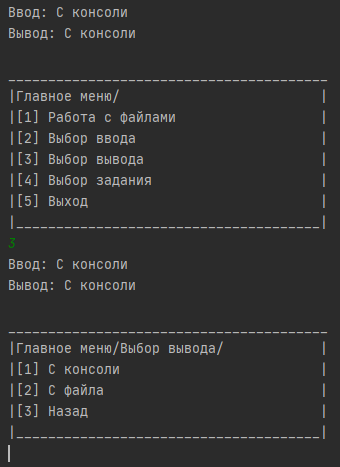
Тест 1

****

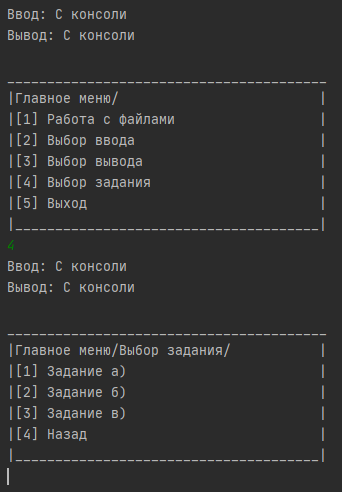
Тест 3



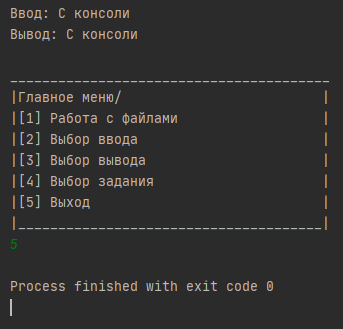
Тест 4



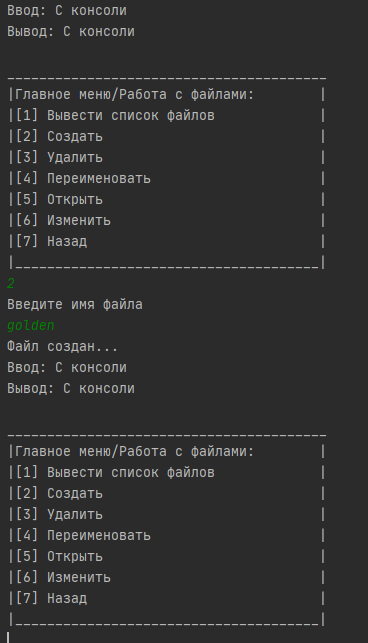
Тест 5



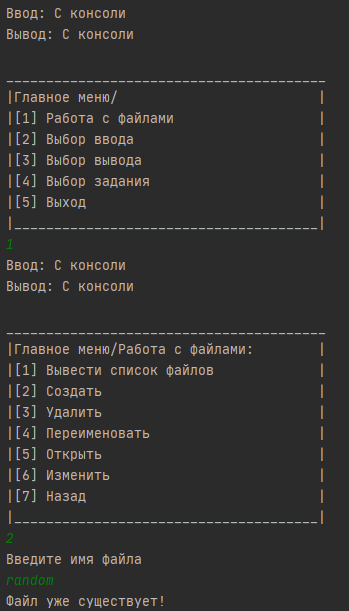
Тест 6



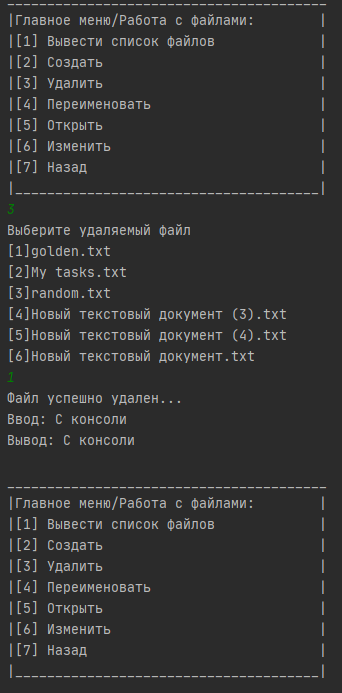
Тест 7



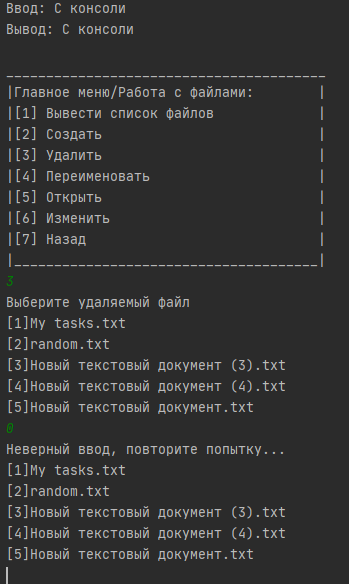
Тест 8



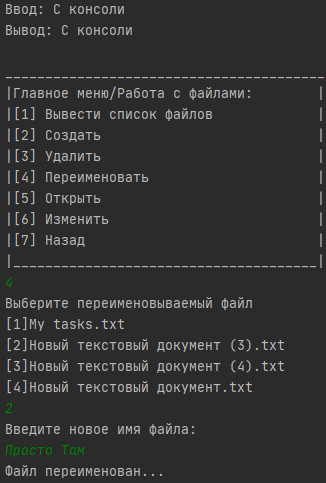
Тест 9



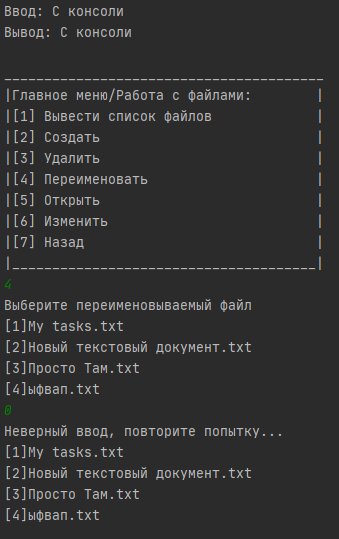
Тест 10



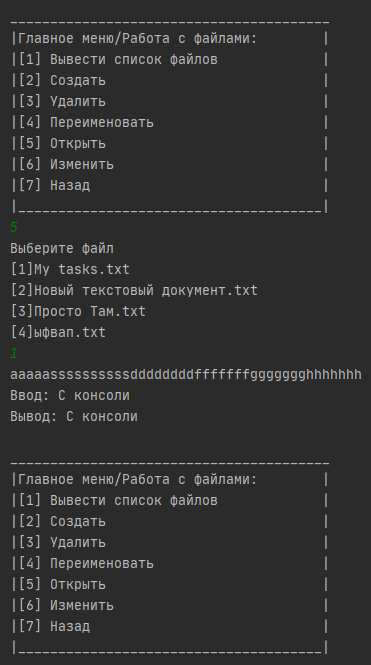
Тест 11



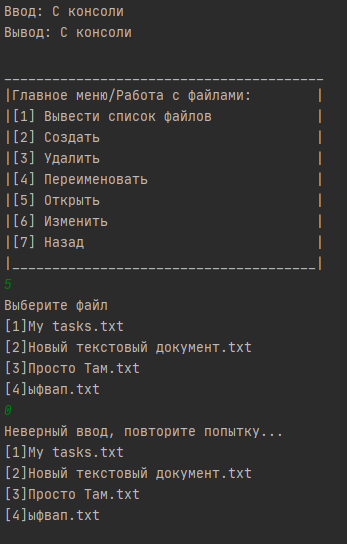
Тест 12



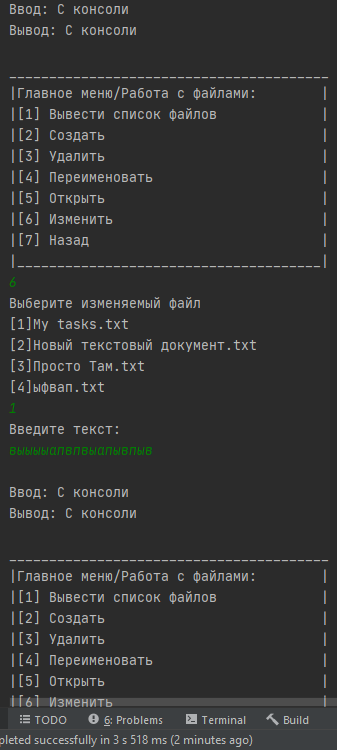
Тест 13



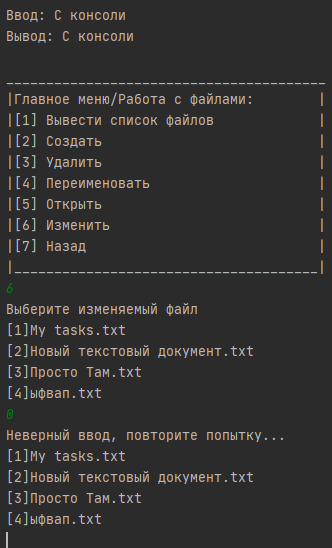
Тест 14



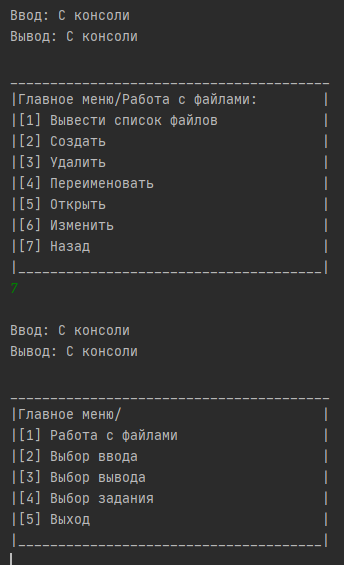
Тест 15



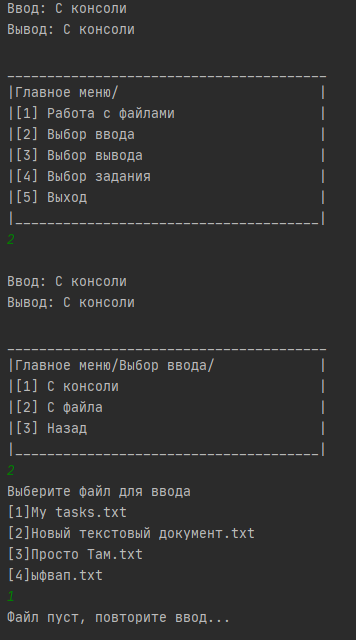
Тест 16



Тест 17



Тест 18



Тест 19

7 Листинг программы

Класс Main

package lab3;  
  
import java.io.IOException;  
  
public class Main {  
  
 public static void main(String[] args) throws IOException, ClassNotFoundException {  
 Menu menu = new Menu();  
 menu.mainMenu();  
 }  
  
}

Класс Menu

package lab3;  
import java.io.\*;  
import java.util.Scanner;  
  
public class Menu {  
 private static String[] *modeIO* = {"С консоли","С консоли"};  
 public static void setModeIO(String mode, int num, int numOther){  
 if(*modeIO*[numOther].equals(mode) && !(mode.equals("С консоли"))){  
 System.*out*.println("Файл ввода совпадает с файлом вывода, повторите попытку...");  
 }  
 else {  
 if(num==0 && !(mode.equals("С консоли"))) {  
 if (*checkVoidFile*(mode)) System.*out*.println("Файл пуст, повторите ввод...");  
 else *modeIO*[num] = mode;  
 }  
 else *modeIO*[num] = mode;  
 }  
 }  
 public static String getModeIO(int num){return *modeIO*[num];}  
 public static void getModeIO(){  
 System.*out*.println("Ввод: " + *modeIO*[0] + "\nВывод: " + *modeIO*[1]);  
 }  
  
 private static boolean checkVoidFile(String mode){  
 File file = new File("\"D:/Documents/ИРНИТУ/АСУб-19-1/2 курс/ООП/3/Files/\"" + mode);  
 if(file.length() == 0) return true;  
 else return false;  
 }  
  
 public void mainMenu() throws IOException, ClassNotFoundException {  
 while(true){  
 Scanner in = new Scanner(System.*in*);  
 *getModeIO*();  
 int count = 40;  
 String menuString = "|Главное меню/";  
 String[] arrayMenuString = "Работа с файлами-Выбор ввода-Выбор вывода-Выбор задания-Выход".split("-");  
 *frame*(menuString,arrayMenuString,count);  
  
 int num = 0;  
 try {  
 num = Integer.*parseInt*(in.nextLine());  
 if (num < 1 || num >= 6) throw new IllegalArgumentException();  
 else {  
 switch (num) {  
 case 1:  
 *fileMenu*();  
 break;  
 case 2:  
 *inputMode*();  
 break;  
 case 3:  
 *outputMode*();  
 break;  
 case 4:  
 *taskMenu*();  
 break;  
 case 5:  
 break;  
 }  
 if(num==5) break;  
 }  
 }  
 catch (IllegalArgumentException e) {  
 System.*out*.println("Некорректный ввод");  
 }  
  
 }  
 }  
 public static void inputMode(){  
 while(true){  
 Scanner in = new Scanner(System.*in*);  
 *getModeIO*();  
 int count = 40;  
 String menuString = "|Главное меню/Выбор ввода/";  
 String[] arrayMenuString = "С консоли-С файла-Назад".split("-");  
 *frame*(menuString,arrayMenuString,count);  
  
 int num = 0;  
 try {  
 num = Integer.*parseInt*(in.nextLine());  
 if (num < 1 || num >= 4) throw new IllegalArgumentException();  
 else {  
 switch (num) {  
 case 1:  
 *setModeIO*("С консоли",0, 1);  
 break;  
 case 2:  
 System.*out*.println("Выберите файл для ввода");  
 *setModeIO*(*listFiles*()[*choiceFile*()-1],0, 1);  
  
 break;  
 case 3:  
 break;  
 }  
 if(num==3||num==2||num==1) break;  
 }  
 }  
 catch (IllegalArgumentException e) {  
 System.*out*.println("Некорректный ввод");  
 }  
  
 }  
 }  
  
 public static void outputMode(){  
 while(true){  
 Scanner in = new Scanner(System.*in*);  
 *getModeIO*();  
 int count = 40;  
 String menuString = "|Главное меню/Выбор вывода/";  
 String[] arrayMenuString = "С консоли-С файла-Назад".split("-");  
 *frame*(menuString,arrayMenuString,count);  
  
 int num = 0;  
 try {  
 num = Integer.*parseInt*(in.nextLine());  
 if (num < 1 || num >= 4) throw new IllegalArgumentException();  
 else {  
 switch (num) {  
 case 1:  
 *setModeIO*("С консоли",1,0);  
 break;  
 case 2:  
 System.*out*.println("Выберите файл для вывода");  
 *setModeIO*(*listFiles*()[*choiceFile*()-1],1,0);  
 break;  
 case 3:  
 break;  
 }  
 if(num==3||num==2||num==1) break;  
 }  
 }  
 catch (IllegalArgumentException e) {  
 System.*out*.println("Некорректный ввод");  
 }  
  
 }  
 }  
  
 public static void taskMenu() throws IOException, ClassNotFoundException {  
 *getModeIO*();  
 Arrays arrays = new Arrays();  
 Strings strings = new Strings();  
 Scanner in = new Scanner(System.*in*);  
 while (true) {  
 int count = 40;  
 String menuString = "|Главное меню/Выбор задания/";  
 String[] arrayMenuString = "Задание а)-Задание б)-Задание в)-Назад".split("-");  
 *frame*(menuString, arrayMenuString, count);  
  
 int num = 0;  
 try {  
 num = Integer.*parseInt*(in.nextLine());  
 if (num < 1 || num >= 5) throw new IllegalArgumentException();  
 else {  
 if (*modeIO*[0].equals("С консоли")) System.*out*.println("Введите предложение: ");  
 strings.setInputString();  
 }  
 switch (num) {  
 case 1:  
 arrays.setArrayString(strings.getInputString());  
 arrays.compareArray(arrays.getArrayString().split(" "), 3);  
 if (*modeIO*[1].equals("С консоли")) System.*out*.println(arrays.getArrayString());  
 else *editFiles*(arrays.getArrayString());  
 break;  
  
 case 2:  
 if (*modeIO*[1].equals("С консоли")) System.*out*.println("Введите предложение: ");  
 String inputString = in.nextLine();  
 if (strings.getInputString().contains(inputString))  
 if (*modeIO*[1].equals("С консоли")) System.*out*.println("Найдено");  
 else *editFiles*("Найдено");  
 else if (*modeIO*[1].equals("С консоли")) System.*out*.println("Не найдено");  
 else *editFiles*("Не найдено");  
 break;  
  
 case 3:  
 arrays.setArrayString(strings.getInputString());  
 if (*modeIO*[1].equals("С консоли")) System.*out*.println(arrays.deleteArray(3));  
 else *editFiles*(arrays.deleteArray(3));  
 break;  
 case 4:  
 break;  
 }  
 if (num == 4) break;  
 } catch (IllegalArgumentException e) {  
 System.*out*.println("Некорректный ввод");  
 }  
  
 }  
 }  
 public static void fileMenu() {  
 Scanner in = new Scanner(System.*in*);  
 while(true){  
 *getModeIO*();  
 int count = 40;  
 String menuString = "|Главное меню/Работа с файлами:";  
 String[] arrayMenuString = "Вывести список файлов-Создать-Удалить-Переименовать-Открыть-Изменить-Назад".split("-");  
 *frame*(menuString,arrayMenuString,count);  
  
 int num = 0;  
 try {  
 num = Integer.*parseInt*(in.nextLine());  
 if (num < 1 || num >= 8) throw new IllegalArgumentException();  
 else {  
 switch (num) {  
 case 1:  
 for(int i=0; i<*listFiles*().length; i++) {  
 System.*out*.println(*listFiles*()[i]);  
 }  
 break;  
 case 2:  
 *createFiles*();  
 break;  
 case 3:  
 if(*listFiles*().length!=0)  
 *removeFiles*();  
 else System.*out*.println("Файлов не существуют, создайте файл");  
 break;  
 case 4:  
 if(*listFiles*().length!=0)  
 *renameFiles*();  
 else System.*out*.println("Файлов не существуют, создайте файл");  
 break;  
 case 5:  
 if(*listFiles*().length!=0)  
 *openFiles*();  
 else System.*out*.println("Файлов не существуют, создайте файл");  
 break;  
 case 6:  
 if(*listFiles*().length!=0)  
 *editFiles*();  
 else System.*out*.println("Файлов не существуют, создайте файл");  
 break;  
 case 7:  
 break;  
 }  
 if(num==7) break;  
 }  
 }  
 catch (IllegalArgumentException e) {  
 System.*out*.println("Некорректный ввод");  
 }  
 }  
 }  
  
 private static boolean isDigit(String s) throws NumberFormatException {  
 try {  
 Integer.*parseInt*(s);  
 return true;  
 } catch (NumberFormatException e) {  
 return false;  
 }  
 }  
  
 public static int choiceFile(){  
 Scanner in = new Scanner(System.*in*);  
 int indexInput;  
 String indexInputString = new String();  
 do {  
 for (int i = 0; i < *listFiles*().length; i++) {  
 System.*out*.println("[" + (i + 1) + "]" + *listFiles*()[i]);  
 }  
 indexInputString = in.nextLine();  
   
 if (*isDigit*(indexInputString))  
 if(Integer.*parseInt*(indexInputString)>0 && Integer.*parseInt*(indexInputString)<*listFiles*().length)  
 indexInput = Integer.*parseInt*(indexInputString);  
 else {  
 System.*out*.println("Неверный ввод, повторите попытку...");  
 indexInput = 0;  
 }  
 else {  
 System.*out*.println("Неверный ввод, повторите попытку...");  
 indexInput = 0;  
 }  
 }while (indexInput>*listFiles*().length || indexInput<=0);  
 return indexInput;  
 }  
  
  
 public static void editFiles(){  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Выберите изменяемый файл");  
  
  
 try(FileWriter writer = new FileWriter("D:/Documents/ИРНИТУ/АСУб-19-1/2 курс/ООП/3/Files/" + *listFiles*()[*choiceFile*()-1], false))  
 {  
 System.*out*.println("Введите текст:");  
 writer.write(in.nextLine());  
 writer.flush();  
 }  
 catch(IOException ex){  
 System.*out*.println(ex.getMessage());  
 }  
 }  
  
 public static void editFiles(String data){  
 try(FileWriter writer = new FileWriter("D:/Documents/ИРНИТУ/АСУб-19-1/2 курс/ООП/3/Files/" + *modeIO*[1], true))  
 {  
 writer.write(data + "\r\n");  
 writer.flush();  
 }  
 catch(IOException ex){  
 System.*out*.println(ex.getMessage());  
 }  
 }  
  
 public static void openFiles(){  
 System.*out*.println("Выберите файл");  
 try(FileReader reader = new FileReader("D:/Documents/ИРНИТУ/АСУб-19-1/2 курс/ООП/3/Files/" + *listFiles*()[*choiceFile*()-1]))  
 {  
 int c;  
 while((c=reader.read())!=-1){  
 System.*out*.print((char)c);  
 }  
 }  
 catch(IOException ex){  
 System.*out*.println(ex.getMessage());  
 }  
 }  
  
 public static String openFiles(boolean flag){  
 try(FileReader reader = new FileReader("D:/Documents/ИРНИТУ/АСУб-19-1/2 курс/ООП/3/Files/" + *modeIO*[0]))  
 {  
 int c;  
 StringBuilder sb = new StringBuilder();  
 while((c=reader.read())!=-1){  
 sb.append((char)c);  
 }  
 return sb.toString();  
 }  
 catch(IOException ex){  
 System.*out*.println(ex.getMessage());  
 return null;  
 }  
 }  
  
 public static void renameFiles(){  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Выберите переименовываемый файл");  
 File file = new File("D:/Documents/ИРНИТУ/АСУб-19-1/2 курс/ООП/3/Files/" + *listFiles*()[*choiceFile*()-1]);  
 System.*out*.println("Введите новое имя файла:");  
 File file1 = new File("D:/Documents/ИРНИТУ/АСУб-19-1/2 курс/ООП/3/Files/" + in.nextLine() + ".txt");  
 try {  
 if(file.renameTo(file1))  
 {  
 System.*out*.println("Файл переименован...");  
 }  
 } catch (Exception e) {  
 System.*err*.println(e.getMessage());  
 }  
 }  
  
 public static void createFiles(){  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("Введите имя файла");  
 String name = in.nextLine() + ".txt";  
 File newFile = new File("D:/Documents/ИРНИТУ/АСУб-19-1/2 курс/ООП/3/Files/" + name);  
 try {  
 if (!newFile.exists()) {  
 boolean created = newFile.createNewFile();  
 if (created) System.*out*.println("Файл создан...");  
 }  
 }  
 catch (IOException ex){  
 System.*err*.println(ex.getMessage());  
 }  
 }  
  
 public static void removeFiles(){  
 System.*out*.println("Выберите удаляемый файл");  
 File file = new File("D:/Documents/ИРНИТУ/АСУб-19-1/2 курс/ООП/3/Files/" + *listFiles*()[*choiceFile*()-1]);  
 try {  
 if(file.delete())  
 {  
 System.*out*.println("Файл успешно удален...");  
 }  
 } catch (Exception ex) {  
 System.*err*.println(ex.getMessage());  
 }  
 }  
  
 public static String[] listFiles(){  
 try {  
 File f = new File("D:/Documents/ИРНИТУ/АСУб-19-1/2 курс/ООП/3/Files");  
 FilenameFilter filter = new FilenameFilter() {  
 @Override  
 public boolean accept(File f, String name) {  
 return name.endsWith(".txt");  
 }  
 };  
  
 File[] files = f.listFiles(filter);  
 String[] filesNames = new String[files.length];  
  
 for (int i = 0; i < files.length; i++) {  
 filesNames[i] = files[i].getName();  
 }  
 return filesNames;  
 } catch (Exception e) {  
 System.*err*.println(e.getMessage());  
 return null;  
 }  
 }  
  
  
  
 public static void frame(String menuString, String[] arrayMenuString, int count){  
 System.*out*.println();  
 System.*out*.println("\_".repeat(count));  
 System.*out*.println(menuString = menuString.length()>=count ? menuString.substring(0,count-1)  
 + "|" : menuString + " ".repeat(count-1-menuString.length()) + "|");  
 for(int j=1; j<=arrayMenuString.length; j++) {  
 menuString = ("|" + "[" + j + "] " + arrayMenuString[j-1]);  
 System.*out*.println(menuString = menuString.length()>=count ? menuString.substring(0,count-1)  
 + "|" : menuString + " ".repeat(count-1-menuString.length()) + "|");  
 }  
 System.*out*.println("|" + "\_".repeat(count-2) + "|");  
 }  
}

Класс Arrays

package lab3;  
  
public class Arrays{  
  
 private String[] arrayString;  
  
  
 public String getArrayString(){  
 StringBuilder builder = new StringBuilder();  
 for (String elemArrayString : arrayString) {  
 builder.append(elemArrayString).append(' ');  
 }  
 return builder.toString();  
 }  
  
 public void setArrayString(String inputString){  
 String[] array;  
 array = inputString.split(" ");  
 arrayString = array;  
 }  
  
  
  
 public void compareArray(String[] arrayString, int num){  
 num = num>0 ? num : 1;  
 for(int i=0; i<arrayString.length-1; i++){  
 for(int j=i+1;j<arrayString.length;j++) {  
  
 char a = Strings.*searchChar*(arrayString[i], num);  
  
 char b = Strings.*searchChar*(arrayString[j], num);  
 if (a > b) {  
 String temp = arrayString[i];  
 arrayString[i] = arrayString[j];  
 arrayString[j] = temp;  
 }  
 }  
 }  
 this.arrayString = arrayString;  
 }  
  
 public String deleteArray(int numElement){  
 numElement = numElement>0 ? numElement : arrayString.length+1;  
 StringBuilder builder = new StringBuilder();  
 for (int i=0; i< arrayString.length; i++){  
 if ((i+1)%numElement!=0){  
 builder.append(arrayString[i]).append(" ");  
 }  
 }  
 return builder.toString();  
 }  
}

Класс String

package lab3;  
  
import java.util.Scanner;  
public class Strings {  
  
 private String inputString;  
  
 public String getInputString(){  
 return inputString;  
 }  
  
 public void setInputString(){  
 Scanner in = new Scanner(System.*in*);  
 Menu menu = new Menu();  
 if(menu.*getModeIO*(0).equals("С консоли")) inputString = in.nextLine();  
 else inputString = Menu.*openFiles*(true);  
 }  
  
 public static char searchChar(String elemArrayString, int n){  
 return elemArrayString.length() >= n ?  
 elemArrayString.charAt(n-1) :  
 elemArrayString.charAt(elemArrayString.length() - 1);  
 }  
}

Список использованных источников

1. API Specification for the Java 7 SE. [официальный сайт]

URL: <http://docs.oracle.com/javase/7/docs/api/> (дата обращения:08.09.2020)

1. The Java Tutorials. SE [электронный ресурс],

URL:<http://docs.oracle.com/javase/tutorial/index.html>

(дата обращения:08.09.2020)