

## Здравствуйте!

Вас приветствует участник отбора Ведин Дмитрий. В данном документе рассмотрен принцип работы, написанный по следующему тестовому заданию (версия JDK в приложении):

Утилита фильтрации содержимого файлов.

При запуске утилиты в командной строке подается несколько файлов, содержащих в перемешку целые числа, строки и вещественные числа. В качестве разделителя используется перевод строки. Строки из файлов читаются по очереди в соответствии с их перечислением в командной строке.

Задача утилиты записать разные типы данных в разные файлы. Целые числа в один выходной файл, вещественные в другой, строки в третий. По умолчанию файлы с результатами располагаются в текущей папке с именами `integers.txt`, `floats.txt`, `strings.txt`. Дополнительно с помощью опции `-o` нужно уметь задавать путь для результатов. Опция `-p` задает префикс имен выходных файлов. Например `-o /some/path -p result_` задают вывод в файлы `/some/path/result_integers.txt`, `/some/path/result_strings.txt` и тд.

По умолчанию файлы результатов перезаписываются. С помощью опции `-a` можно задать режим добавления в существующие файлы.

Файлы с результатами должны создаваться по мере необходимости. Если какого-то типа данных во входящих файлах нет, то и создавать исходящий файл, который будет заведомо пустым, не нужно.

В процессе фильтрации данных необходимо собирать статистику по каждому типу данных. Статистика должна поддерживаться двух видов: краткая и полная. Выбор статистики производится опциями `-s` и `-f` соответственно. Краткая статистика содержит только количество элементов записанных в исходящие файлы. Полная статистика для чисел дополнительно содержит минимальное и максимальное значения, сумма и среднее. Полная статистика для строк, помимо их количества, содержит также размер самой короткой строки и самой длинной.

Статистику по каждому типу отфильтрованных данных утилита должна вывести в консоль.

Все возможные виды ошибок должны быть обработаны. Программа не должна «падать». Если после ошибки продолжить выполнение невозможно, программа должна

сообщить об этом пользователю с указанием причины неудачи. Частичная обработка при наличии ошибок более предпочтительна чем останов программы. Код программы должен быть «чистым».

Для реализации необходимо использовать язык программирования Java, допустимо использовать стандартные системы сборки проекта (Maven, Gradle) Решение принимается в виде исходного кода проекта.

К решению должна прилагаться инструкция по запуску. В ней можно отображать особенности реализации, не уточненные в задании. В частности, в инструкции необходимо указывать:

- версию Java;
- при использовании системы сборки – указать систему сборки и ее версию;
- при использовании сторонних библиотек указать их название и версию, а также приложить ссылки на такие библиотеки (можно в формате зависимостей системы сборки).

## Оглавление

Инструкция.....	4
Анализ блоков .....	9
Приложение .....	11

## Инструкция

Данное решение поставленной задачи состоит из нескольких блоков:

1. Введение файлов;
2. Создание ArrayList для помещения всех данных в один массив;
3. Подсчёт элементов;
4. Создание специализированных массивов под каждый тип данных;
5. Создание файлов и запись информации;
6. Внедрение функций -р и -о;
7. Внедрение функции -а;
8. Сбор статистики и вывод статистических данных при помощи методов -s и -f.

Для работы программы необходимо в каталоге с кодом создать папку «Входные данные» и поместить туда файлы, которые планируется проанализировать. **ОБЯЗАТЕЛЬНО** назвать файлы, предназначенные для анализа, *in1*, *in2*, *in3* и т.д. Таким образом программа поймёт сколько файлов будет дано на вход и сможет корректно их обработать.

В ходе работы с программой в терминал будут выводиться подсказки для корректного управления программой. Стоит отметить, что при пропуске одного из действий вернуться к нему уже будет невозможно.

Для запуска программы, Вам потребуется любой из компиляторов (автор пользовался Microsoft Visual Studio). На старте программы будет запрошено количество анализируемых файлов (при указании количества явно большего файлы не пойдут в учёт).

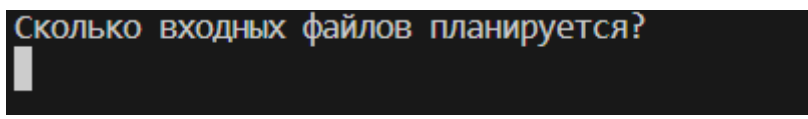


Рисунок 1. Первый шаг в реализации программы

Далее будет уведомление о количестве анализируемых файлов и их пути:

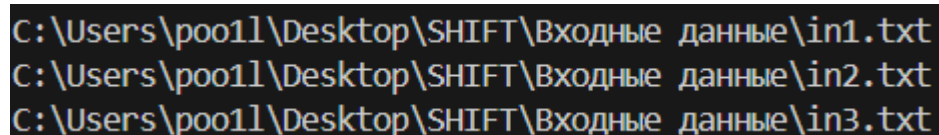


Рисунок 2. Последствия правильно выполненного первого шага

И будет указано следующее действие:

```
Вы можете использовать команды -p для создания префикса и -o для изменения пути; enter - выход
```

Рисунок 3. Настройка сохранения файла

```
Вы можете использовать команды -p для создания префикса и -o для изменения пути; enter - выход
-p
Введите желаемый префикс
SHIFT_
Вы можете использовать команды -p для создания префикса и -o для изменения пути; enter - выход
-o
Введите дополнительный путь для сохранения результатов
RES4SHIFT\MineRes
Вы можете использовать команды -p для создания префикса и -o для изменения пути; enter - выход
```

Рисунок 4. Пример настройки сохранения

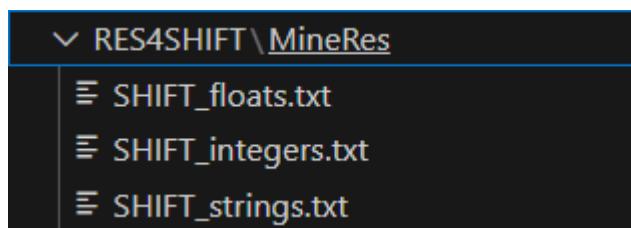


Рисунок 5. Результат настройки

Для выхода из этого режима достаточно нажать enter при выборе действия.

Для добавления различных данных была создана функция -a, которая активируется после выхода из настройки сохранения:

```
Вы можете использовать команды -p для создания префикса и -o для изменения пути; enter - выход

С помощью метода -a Вы можете добавить любые данные в программу
```

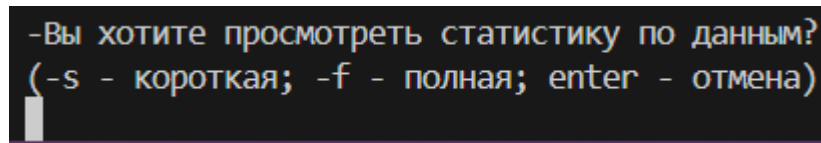
Рисунок 6. Активация функции -a

Для пропуска шага необходимо нажать enter, в случае же активации, появляется возможность добавлять неопределённое количество значений в итоговый файл:

```
С помощью метода -a Вы можете добавить любые данные в программу
-a
Введите добавочные элементы. Программа будет записывать Ваши действия до команды -c
1
Введите добавочные элементы. Программа будет записывать Ваши действия до команды -c
10000
Введите добавочные элементы. Программа будет записывать Ваши действия до команды -c
North
Введите добавочные элементы. Программа будет записывать Ваши действия до команды -c
2.718
Введите добавочные элементы. Программа будет записывать Ваши действия до команды -c
```

Рисунок 7. Внесение дополнительных данных

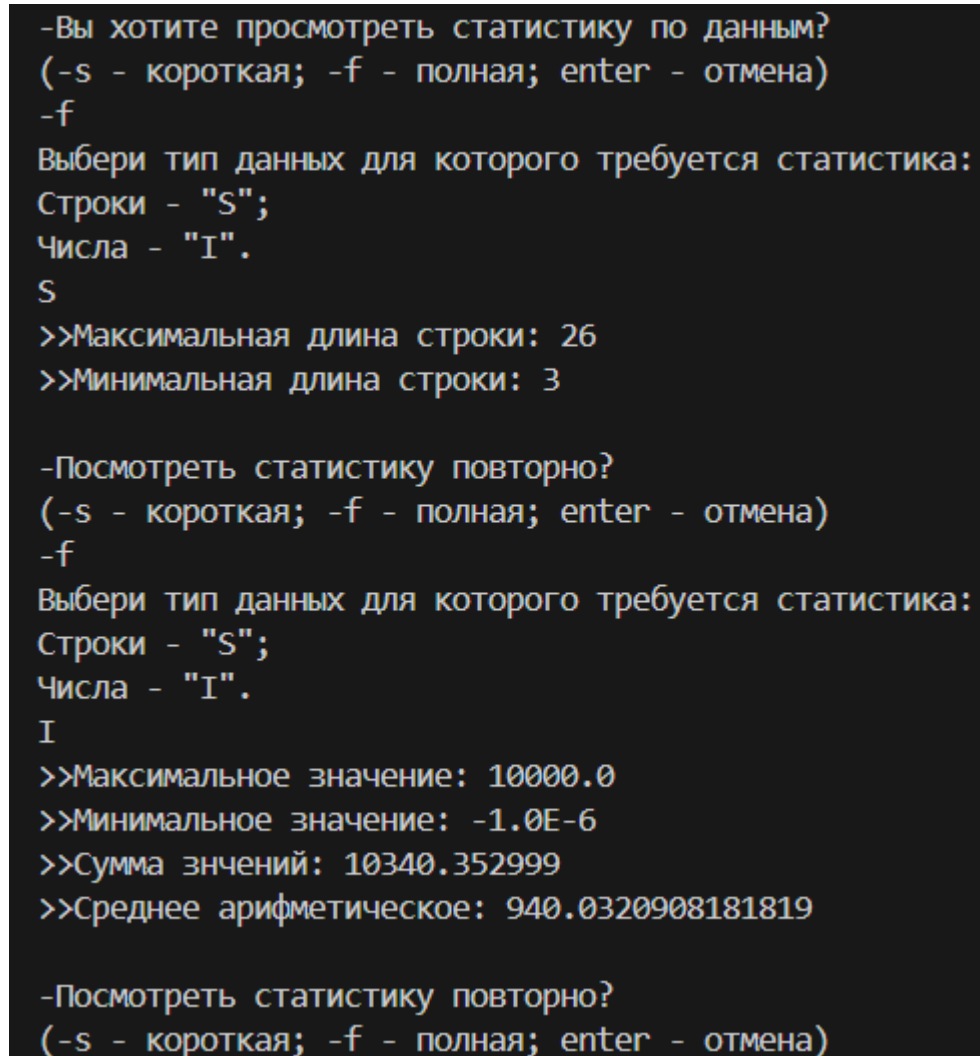
После обозначения конца (-с) появится последняя возможность что-то добавить.  
После внесения добавочных данных, появляется возможность посмотреть статистику:



```
-Вы хотите посмотреть статистику по данным?  
(-s - короткая; -f - полная; enter - отмена)  
_
```

Рисунок 8. Режим выбора статистики

Рассмотрим статистику на примере полной:



```
-Вы хотите посмотреть статистику по данным?  
(-s - короткая; -f - полная; enter - отмена)  
-f  
Выбери тип данных для которого требуется статистика:  
Строки - "S";  
Числа - "I".  
S  
>>Максимальная длина строки: 26  
>>Минимальная длина строки: 3  
  
-Посмотреть статистику повторно?  
(-s - короткая; -f - полная; enter - отмена)  
-f  
Выбери тип данных для которого требуется статистика:  
Строки - "S";  
Числа - "I".  
I  
>>Максимальное значение: 10000.0  
>>Минимальное значение: -1.0E-6  
>>Сумма значений: 10340.352999  
>>Среднее арифметическое: 940.0320908181819  
  
-Посмотреть статистику повторно?  
(-s - короткая; -f - полная; enter - отмена)
```

Рисунок 9. Окно полной статистики

Программа полностью завершается после выхода из окна статистики.

Итог:

### Входные данные

In1:

Lorem ipsum dolor sit amet

1

Пример

2.2

consectetur adipiscing

4.312

тестовое задание

-0.000001

In2:

(empty)

In3:

Seven

Six

8

312

09

0.123

Шмель

### Выходные данные

SHIFT\_integers.txt:

1

8

312

9

1

10000

SHIFT\_floats.txt:

2.2

4.312

-1.0E-6

0.123

2.718

SHIFT\_strings.txt:

Lorem ipsum dolor sit amet

Пример

consectetur adipiscing

тестовое задание

Seven

Six

Шмель

North

---

Для более подробного изучения программы, советую изучить исходный код в приложении, для анализа мной будет посвящён следующий раздел с анализом некоторых блоков.



## Анализ блоков

Данный пункт предназначен для подробного ознакомления с программой.

### Методы

В коде из приложения Вы можете столкнуться со следующими методами:

isInt

isDouble

isString

И хотя из названия понятно, что делают эти методы, хочется уделить внимание их содержанию:

```
private static boolean isInt(String s) throws NumberFormatException {
    try {
        Integer.parseInt(s);
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}

private static boolean isDouble(String s) throws NumberFormatException, NullPointerException {
    try {
        if (s != null) {
            Double.parseDouble(s);
        }
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}

private static boolean isString (String s) {
    if (s == " " | s.isBlank()) {
        return false;
    } else {
        return true;
    }
}
```

Необходимость в их создании возникла при распределении и подсчёте данных, поэтому каждый из методов возвращает свой тип значения, а метод isString отсеивает пропуски и пустые строки из программы.

В программе также присутствуют методы для создания файлов каждого из типов данных:

createNewIntFile

createNewDoubleFile

createNewStringFile

Они создают файлы и записывают в них информацию, которая содержится в определённых ранее массивах.

### Дополнительные функции (-p, -o, -a)

Каждая из этих функций реализована на механизме:

```
switch (condition) {  
    case value:  
        break;  
    default:  
        break;  
}
```

Итак, пример реализации функции -o:

```
case "-o":  
    dataInt.delete();  
    dataDouble.delete();  
    dataString.delete();  
    System.out.println(x:"Введите дополнительный путь для сохранения результатов");  
    additionalPathToFile = new Scanner(System.in);  
    additionalPath = additionalPathToFile.nextLine();  
    File additionalFile = new File(additionalPath);  
    if (!additionalFile.exists()) {  
        additionalFile.mkdirs();  
    }  
    pathToSave += separator + additionalPath + separator;  
    dataInt = new File(pathToSave + prefixName + "integers.txt");  
    createNewIntFile(intCounter, dataInt, intContainer);  
    dataDouble = new File(pathToSave + prefixName + "floats.txt");  
    createNewDoubleFile(doubleCounter, dataDouble, doubleContainer);  
    dataString = new File(pathToSave + prefixName + "strings.txt");  
    createNewStringFile(stringCounter, dataString, stringContainer);  
    break;
```

Функция удаляет файлы и создаёт новые в каталогах, которые создаются другой функцией: File.mkdirs().

## Приложение

Программа была написана на версии JDK: 17.0.13.

```
import java.util.ArrayList;
import java.util.Scanner;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.nio.charset.StandardCharsets;

public class Shift {
    public static void main(String[] args) throws FileNotFoundException,
    IOException{

        //Введение файлов
        File absDirection = new File("");
        String originalPath = absDirection.getCanonicalPath();
        String separator = File.separator;

        System.out.println("Сколько входных файлов планируется?");
        Scanner sc = new Scanner(System.in);
        String answer0 = sc.nextLine();
        while (!isInt(answer0)) {
            System.out.println("Необходимо вписать число входящих файлов");
            answer0=sc.nextLine();
        }

        String path;
        File inFile;
        Scanner fileScanner;
        ArrayList<String> dataArrayList = new ArrayList<>();

        for (int i = 0; i < Integer.parseInt(answer0); i++) {
            path = originalPath + separator + "Входные данные" + separator + "in"
+ (i+1) + ".txt";
            inFile = new File(path);
            if (inFile.exists()) {
                System.out.println(path);
                fileScanner = new Scanner(inFile, StandardCharsets.UTF_8);
                while (fileScanner.hasNextLine()) {
                    dataArrayList.add(fileScanner.nextLine());
                }
            }
        }
        String[] data = dataArrayList.toArray(new
String[dataArrayList.size()]);
    }
}
```

```

// Подсчёт элементов каждого типа
int ctInt = 0, ctDouble = 0, ctString = 0;
for (int i = 0; i < dataArrayList.size(); i++){
    if ( isInt(data[i])) {
        ctInt++;
    }else if (isDouble(data[i])) {
        ctDouble++;
    }else if (isString(data[i])) {
        ctString++;
    }else {
        continue;
    }
}

// Перераспределение элементов по трём массивам: stringContainer,
intContainer и doubleContainer
int intCounter=0, doubleCounter=0, stringCounter=0;
int[] intContainer = new int[ctInt];
Double[] doubleContainer = new Double[ctDouble];
String[] stringContainer = new String[ctString];
for (int i = 0; i < data.length; i++){
    if (data[i].equals(null)) {
        continue;
    }
    if ( isInt(data[i])) {
        intContainer[intCounter] = Integer.parseInt(data[i]);
        intCounter++;
    }else if (isDouble(data[i])) {
        doubleContainer[doubleCounter] = Double.parseDouble(data[i]);
        doubleCounter++;
    }else if (isString(data[i])) {
        stringContainer[stringCounter] = data[i];
        stringCounter++;
    }else {
        continue;
    }
}

String pathToSave = absDirection.getAbsolutePath();
String dataIntPath = pathToSave + separator;
String dataDoublePath = pathToSave + separator;
String dataStringPath = pathToSave + separator;

// Создание файлов и запись информации с специализированных массивов

// Числа
File dataInt = new File(dataIntPath + "integers.txt");
createNewIntFile(intCounter, dataInt, intContainer);

// Числа с дробной частью

```

```

File dataDouble = new File(dataDoublePath + "floats.txt");
createNewDoubleFile(doubleCounter, dataDouble, doubleContainer);

// Строки
File dataString = new File(dataStringPath + "strings.txt");
createNewStringFile(stringCounter, dataString, stringContainer);

//Введение метода добавления префикса и изменения пути создания файла
Scanner prefix, additionalPathToFile;
String prefixName = "", additionalPath;
Scanner methodsScanner = new Scanner(System.in);
System.out.println("Вы можете использовать команды -p для создания
префикса и -o для изменения пути; enter - выход");
String methods;
methods = methodsScanner.nextLine();
while ( methods != "") {
    switch (methods) {
        case "-p":
            dataInt.delete();
            dataDouble.delete();
            dataString.delete();
            prefix = new Scanner(System.in);
            System.out.println("Введите желаемый префикс ");
            prefixName = prefix.nextLine();
            dataIntPath = pathToSave + prefixName + "integers.txt";
            dataDoublePath = pathToSave + prefixName + "floats.txt";
            dataStringPath = pathToSave + prefixName + "strings.txt";
            dataInt = new File(pathToSave + separator + prefixName +
"integers.txt");
            createNewIntFile(intCounter, dataInt, intContainer);
            dataDouble = new File(pathToSave + separator + prefixName +
"floats.txt");
            createNewDoubleFile(doubleCounter, dataDouble,
doubleContainer);
            dataString = new File(pathToSave + separator + prefixName +
"strings.txt");
            createNewStringFile(stringCounter, dataString,
stringContainer);
            break;
        case "-o":
            dataInt.delete();
            dataDouble.delete();
            dataString.delete();
            System.out.println("Введите дополнительный путь для
сохранения результатов");
            additionalPathToFile = new Scanner(System.in);
            additionalPath = additionalPathToFile.nextLine();
            File additionalFile = new File(additionalPath);
            if (!additionalFile.exists()) {

```

```

        additionalFile.mkdirs();
    }
    pathToSave += separator + additionalPath + separator;
    dataInt = new File(pathToSave + prefixName + "integers.txt");
    createNewIntFile(intCounter, dataInt, intContainer);
    dataDouble = new File(pathToSave + prefixName +
"floats.txt");
    createNewDoubleFile(doubleCounter, dataDouble,
doubleContainer);
    dataString = new File(pathToSave + prefixName +
"strings.txt");
    createNewStringFile(stringCounter, dataString,
stringContainer);
    break;
    default:
        break;
    }
    System.out.println("Вы можете использовать команды -p для создания
префикса и -o для изменения пути; enter - выход");
    methods = methodsScanner.nextLine();
}

//Внедрение функции расширения данных
String additionalMethods, additionalAnswer;
Scanner additionalScanner = new Scanner(System.in);
Scanner additionerListener = new Scanner(System.in);
FileWriter additionalIntWriter = new FileWriter(dataInt, true);
FileWriter additionalDoubleWriter = new FileWriter(dataDouble, true);
FileWriter additionalStringWriter = new FileWriter(dataString, true);
System.out.println("С помощью метода -a Вы можете добавить любые данные в
программу");
additionalMethods = additionalScanner.nextLine();
while ( additionalMethods != "" ) {
    switch (additionalMethods) {
        case "-a":
            System.out.println("Введите добавочные элементы. Программа
будет записывать Ваши действия до команды -c");
            additionalAnswer = additionerListener.nextLine();
            if ( additionalAnswer.equals("-c")) {
                break;
            }else {
                dataArrayList.add(additionalAnswer);
                if ( isInt(additionalAnswer)) {
                    Integer.parseInt(additionalAnswer);
                    additionalIntWriter.write(additionalAnswer + "\n");
                    ctInt++;
                }else if ( isDouble(additionalAnswer)) {
                    Double.parseDouble(additionalAnswer);
                    additionalDoubleWriter.write(additionalAnswer +
"\n");

```

```

        ctDouble++;
    }else if (isString(additionalAnswer)) {
        additionalStringWriter.write(additionalAnswer +
"\n");

        ctString++;
    }
    continue;
}
default:
    break;
}
System.out.println("С помощью метода -a Вы можете добавить любые
данные в программу");
additionalMethods = additionalScanner.nextLine();
}
additionalIntWriter.close();
additionalDoubleWriter.close();
additionalStringWriter.close();

// Сбор статистики
int maxStringLen = 0, minStringLen = stringContainer[0].length();
Double sumNum = 0.0, averageNum = null, maxNum = 0.0, minNum = 0.0;
if (intCounter > 0) {
    minNum = Double.valueOf(intContainer[0]);
}

data = dataArrayList.toArray(new String[dataArrayList.size()]);
for (int i = 0; i < dataArrayList.size();i++) {
    if (isInt(data[i]) || isDouble(data[i])) {
        if (Double.parseDouble(data[i]) > Double.valueOf(maxNum)) {
            maxNum = Double.parseDouble(data[i]);
        }
        if (Double.parseDouble(data[i]) < Double.valueOf(minNum)) {
            minNum = Double.parseDouble(data[i]);
        }
        sumNum += Double.parseDouble(data[i]);
    }else if (isString(data[i])) {
        if (data[i].length() > maxStringLen) {
            maxStringLen = data[i].length();
        }
        if (data[i].length() < minStringLen) {
            minStringLen = data[i].length();
        }
    }else {
        continue;
    }
}
averageNum = sumNum/(ctInt+ctDouble);

//Работа с доп. опциями ([-s, -f] - статистика)

```

```

        System.out.println("-Вы хотите посмотреть статистику по данным? \n(-s -
короткая; -f - полная; enter - отмена)");
        Scanner option = new Scanner(System.in);
        Scanner chose = new Scanner(System.in);
        String stringOrInt;
        String additionalOption;

        additionalOption = option.nextLine();
        while (additionalOption != "") {
            switch (additionalOption) {
                case "-f":
                    System.out.println("Выбери тип данных для которого требуется
статистика: \nСтроки - \"S\";\nЧисла - \"I\".");
                    stringOrInt = chose.nextLine();
                    switch (stringOrInt) {
                        case "S":
                            System.out.println(">>Максимальная длина строки: " +
maxStringLen + "\n>>Минимальная длина строки: " + minStringLen);
                            break;
                        case "I":
                            System.out.println(">>Максимальное значение: " +
maxNum + " \n>>Минимальное значение: " + minNum + " \n>>Сумма значений: " + sumNum
+ "\n>>Среднее арифметическое: " + averageNum);
                            break;
                        default:
                            break;
                    }
                    break;
                case "-s":
                    System.out.println("Выбери тип данных для которого требуется
статистика: \nСтроки - \"S\";\nЧисла - \"I\".");
                    stringOrInt = chose.nextLine();
                    switch (stringOrInt) {
                        case "S":
                            System.out.println(">>Количество строк: " +
ctString);
                            break;
                        case "I":
                            System.out.println(">>Количество чисел: " + (ctInt +
ctDouble));
                            break;
                        default:
                            break;
                    }
                    break;
                default:
                    break;
            }
        }
        System.out.println("\n-Посмотреть статистику повторно? \n(-s -
короткая; -f - полная; enter - отмена)");

```



```

        additionalOption = option.nextLine();
    }
    sc.close();
    methodsScanner.close();
    option.close();
    choise.close();
    additionerListenner.close();
    additionalScanner.close();
}

private static boolean isInt(String s) throws NumberFormatException {
    try {
        Integer.parseInt(s);
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}

private static boolean isDouble(String s) throws NumberFormatException,
NullPointerException {
    try {
        if (s != null) {
            Double.parseDouble(s);
        }
        return true;
    } catch (NumberFormatException e) {
        return false;
    }
}

private static boolean isString (String s) {
    if (s == " " | s.isBlank()) {
        return false;
    } else {
        return true;
    }
}

private static void createNewIntFile (int counter, File data, int[]
array) throws FileNotFoundException{
    if (counter <= 0) {
        data.delete();
    } else {
        PrintWriter dataPrinter = new PrintWriter(data);
        for(int i = 0; i < array.length; i++){
            dataPrinter.println(array[i]);
        }
        dataPrinter.close();
    }
}

private static void createNewDoubleFile (int counter, File data, Double[]
array) throws FileNotFoundException{

```

```

        if (counter <= 0) {
            data.delete();
        } else {
            PrintWriter dataPrinter = new PrintWriter(data);
            for(int i = 0; i < array.length; i++){
                dataPrinter.println(array[i]);
            }
            dataPrinter.close();
        }
    }

    private static void createNewStringFile (int counter, File data, String[]
array) throws FileNotFoundException{
        if (counter <= 0) {
            data.delete();
        } else {
            PrintWriter dataPrinter = new PrintWriter(data);
            for(int i = 0; i < array.length; i++){
                dataPrinter.println(array[i]);
            }
            dataPrinter.close();
        }
    }
}

```