Задание ЦФТ:

Написать программу сортировки слиянием нескольких файлов.

JDK: Oracle OpenJDK version 17.0.5

Build System: Gradle

ВариантNo1 : App.java

https://github.com/StanleyKeys/SHIFT_test_01/blob/master/src/main/java/App.java

Программу можно запустить с параметрами, который принимается в виде массива.

Например:

Javac App

Java App.java -i –a out.txt in1.txt in2.txt in3.txt

Либо его можно передать как массив строк **String[] userArgs** где:

args[0] = параметр типа данных -i –s

args[1] = параметр сортировки -a –d

args[2] = название выходного файла

Остальные элементы args = названия или абсолютный путь к входным файлам.

Функция searchTypeCommand(String[] args) = принимает на вход массив параметров и возвращает необходимый в список команд (commandList)

Функция searchSortCommand(String[] args) = принимает на вход массив параметров и возвращает необходимый тип сортировки в список команд (commandList)

```java
public class App {
    StanleyGunz
    public static void main(String[] args) throws IOException {
        App app = new App();
        System.out.println(Arrays.toString(args));

        String[] userArgs = {"-i", "-a", "out.txt", "in1.txt", "in2.txt", "in3.txt"};
        app.chooseCommand(args);

    }

    1 usage  StanleyGunz
    public String searchTypeCommand(String[] args) {
        /*
            Проверяем параметр типа данных.
            -i = int
            -s = str
         */
        for (String arg : args) {
            if (arg.equals("-i")) {
                return arg;
            } else if (arg.equals("-s")) {
                return arg;
            }
        }
        return null;
    }

    1 usage  StanleyGunz
    public String searchSortCommand(String[] args) {
        /*
            Проверяем параметр сортировки.
            -a = ascending
            -d = descending
         */
        for (String arg : args) {
            if (arg.contains("-d")) {
                return "-d";
            }
        }
        return "-a";
    }
```

Функция **chooseCommand(String[] args)** = является основной функцией, которая:

Создает списки для чисел, строк, команд, и файлов.

Заполняет список комманд параметрами.

Заполняет список файлов названиями (или ссылками)

Проверяет на наличие входных и выходных файлов

Проверяет наличие параметра для типа данных

Заполняет список и сортирует.

```java
public void chooseCommand(String[] args) throws IOException {
    /*
        Основной метод, который:
        1. Создает списки для чисел, строк, команд, и файлов.
        2. Заполняет commandList параметрами полученными от пользователя.
        3. Заполняет fileList названиями файлов.
        4. Проверяет, ввел ли пользователь Выходной и Входные файлы.
        5. Проверяет наличие параметра для типа данных.
        6. Заполняет необходимый список и сортирует его.
        7. Записывает список(или массив) в выходной файл.
     */
    ArrayList<String> strList;
    ArrayList<Integer> numList;
    ArrayList<String> commandList = new ArrayList<>();
    ArrayList<String> fileList = new ArrayList<>();
    commandList.add(searchTypeCommand(args));
    commandList.add(searchSortCommand(args));


    for (String arg : args) {
        if (arg.contains(".txt")) {
            fileList.add(arg);
        }
    }
    if (fileList.size() == 0) {
        System.out.println("Необходимо ввести в параметрах название выходного файла. \nНапример: out.txt");
    } else if (fileList.size() == 1) {
        System.out.println("В параметрах должен быть хотя бы один входной файл. \nНапример: in.txt");
    } else {
        if (commandList.get(0) == null) {
            System.out.println("Вы не ввели параметр для типа данных. \nДопустимые: -i Integer, -s String");
        } else if (commandList.get(0).equals("-i")) {
            numList = fillNumList(fileList);

            // Основное решение:
            if (commandList.get(1).equals("-a")) {
                Collections.sort(numList);
            } else {
                numList.sort(Collections.reverseOrder());
            }
            saveToFile(numList, fileList.get(0));
```

Функции **fillNumList** и **fillStrList** = заполняют списки из входных файлов

Функция **isNumeric** = проверяет символы в строках.

```java
public ArrayList<Integer> fillNumList(ArrayList<String> fileList) throws IOException {
    /*
        Функция:
        1. Принимает список файлов *.txt и проходится циклом. (лучше абсолютный путь к файлу)
        2. Проверяет на наличие строк с пробелами.
        3. Проверяет символы на цифры.
        4. Возвращает список чисел.
     */
    ArrayList<Integer> numList = new ArrayList<>();
    for (int i = 1; i < fileList.size(); i++) {
        String filePath = fileList.get(i);
        //String filePath = String.format("T:\\Important\\Projects\\IdeaProjects\\SHIFT_test_01
        BufferedReader reader = new BufferedReader(new FileReader(filePath));
        String line = reader.readLine();
        while (line != null) {
            if (!line.contains(" ")) {
                boolean b = isNumeric(line);
                if (b) {
                    numList.add(Integer.parseInt(line));
                }
            }
            line = reader.readLine();
        }
    }

    return numList;
}

1 usage  StanleyGunz
public ArrayList<String> fillStrList(ArrayList<String> fileList) throws IOException {
    /*
        Функция:
        1. Принимает список файлов *.txt и проходится циклом. (лучше абсолютный путь к файлу)
        2. Проверяет на наличие строк с пробелами.
        3. Проверяет символы на строки.
        4. Возвращает список строк.
     */
    ArrayList<String> strList = new ArrayList<>();
    for (int i = 1; i < fileList.size(); i++) {
        String filePath = fileList.get(i);
        // String filePath = String.format("T:\\Important\\Projects\\IdeaProjects\\SHIFT_test_0
```

В конце Функция **saveToFile** = записывает результирующий список в Выходной файл.

P.S. Да, в этой программе отсутствует сортировка слиянием и проверка на больших файлах. Поторопился, заволновался и совсем забыл как все делать.

Вариант №2 : AnotherApp

https://github.com/StanleyKeys/SHIFT_test_01/blob/master/src/main/java/AnotherApp.java

В данной программе все так же есть Функции:

SearchTypeCommand()

SearchSortCommand()

ChooseCommand()

Функция **fillList()** = Создает список, создает переменную, отвечающую за ограничение используемой памяти. Заполняет список и отправляет на запись в файл.

В случае если память превышает установленный предел (`usedMemoryLimit`), то программа прерывает чтение файлов(их же может быть много), затем полученный список сортирует и записывает в выходной файл.

Затем освобождается память и снова читает файлы.

```java
public void fillList(ArrayList<String> fileList, String sortCommand) throws IOException {
    /*
        1. Создаем список.
        2. Создаем переменную, которая будет проверять ограничение используемой памяти.
        3. Заполняем список и отправляем на запись в файл.
    */

    ArrayList<String> strList = new ArrayList<>();

    MemoryMXBean memoryMXBean = ManagementFactory.getMemoryMXBean();
    System.out.printf("Max heap memory: %.2f GB%n",
            (double) memoryMXBean.getHeapMemoryUsage().getMax() / 1073741824);

    // Проверяем допустимое кол-во памяти, затем используем только часть его.
    // Берем прям очень маленькое значение, чтоб проверить работоспособность.
    double usedMemoryLimit = Math.ceil((((double) memoryMXBean.getHeapMemoryUsage().getMax() / 1073741824) / (fileList.size() - 1)) * 10);
    usedMemoryLimit = usedMemoryLimit / 10 - 0.5;

    System.out.println(usedMemoryLimit);

    for (int i = 1; i < fileList.size(); i++) {
        BufferedReader reader = new BufferedReader(new FileReader(fileList.get(i)));
        String line = reader.readLine();
        double usedMemory = (double) memoryMXBean.getHeapMemoryUsage().getUsed() / 1073741824;
        System.out.printf("Used heap memory: %.2f GB%n", usedMemory);

        /*
            Если используемая память превышает допустимую (usedMemory), то программа прерывает чтение файлов,
            затем полученный список сортирует и записывает в Выходной файл.
            Освобождает память и снова читает файлы.
        */
        if (usedMemory >= usedMemoryLimit) {
            System.out.printf("Used Memory is higher than %,.1f GB \n", usedMemoryLimit);
            saveToFile(strList, sortCommand, fileList.get(0));
            strList.clear();
            System.out.println("successfully cleared List");
        }
        while (line != null) {
            if (!line.contains(" ")) {
                strList.add(line);
                line = reader.readLine();
```

Функция **sortTheList** = получает список и сортирует по команде.

Функция **saveToFile** = сохраняет результат в файл.

```java
public ArrayList<Integer> sortTheList(ArrayList<String> strList, String command) {
    /*
        1. Получаем список и команду по сортировке.
        2. Конвертируем и сортируем (стандартными библиотеками).
    */
    System.out.println("Converting String to Integer");
    ArrayList<Integer> numList = new ArrayList<>();
    for (String s : strList) {
        int temp = Integer.parseInt(s);
        numList.add(temp);
    }
    if (command.equals("-a")) {
        Collections.sort(numList);
    } else {
        numList.sort(Collections.reverseOrder());
    }
    return numList;
}

2 usages  StanleyGunz
public void saveToFile(ArrayList<String> strList, String sortCommand, String outFile) throws IOException {
    /*
        1. Создаем список для чисел.
        2. Конвертируем из String в int и наоборот.

        P.S. Если сразу делать запись strList, то по непонятным причинам идет запись в кодировке UTF-16 (иероглифы)
        хотя везде стоит UTF-8.
    */

    ArrayList<Integer> numList = sortTheList(strList, sortCommand);

    FileWriter writer = new FileWriter(outFile,  append: true);
    System.out.println("Saving out file");
    for (int value : numList) {
        writer.write( str: value + System.getProperty("line.separator"));
    }
    writer.close();
    numList.clear();
    System.out.println("successfully saved the out file");
}
```

P.S. сортировка производится стандартными библиотеками. Программу написал только для целочисленных данных.

НО, хотя бы проверяется загрузка памяти.

Вариант №3 : MergeSortApp

https://github.com/StanleyKeys/SHIFT_test_01/blob/master/src/main/java/MergeSortApp.java

Программа также имеет функции:

SearchTypeCommand()

SearchSortCommand()

ChooseCommand()

FillList()

Функции **mergeSort()** и **merge()** занимаются полученным списком, конвертируют в массив, и производят сортировку слиянием.

mergeSort()

```
private int[] mergeSort(ArrayList<String> strList, String sortCommand) {
    /*
        1. Получаем список, создаем массив и заполняем его.
        2. Проверяем команду сортировки.
        3. Сортируем слиянием.
    */
    int[] array = new int[strList.size()];
    for (int i = 0; i < array.length; i++) {
        array[i] = Integer.parseInt(strList.get(i));
    }
    if (sortCommand.equals("-a")) {
        int[] temp;
        int[] sourceArray = array;
        int[] destArray = new int[array.length];

        int size = 1;
        while (size < array.length) {
            for (int i = 0; i < array.length; i += 2 * size) {
                merge(sourceArray, i, sourceArray,  source2Start: i + size, destArray, i, size);
            }

            temp = sourceArray;
            sourceArray = destArray;
            destArray = temp;

            size = size * 2;

        }
        return sourceArray;
    } else {

          Integer[] integerArray = IntStream.of(array).boxed().toArray(Integer[]::new);
          Arrays.sort(integerArray, Collections.reverseOrder());
          return Arrays.stream(integerArray).mapToInt(i -> i).toArray();

        return Arrays.stream(array).boxed().sorted(Collections.reverseOrder()).mapToInt(Integer::intValue)
                .toArray();
    }
}
```

merge()

```
1 usage  StanleyGunz
private void merge(int[] sourceArray1, int source1Start, int[] sourceArray2,
                   int source2Start, int[] destArray, int destStart, int size) {
    int index1 = source1Start;
    int index2 = source2Start;

    int source1End = Math.min(source1Start + size, sourceArray1.length);
    int source2End = Math.min(source2Start + size, sourceArray2.length);

    if (source1Start + size > sourceArray1.length) {
        for (int i = source1Start; i < source1End; i++) {
            destArray[i] = sourceArray1[i];
        }
        return;
    }

    int iterationCount = source1End - source1Start + source2End - source2Start;

    for (int i = destStart; i < destStart + iterationCount; i++) {
        if (index1 < source1End && (index2 >= source2End || sourceArray1[index1] < sourceArray2[index2])) {
            destArray[i] = sourceArray1[index1];
            index1++;
        } else {
            destArray[i] = sourceArray2[index2];
            index2++;
        }
    }
}

2 usages  StanleyGunz
public void saveToFile(int[] array, String outFile) throws IOException {
    FileWriter writer = new FileWriter(outFile,  append: true);
    for (int value : array) {
        writer.write( str: value + System.getProperty("line.separator"));
    }
    writer.close();
    System.out.println("successfully saved the out file");

}
```

Данная программа принимает на вход параметры запуска(или массив).

Проходится циклом по файлам, заполняет память, ЕСЛИ получает ограничение - прерывает цикл, сортирует слиянием, записывает массив в файл, освобождает память, продолжает заполнять список, и так по кругу, пока не просмотрит полученный список файлов.

P.S. Отсутствует сортирока для буквенных символов. (только для численных)

P.P.S. В описании ТЗ заметил ошибку (или опечатку) :

Параметры программы задаются при запуске через аргументы командной строки, по порядку:

1. режим сортировки (-a или -d), необязательный, по умолчанию сортируем по возрастанию;
2. тип данных (-s или -i), обязательный;
3. имя выходного файла, обязательное;
4. остальные параметры – имена входных файлов, не менее одного.

Примеры запуска из командной строки для Windows:

sort-it.exe -i -a out.txt in.txt (для целых чисел по возрастанию)

sort-it.exe -s out.txt in1.txt in2.txt in3.txt (для строк по возрастанию)

sort-it.exe -d -s out.txt in1.txt in2.txt (для строк по убыванию)

В задании указаны параметры и что они должны быть “по порядку”

1. параметр сортировки
2. Параметр типа данных
3. Имя выходного файла
4. Имена входных файлов

В первом примере порядок параметров нарушен. Сначала идет параметр типа данных, затем параметр сортировки.

Поэтому в программы я добавил функции, проверяющие введеные параметры НЕ на своих местах.

С Уважением, Станислав Кинслер.

https://github.com/StanleyKeys

:)