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 лабораторной работе №2 по дисциплине «Разработка классов и их использование» |
| 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](#_Toc53048689)

[2 Описание структуры классов 4](#_Toc53048690)

[3 Описание методов классов и спецификация локальных переменных 5](#_Toc53048691)

[4 Таблица тестов 7](#_Toc53048692)

[5 Результаты тестирования 8](#_Toc53048693)

[6 Исходный код 9](#_Toc53048694)

[Список использованных источников 13](#_Toc53048695)

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 не обходимых 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.

2 Описание структуры классов

Программа разбита на три класса. Один главный класс, в котором содержится метод main являющийся точкой входа в программу, класс Strings в котором содержатся необходимые методы реализующие функционал для обработки строки, и класс Arrays содержащий функционал для обработки массива из строк

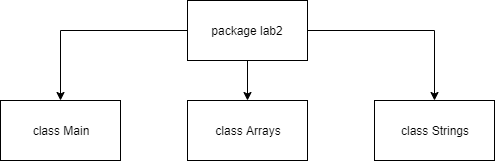


Рисунок 1 – Структура классов

3 Описание методов классов и спецификация локальных переменных

Таблица 1 – Описание методов класса Main

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| main | Точка входа в программу. Создает объекты класса задач и выполняет их методы run | - | Public static | - |

Таблица 2 – Описание методов класса Strings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| Strings | Конструктор по умолчанию. Устанавливает поля объекта значениями по умолчанию | - | public | - |
| Strings | Перегрузка конструктора, принимает аргументы для заполнения полей объекта переданными значениями | - | public | String |
| Strings | Конструктор копирования. Создает копию объекта переданного в аргументе | - | public | Strings |
| 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 |

Продолжение таблицы 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Назначение | Возвращаемый тип | Модификаторы | Аргументы |
| 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 | - |
| copyOfSerialization | Метод сериализации возвращающий объект того типа что и класс | Arrays | public static | Arrays |

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 – Спецификация локальных переменных

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Тип | Диапазон |
| n | Хранит номер выбранного пункта меню | Int | -2^32 … 2^32 - 1 |
| inputString | Хранит вводимую строку в задании 2 | String | - |
| array | Хранит массив из строк из строки inputString | String[] | - |
| a | Хранит сравниваемую букву из строки массива arrayString | char | A .. Z, a .. z |
| b | Хранит сравниваемую букву из следующей строки массива arrayString | char | A .. Z, a .. z |
| numElement | Хранит номер удаляемого элемента в массиве из строк | int | -2^32 … 2^32 - 1 |
| num | Хранит номер сравниваемого элемента | int | -2^32 … 2^32 - 1 |

4 Таблица тестов

Таблица 7 - Тесты

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Задание | Вход | Выход | Примечания |
| 1 | Проверка копирование объекта | - | Соммы грамм и нету грамм | Создаем объект arrays1 и клон arrays2 с помощью конструктора клонирования.  Меняем значение у arrays1.  Объекты должны получиться независимыми друг от друга |
| Каждый принадлежит всем |
| 2 | Проверка копирования с помощью сериализации | СТАРШИЙ БРАТ СЛЕДИТ ЗА ТОБОЙ | СТАРШИЙ БРАТ СЛЕДИТ ЗА ТОБОЙ | Создаем strings1 и также его клон с помощью конструктора сериализации, определенного в классе Arrays |
| Состояние в норме… |
| 3 | Проверка конструктора по умолчанию | СТАРШИЙ БРАТ СМОТРИТ НА ТЕБЯ | рааааал | Создаем объекты strings3 strings4 с помощью конструктора по умолчанию.  Вызываем метод у strings4.  Объекты должны получиться независимыми друг от друга |
| СТАРШИЙ БРАТ СМОТРИТ НА ТЕБЯ |

5 Результаты тестирования

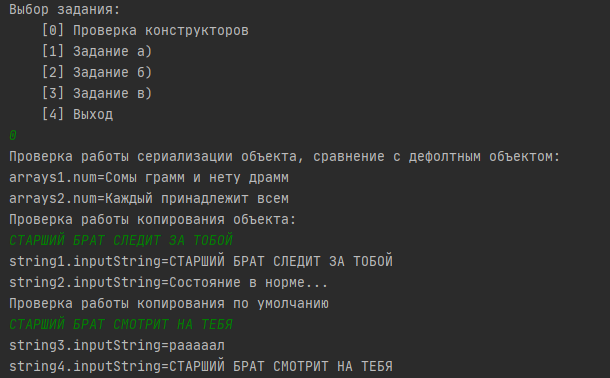


Рисунок 4 – Результат теста 1-3

6 Исходный код

1. Класс Main

package lab2;  
import java.io.IOException;  
import java.util.Scanner;  
public class Main {  
  
 public static void main(String[] args) throws IOException, ClassNotFoundException {  
  
 Scanner in = new Scanner(System.*in*);  
 Arrays arrays = new Arrays();  
 Strings strings = new Strings();  
  
 while (true) {  
 System.*out*.println("Выбор задания:\n \t[0] Проверка конструкторов\n \t[1] Задание а)\n \t[2] Задание б)\n \t[3] Задание в)\n \t[4] Выход");  
 int n = Integer.*parseInt*(in.nextLine());  
  
 if (n < 4 && n > 0) {  
 System.*out*.println("Введите предложение: ");  
 strings.setInputString();  
 }  
 switch (n){  
 case 0:  
 System.*out*.println("Проверка работы сериализации объекта, сравнение с дефолтным объектом:");  
 Arrays arrays1 = new Arrays();  
 Arrays arrays2 = Arrays.*copyOfSerialization*(arrays1);  
 arrays1.setArrayString("Сомы грамм и нету драмм");  
 System.*out*.println("arrays1.num="+arrays1.getArrayString());  
 System.*out*.println("arrays2.num="+arrays2.getArrayString());  
 System.*out*.println("Проверка работы копирования объекта:");  
 Strings strings1 = new Strings("Состояние в норме...");  
 Strings strings2 = new Strings(strings1);  
 strings1.setInputString();  
 System.*out*.println("string1.inputString="+strings1.getInputString());  
 System.*out*.println("string2.inputString="+strings2.getInputString());  
 System.*out*.println("Проверка работы копирования по умолчанию");  
 Strings strings3 = new Strings();  
 Strings strings4 = new Strings();  
 strings4.setInputString();  
 System.*out*.println("string3.inputString="+strings3.getInputString());  
 System.*out*.println("string4.inputString="+strings4.getInputString());  
 break;  
  
 case 1:  
 arrays.setArrayString(strings.getInputString());  
 arrays.compareArray(arrays.getArrayString().split(" "), 3);  
 System.*out*.println(arrays.getArrayString());  
 break;  
  
 case 2:  
 System.*out*.println("Введите предложение: ");  
 String inputString = in.nextLine();  
 if (strings.getInputString().contains(inputString))  
 System.*out*.println("Найдено");  
 else  
 System.*out*.println("Не найдено");  
 break;  
  
 case 3:  
 arrays.setArrayString(strings.getInputString());  
 System.*out*.println(arrays.deleteArray(3));  
 break;  
  
 case 4:  
 return;  
  
 default:  
 System.*out*.println("Неверный ввод, повторите попытку...");  
 break;  
 }  
 }  
 }  
}

2) Класс Arrays

package lab2;  
  
import java.io.\*;  
  
  
public class Arrays implements Serializable {  
  
 private String[] arrayString;  
  
 public Arrays(){  
 this.arrayString = "Каждый принадлежит всем".split(" ");  
 }  
  
 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;  
 }  
 }  
 }  
 }  
  
 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();  
 }  
  
 public static Arrays copyOfSerialization(Arrays object) throws IOException, ClassNotFoundException {  
  
 //Сериализация с помощью класса ByteArrayOutputStream  
 ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();  
 ObjectOutputStream objectOutputStream = new ObjectOutputStream(byteArrayOutputStream);  
 objectOutputStream.writeObject(object);  
  
 objectOutputStream.flush();  
  
 //Восстановление с помощью класса ByteArrayInputStream  
 ObjectInputStream objectInputStream = new ObjectInputStream(  
 new ByteArrayInputStream(byteArrayOutputStream.toByteArray()));  
 Arrays objectRestoredFromByte = (Arrays) objectInputStream.readObject();  
  
 objectInputStream.close();  
  
 return objectRestoredFromByte;  
 }  
  
}

1. Класс Strings

package lab2;  
import java.util.Scanner;  
public class Strings {  
  
 private String inputString;  
 public Strings(){  
 this.inputString = "рааааал";  
 }  
 public Strings(String arg1){  
 this.inputString = arg1;  
 }  
 public Strings(Strings object){  
 this.inputString = object.getInputString();  
 }  
  
 public String getInputString(){  
 return inputString;  
 }  
  
 public void setInputString(){  
 Scanner in = new Scanner(System.*in*);  
 inputString = in.nextLine();  
 }  
  
 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>

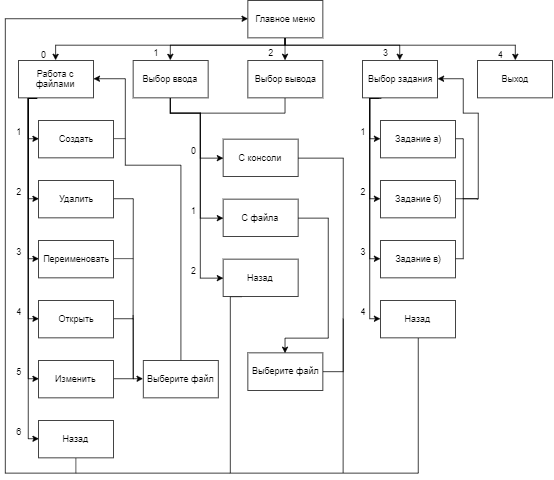
(дата обращения:08.09.2020)

1. StringBuffer, StringBuilder в Java [электронный ресурс],

URL: https://javarush.ru/groups/posts/2351-znakomstvo-so-string-stringbuffer-i-stringbuilder-v-java

(дата обращения:08.09.2020)

* 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;
* 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 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;
* 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 void openFiles() – позволяет посмотреть содержимое файл, который выбирает пользователь, вызывается методом listFiles()[choiceFile()-1];
* 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];



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 = Integer.*parseInt*(in.nextLine());  
 switch (num) {  
 case 1:  
 *fileMenu*();  
 break;  
 case 2:  
 *inputMode*();  
 break;  
 case 3:  
 *outputMode*();  
 break;  
 case 4:  
 *taskMenu*();  
 break;  
 case 5:  
 break;  
 default:  
 System.*out*.println("Некорректный ввод, повторите попытку...");  
 break;  
 }  
 if(num==5) break;  
 }  
 }  
 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 = Integer.*parseInt*(in.nextLine());  
 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;  
 default:  
 System.*out*.println("Некорректный ввод, повторите попытку...");  
 break;  
 }  
 if(num==3||num==2||num==1) break;  
 }  
 }  
  
 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 = Integer.*parseInt*(in.nextLine());  
 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;  
 default:  
 System.*out*.println("Некорректный ввод, повторите попытку...");  
 break;  
 }  
 if(num==3||num==2||num==1) break;  
 }  
 }  
  
 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 = Integer.*parseInt*(in.nextLine());  
 if (num < 3 && num > 0) {  
 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;  
 default:  
 System.*out*.println("Некорректный ввод, повторите попытку...");  
 break;  
 }  
 if(num==4) break;  
 }  
 }  
  
 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 = Integer.*parseInt*(in.nextLine());  
 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;  
 default:  
 System.*out*.println("Некорректный ввод, повторите попытку...");  
 break;  
 }  
 if(num==7) break;  
 }  
 }  
  
 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) + "|");  
 }  
}