МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”
ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ
КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №2

Специальность ПО-9

Выполнил
К. И. Зеленков,
студент группы ПО-9
Проверил
А. А. Крощенко,
ст. преп. кафедры ИИТ,

Брест 2024

Цель работы: научиться разрабатывать простейшие программы на языке программирования Java, получить практический опыт работы с компилятором javac.

Вариант № 6

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.*;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

public class Main1 {
    public static void main(String[] args) throws IOException {
        List<String> lines1, lines2;

        lines1 =
Files.readAllLines(Paths.get("C:\\Users\\kosty\\IdeaProjects\\Lab2\\src\\file1.txt"));
        lines2 =
Files.readAllLines(Paths.get("C:\\Users\\kosty\\IdeaProjects\\Lab2\\src\\file2.txt"));
        if(lines1.equals(lines2)){
            System.out.println("Текстовые файлы идентичны");
        } else {
            for (int i = 0; i < lines1.size(); i++) {
                if(!lines1.get(i).equals(lines2.get(i))){
                    for (int j = 0; j < lines1.get(i).length(); j++) {
                        if(lines1.get(i).charAt(j) != lines2.get(i).charAt(j)){
                            System.out.println("Строка не совпадает.\nСимвол: " + "[" +
lines1.get(i).charAt(j) + "]" +
                                    "\n" + "Позиция: "+ (j+1) + "\nСтрока 1: " + lines1.get(i)
+ "\nСтрока 2: " + lines2.get(i));
                            return;
                        }
                    }
                }
            }
        }
    }
}
```

Пример

1) Содержимое файлов:

```
Как умудряется он успеть
свое промолчать и свое пропеть,
по планете просеменить,
гнев на милость переменить?
```

```
Как умудряется он успеть
свое промолчать и свое пропеть,
по планете просеменить,
гнев на милость переменить?
```

Вывод:

```
Текстовые файлы идентичны
```

2) Содержимое файлов:

```
Как умудряется он успеть
свое промолчать и свое пропеть,
по планете просеменить,
гнев на милость переменить?
```

```
Как умудряется он успеть
свое промолчать и свое пропеть,
по пЩанете просеменить,
гнев на милость переменить?
```

Вывод:

```
Строка не совпадает.
Символ: [Щ]
Позиция: 5
Строка 1: по пЩанете просеменить,
Строка 2: по планете просеменить,
```

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 выходных файлов 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 Main2 {

    private static final String OPTION_BLOCK_SIZE = "-b";
    private static final String OPTION_LINE_COUNT = "-l";
    private static final String OPTION_NUMERIC_SUFFIXES = "-d";

    public static void main(String[] args) {
```

```
        System.out.println(args.length);
        System.out.println(args[0]);
        if (args.length <= 2 || args.length > 6 || !args[0] .equals("split")) {
            System.out.println("Usage: java SplitUtility split [-b num | -l num] [-d]
[input_file] [output_prefix]");
            System.exit(1);
        }

        int memorySize = 0;
        int lineCount = 10;
        boolean numericSuffix = false;
        String inputFile = null;
        String prefix = "x";

        try {
            for (int i = 1; i < args.length; i++) {
                switch (args[i]) {
                    case OPTION_BLOCK_SIZE:
                        memorySize = parseSizeArgument(args[++i]);
                        lineCount = 0;
                        break;
                    case OPTION_LINE_COUNT:
                        lineCount = Integer.parseInt(args[++i]);
                        break;
                    case OPTION_NUMERIC_SUFFIXES:
                        numericSuffix = true;
                        break;
                    default:
                        if (inputFile == null) {
                            inputFile = args[i];
                        } else {
                            prefix = args[i];
                        }
                        break;
                }
            }

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

            split(inputFile, prefix, memorySize, lineCount, numericSuffix);
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            System.out.println("2Usage: java SplitUtility split [-b num | -l num] [-d]
[input_file] [output_prefix]");
            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 split(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.isEmpty()) {
                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();
    }
}
```

Пример
Содержимое и размер файла:

```
input 1
input 2
input 3
input 4
input 5
input 6
input 7
input 8
input 9
input 10
input 11
input 12
input 13
input 14
input 15
input 16
input 17
input 18
input 19
input 20
```

Size: 189 bytes (189 bytes)

1) Цифровые суффиксы + разбиение файлов по 30 байт

```
C:\Users\kosty\IdeaProjects\Lab2>java -cp C:\Users\kosty\IdeaProjects\Lab2\out\artifacts\Lab2_jar\Lab2.jar Main2 split -b 30 -d input.txt output
6
split
Created: output00.txt
Created: output01.txt
Created: output02.txt
Created: output03.txt
Created: output04.txt
Created: output05.txt
Created: output06.txt
```

```
Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         3/26/2024  12:06 PM                .idea
d-----          3/5/2024   1:45 PM                out
d-----         3/26/2024  12:08 PM                src
-a----         3/26/2024  12:06 PM            189 input.txt
-a----         2/11/2024   2:53 PM            433 Lab2.iml
-a----         3/26/2024  12:06 PM             30 output00.txt
-a----         3/26/2024  12:06 PM             30 output01.txt
-a----         3/26/2024  12:06 PM             30 output02.txt
-a----         3/26/2024  12:06 PM             30 output03.txt
-a----         3/26/2024  12:06 PM             30 output04.txt
-a----         3/26/2024  12:06 PM             30 output05.txt
-a----         3/26/2024  12:06 PM              9 output06.txt
```

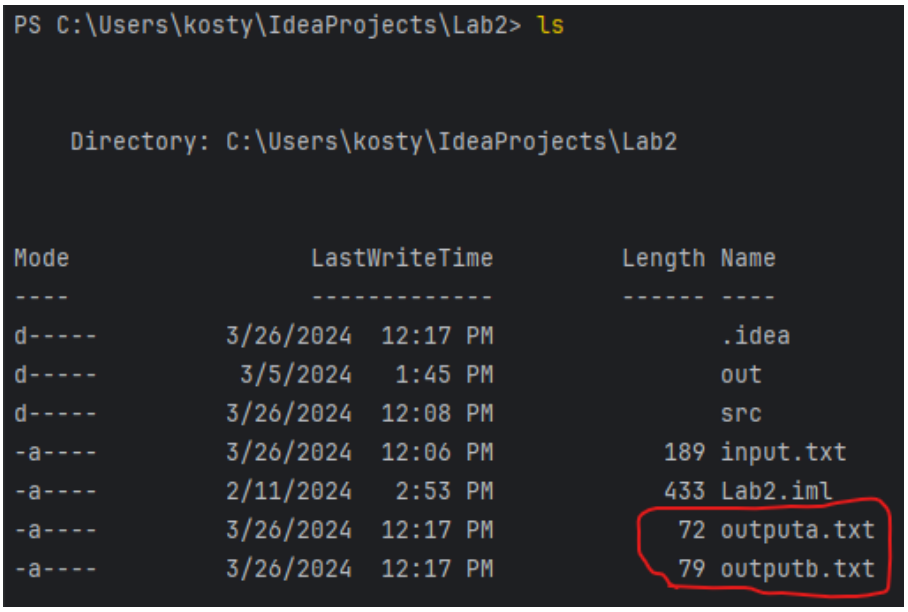
2) Цифровой суффикс + разбиение файлов по 8 строк.

```
C:\Users\kosty\IdeaProjects\Lab2>java -cp C:\Users\kosty\IdeaProjects\Lab2\out\artifacts\Lab2_jar\Lab2.jar Main2 split -l 8 -d input.txt output
6
split
Created: output00.txt
Created: output01.txt
Created: output02.txt
```

```
PS C:\Users\kosty\IdeaProjects\Lab2> cat output00.txt
input 1
input 2
input 3
input 4
input 5
input 6
input 7
input 8
PS C:\Users\kosty\IdeaProjects\Lab2> cat output01.txt
input 9
input 10
input 11
input 12
input 13
input 14
input 15
input 16
PS C:\Users\kosty\IdeaProjects\Lab2> cat output02.txt
input 17
input 18
input 19
input 20
```

3) Алфавитные суффиксы + ручной ввод

```
C:\Users\kosty\IdeaProjects\Lab2>java -cp C:\Users\kosty\IdeaProjects\Lab2\out\artifacts\Lab2_jar\Lab2.jar Main2 split -l 8 -
6
split
input 1
input 2
input 3
input 4
input 5
input 6
input 7
input 8
Created: outputa.txt
input 9
input 10
input 11
input 12
input 13
input 14
input 15
input 16
Created: outputb.txt
^CTerminate batch job (Y/N)? y
PS C:\Users\kosty\IdeaProjects\Lab2>
```



Вывод: научился работать с файловой системой с помощью Java