МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчёт
по лабораторной работе №2

Выполнил:
студент группы ПО-9
Солышко Дмитрий Андреевич

Проверил:
Крощенко А. А.

Брест 2024

Вариант 6

Цель работы: приобрести практические навыки обработки параметров командной строки, закрепить базовые знания языка программирования Java при решении практических задач

Задание 1

Напишите программу сравнения двух файлов, которая будет печатать первую строку и позицию символа, где они различаются. В противном случае должно выводится сообщение об эквивалентности содержимого файлов.

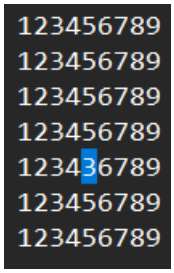
Код программы:

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

public class Task01 {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Введите путь первого файла: ");
        String file1Path = scanner.nextLine();

        System.out.print("Введите путь второго файла: ");
        String file2Path = scanner.nextLine();

        try {
            if (areFilesEqual(file1Path, file2Path)) {
                System.out.println("Файлы эквивалентны");
            } else {
                System.out.println("Файлы отличаются:");
                printFirstDifference(file1Path, file2Path);
            }
        } catch (IOException e) {
            System.err.println("Ошибка при чтении файлов: " + e.getMessage());
        }
    }

    private static boolean areFilesEqual(String file1Path, String file2Path) throws IOException {
        try (BufferedReader reader1 = new BufferedReader(new FileReader(file1Path));
             BufferedReader reader2 = new BufferedReader(new FileReader(file2Path))) {

            String line1, line2;
            while ((line1 = reader1.readLine()) != null && (line2 = reader2.readLine()) != null) {
                if (!line1.equals(line2)) {
                    return false;
                }
            }

            return reader1.readLine() == null && reader2.readLine() == null;
        }
    }

    private static void printFirstDifference(String file1Path, String file2Path) throws IOException {
```

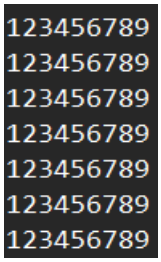
```
        try (BufferedReader reader1 = new BufferedReader(new FileReader(file1Path));
             BufferedReader reader2 = new BufferedReader(new FileReader(file2Path))) {

            int lineNumber = 1;
            int position = 0;

            String line1, line2;
            while ((line1 = reader1.readLine()) != null && (line2 = reader2.readLine()) != null) {
                if (!line1.equals(line2)) {
                    position = findFirstDifferencePosition(line1, line2);
                    break;
                }
                lineNumber++;
            }

            System.out.println("Первая различающаяся строка: " + lineNumber);
            System.out.println("Позиция различия символов: " + position);
        }
    }

    private static int findFirstDifferencePosition(String str1, String str2) {
        int position = 0;
        while (position < str1.length() && position < str2.length() && str1.charAt(position) == str2.charAt(position))
{
            position++;
        }
        return position + 1;
    }
}
```

Результат работы программы:

Первый файл:



Второй файл:



```
Введите путь первого файла: D:\1.txt
Введите путь второго файла: D:\2.txt
Файлы отличаются:
Первая различающаяся строка: 5
Позиция различия символов: 5

Process finished with exit code 0
```

Теперь исправим ошибку, и сделаем файлы идентичными:

```
Введите путь первого файла: D:\1.txt
Введите путь второго файла: D:\2.txt
Файлы эквивалентны
```

Задание 2

Утилита split копирует и разбивает файл на отдельные файлы заданной длины. В качестве аргументов ей надо указать имя исходного файла и префикс имен выходных файлов. Если файл не задан или задан как –, программа читает стандартный ввод.
По умолчанию размер части разбиения равен 10 строк, а префикс равен x. Имена выходныхфайлов будут составляться из этого префикса и двух дополнительных букв aa, ab, ac и т.д. (без пробелов и точек между префиксом и буквами). Если префикс имен файлов не задан, то по умолчанию используется x, так что выходные файлы будут называться xaa, xab и т. д.

Формат использования: split [-b | -l] [-d] [входной_файл [префикс_выходных_файлов]] где ключи имеют следующее значение:
• -b , --bytes=num Записывать в каждый выходной файл заданное число num байт. При задании числа байт можно использовать суффиксы: b означает байты, k – 1kb , m – 1Mb.
• -l , --lines=num Записывать в каждый выходной файл num строк.
• -d , --numericsuffixes Использовать числовые, а не алфавитные суффиксы, начинающиеся с 00. Суффиксы файлов будут иметь вид: 00, 01, 02 и т. д.

Код программы

```
import java.io.*;

public class Task02 {
    public static void main(String[] args) {
        if (args.length < 4 || !args[0].equals("split")) {
            System.out.println("Usage: java SplitUtility split [-b num | -l num] [-d] [input_file] [output_prefix]");
            System.exit(1);
        }

        int blockSize = 0;
        int lineCount = 10;
        boolean useNumericSuffixes = false;
        String inputFile = null;
        String outputPrefix = "x";

        try {
            int i = 1;
```

```
            while (i < args.length) {
                switch (args[i]) {
                    case "-b":
                        blockSize = parseSizeArgument(args[++i]);
                        lineCount = 0;
                        break;
                    case "-l":
                        lineCount = Integer.parseInt(args[++i]);
                        break;
                    case "-d":
                        useNumericSuffixes = true;
                        break;
                    default:
                        if (inputFile == null) {
                            inputFile = args[i];
                        } else {
                            outputPrefix = args[i];
                        }
                        break;
                }
                i++;
            }

            if (blockSize == 0 && lineCount == 0) {
                System.out.println("Error: You must specify either -b or -l option.");
                System.exit(1);
            }

            splitFile(inputFile, outputPrefix, blockSize, lineCount, useNumericSuffixes);
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            System.exit(1);
        }
    }

    private static int parseSizeArgument(String size) {
        size = size.toLowerCase();
        if (size.endsWith("b")) {
            return Integer.parseInt(size.substring(0, size.length() - 1));
        } else if (size.endsWith("k")) {
            return Integer.parseInt(size.substring(0, size.length() - 1)) * 1024;
        } else if (size.endsWith("m")) {
            return Integer.parseInt(size.substring(0, size.length() - 1)) * 1024 * 1024;
        } else {
            return Integer.parseInt(size);
        }
    }

    private static void splitFile(String inputFile, String outputPrefix, int blockSize, int lineCount, boolean
useNumericSuffixes) throws IOException {
        BufferedReader reader = null;
        try {
            reader = (inputFile == null || inputFile.equals("-")) ?
                    new BufferedReader(new InputStreamReader(System.in)) :
```

```
                new BufferedReader(new FileReader(inputFile));

            StringBuilder block = new StringBuilder();
            int blockNumber = 0;
            int tempLineCount = lineCount;

            int ch;
            while ((ch = reader.read()) != -1) {
                char character = (char) ch;

                if (tempLineCount > 0) {
                    block.append(character);
                    if (character == '\n') {
                        tempLineCount--;
                        if (tempLineCount == 0) {
                            writeBlock(outputPrefix, block.toString(), blockNumber++, useNumericSuffixes);
                            block.setLength(0);
                            tempLineCount = lineCount;
                        }
                    }
                } else if (blockSize > 0) {
                    int charSize = Character.toString(character).getBytes("UTF-8").length;
                    if (block.length() + charSize > blockSize) {
                        writeBlock(outputPrefix, block.toString(), blockNumber++, useNumericSuffixes);
                        block.setLength(0);
                    }
                    block.append(character);
                }
            }

            if (block.length() > 0) {
                writeBlock(outputPrefix, block.toString(), blockNumber, useNumericSuffixes);
            }
        }
        finally {
            if (reader != null && inputFile != null && !inputFile.equals("-")) {
                reader.close();
            }
        }
    }

    private static void writeBlock(String outputPrefix, String block, int blockNumber, boolean
useNumericSuffixes) throws IOException {
        String suffix = useNumericSuffixes ? String.format("%02d", blockNumber) :
getAlphabeticSuffix(blockNumber);
        String outputFileName = outputPrefix + suffix + ".txt";
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(outputFileName))) {
            writer.write(block);
        }
        System.out.println("Created: " + outputFileName);
    }

    private static String getAlphabeticSuffix(int number) {
        StringBuilder suffix = new StringBuilder();
```

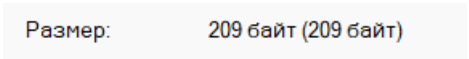
```
        do {
            suffix.insert(0, (char) ('a' + number % 26));
            number /= 26;
        } while (number > 0);
        return suffix.toString();
    }
}
```

Результат работы программы:

За основу берем текстовый файл input.txt

```
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
This is line 6.
This is line 7.
This is line 8.
This is line 9.
This is line 10.
This is line 11.
This is line 12.
```

размер input.txt:



Результат программы при split -b 30 -d input.txt output, оно должно поделить на файлы с размером 30 байт и с цифровым(-d) суффиксом в названии файлов

```
D:\Github_Repositories\spp_po9\reports\Solyshko\2\src>java -cp Task02.jar Task02 split -b 30 -d input.txt output
Created: output00.txt
Created: output01.txt
Created: output02.txt
Created: output03.txt
Created: output04.txt
Created: output05.txt
Created: output06.txt
```

После команды ls видим, что всё разбито согласно условию, по 30 байтов, и в конечный вошло 29 байт (209%30=29)

```
-a----        27.02.2024     19:14            209 input.txt
-a----        27.02.2024     19:57             30 output00.txt
-a----        27.02.2024     19:57             30 output01.txt
-a----        27.02.2024     19:57             30 output02.txt
-a----        27.02.2024     19:57             30 output03.txt
-a----        27.02.2024     19:57             30 output04.txt
-a----        27.02.2024     19:57             30 output05.txt
-a----        27.02.2024     19:57             29 output06.txt
-a----        27.02.2024     15:20           2824 Task01.java
-a----        27.02.2024     19:54             29 Task02.bat
-a----        27.02.2024     19:38           4329 Task02.class
-a----        27.02.2024     19:46           2918 Task02.jar
-a----        27.02.2024     19:13           5083 Task02.java
```

Services Alt+8

Теперь проверим со строками split -l 5 -d input.txt output (поделим по 5 строк)

```
PS D:\Github_Repositories\spp_po9\reports\Solyshko\2\src> D:\Github_Repositories\spp_po9\reports\Solyshko\2\src\Task02.bat split -l 5 -d input.txt output

D:\Github_Repositories\spp_po9\reports\Solyshko\2\src>java -cp Task02.jar Task02 split -l 5 -d input.txt output
Created: output00.txt
Created: output01.txt
Created: output02.txt
```

```
PS D:\Github_Repositories\spp_po9\reports\Solyshko\2\src> cat output00.txt
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
PS D:\Github_Repositories\spp_po9\reports\Solyshko\2\src> cat output01.txt
This is line 6.
This is line 7.
This is line 8.
This is line 9.
This is line 10.
PS D:\Github_Repositories\spp_po9\reports\Solyshko\2\src> cat output02.txt
This is line 11.
This is line 12.
```

Теперь сделаем со стандартного ввода, и создавать будем алфавитные суфиксы

```
D:\Github_Repositories\spp_po9\reports\Solyshko\2\src>java -cp Task02.jar Task02 split -l 8 -
This is line 1.
This is line 1.
This is line 1.
This is line 1.
This is line 1.
This is line 1.
This is line 1.
This is line 1.
Created: xa.txt
This is line 1.
This is line 1.
This is line 1.
This is line 1.
This is line 1.
Created: xb.txt
^CЗавершить выполнение пакетного файла [Y(да)/N(нет)]? y
```

Первый файл:

```
1       This is line 1.
2       This is line 1.
3       This is line 1.
4       This is line 1.
5       This is line 1.
6       This is line 1.
7       This is line 1.
8       This is line 1.
```

Второй файл:

```
1       This is line 1.
2       This is line 1.
3       This is line 1.
4       This is line 1.
5       This is line 1.
```

Ну и последнее поделим на байты со стандартного ввода

```
D:\Github_Repositories\spp_po9\reports\Solyshko\2\src>java -cp Task02.jar Task02 split -b 30b -
This is line 1.
This is line 1.
Created: xa.txt
This is line 1.
This is line 1.
Created: xb.txt
This is line 1.
Created: xc.txt
^CЗавершить выполнение пакетного файла [Y(да)/N(нет)]? y
```

```
-a----         27.02.2024     20:22             30 xa.txt
-a----         27.02.2024     20:23             30 xb.txt
-a----         27.02.2024     20:23             25 xc.txt
```

Все тесты работают корректно.

Вывод: я приобрел базовые навыки работы с файловой системой в Java