Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №2

По дисциплине: «Современные платформы программирования»

Выполнил:
студент 3 курса
группы ПО-8
Сорока В.С.

Проверил:
Крощенко А.А.

Брест, 2024

Цель работы: приобрести базовые навыки работы с файловой системой в Java

Задание 1 (Вариант 8)
Напишите программу, считывающую текст построчно и изменяющую порядок следования слов на случайный.
Строки с новым порядком слов выведите на экран.

Спецификации ввода-вывода программы:
- Программа принимает ввод из командной строки.
- Ввод представляет собой текст, в котором каждая строка разделена символом новой строки.
- Пример входных данных:
  This is a sample text.
  Another line of text.
- Программа вызывает функцию randomizeText(String text) с введенным текстом в качестве аргумента.
- Функция randomizeText(String text) разбивает входной текст на строки, используя символ новой строки (\n).
- Для каждой строки во входном тексте:
- Строка разбивается на слова с помощью функции split(" ").
- Слова помещаются в список words в случайном порядке.
- Порядок слов в списке words случайным образом изменяется с помощью алгоритма Кнута — Фишера (Fisher–Yates).
- Отформатированная строка со словами в новом порядке выводится в стандартный вывод.
- Пример вывода:
  is This a text. sample
  of line Another text.
- В случае возникновения ошибки при чтении входного текста, программа выводит сообщение об ошибке в стандартный вывод ошибок.
- После завершения работы функции randomizeText(String text), программа завершается.

Текст программы

```
public static void randomizeText(String text)
{
    String[] lines = text.split("\n"); // Разбиваем текст на строки

    for (String line : lines)
    {
        // Разбиваем строку на слова
        List<String> words = new ArrayList<>(Arrays.asList(line.split(" ")));
        Random random = new Random();

        // Перемешиваем порядок слов
        for (int i = words.size() - 1; i > 0; i--)
        {
            int j = random.nextInt(i + 1);
```

```
            String temp = words.get(i);
            words.set(i, words.get(j));
            words.set(j, temp);
        }
        // Выводим строку с новым порядком слов
        System.out.println(String.join(" ", words));
    }
}
```

```
case 1:
    System.out.println("Введите текст для изменения порядка слов: ");
    input = in.nextLine();
    System.out.println("\nСтроки с новым порядком слов: ");
    randomizeText(input);
    break;
```

Пример работы программы

```
1. Задание 1
2. Задание 2
3. Выход

Выберите необходимое:  1

========================================
Введите текст для изменения порядка слов:
Если бы в Java была настоящая сборка мусора, большинство программ удаляли бы себя при выполнении.

Строки с новым порядком слов:
удаляли сборка бы программ при мусора, была настоящая в Если большинство бы выполнении. Java себя
========================================
```

Задание 2 (Вариант 4)

Утилита nl выводит переданный файл в стандартный вывод или в другой файл, выполняя нумерацию его строк. Если файл не задан или задан как –, читает стандартный ввод.

Формат использования: nl [-i] [-l] [-n] входной_файл [выходной_файл]

• -i ЧИСЛО Задает шаг увеличения номеров строк

• -l 1/0 Задает флаг нумерации пустых строк

• -n ФОРМАТ Использовать заданный формат для номеров строк.

ln – номер выравнен по левому краю, без начальных нулей

rn – номер выровнен по правому краю, без начальных нулей

rz – номер выровнен по правому краю с начальными нулями

Пример использования: nl -i 2 -l 0 -n ln in.txt

Обрабатывает файл in.txt , выводит результат в стандартный вывод, инкремент счетчика равен двум (-i 2), пустые строки не нумеруются.

Спецификации ввода-вывода программы:

- Программа принимает ввод из командной строки в виде аргументов функции runNlCommand(String nlArgs).
- Входные данные представляют собой строку nlArgs, которая разбивается на отдельные аргументы по пробелам.
- Последний аргумент представляет имя входного файла.
- Если последний аргумент равен "-", программа записывает ввод, полученный из стандартного ввода, во временный файл с именем "./src/stdin.txt".
- Входные файлы находятся в директории "./src/" и имеют имя "in.txt".
- Программа обрабатывает аргументы перед последним аргументом для определения настроек:
- Аргумент -i задает значение переменной step, определяющей шаг (инкремент) для нумерации строк.
- Аргумент -l задает значение переменной emptyLineNumbering, определяющей, должны ли пустые строки во входном файле получать номера строк.
- Аргумент -n задает значение переменной numberingFormat, определяющей формат нумерации строк.
- Программа читает входной файл построчно с помощью класса BufferedReader, начиная с первой строки.
- Для каждой непустой строки во входном файле или для всех строк (в зависимости от значения emptyLineNumbering):
- Программа форматирует номер строки с помощью функции formatLineNumber(int number, String format).
- Программа выводит отформатированный номер строки и содержимое строки, разделенные пробелом, в стандартный вывод (System.out).
- Значение number для форматирования номера строки увеличивается на step с каждой итерацией.
- При возникновении ошибки при чтении файла, программа выводит сообщение об ошибке в стандартный вывод ошибок (System.err).
- Функция writeInputToTempFile(String fileName) используется для записи входных данных, полученных из стандартного ввода, во временный файл с именем fileName.
- Функция formatLineNumber(int number, String format) форматирует номер строки согласно указанному формату:
- Если format равен "ln", функция возвращает строковое представление номера number.
- Если format равен "rn", функция возвращает строковое представление номера number без начальных и конечных пробелов.
- Если format равен "rz", функция возвращает строковое представление номера number с ведущими нулями (если необходимо).
- В противном случае, функция возвращает строковое представление номера number.

Текст программы

```
public static void runNlCommand(String nlArgs)
{
    String[] args = nlArgs.split(" ");
    String inputFile = args[args.length - 1];
    String outputFile = "";
    int step = 1;
    boolean emptyLineNumbering = true;
    String numberingFormat = "ln";

    if (inputFile.equals("-"))
    {
        inputFile = "./src/stdin.txt";
        writeInputToTempFile(inputFile);
    }

    // Указываем абсолютный путь к файлу in.txt
    inputFile = "./src/in.txt";
    for (int i = 0; i < args.length - 1; i++)
    {
        switch (args[i])
        {
            case "-i":
                step = Integer.parseInt(args[i + 1]);
                break;
            case "-l":
                emptyLineNumbering = args[i + 1].equals("1");
                break;
            case "-n":
                numberingFormat = args[i + 1];
                break;
        }
    }
    try {
        BufferedReader reader = new BufferedReader(new
FileReader(inputFile));
        String line;
        int lineNumber = 1;

        while ((line = reader.readLine()) != null)
        {
            if (!line.trim().isEmpty() || emptyLineNumbering)
            {
                String formattedLineNumber = formatLineNumber(lineNumber,
numberingFormat);
                System.out.println(formattedLineNumber + " " + line);
                lineNumber += step;
            }
        }
        reader.close();
    } catch (IOException e) {
        System.out.println("Ошибка при чтении файла: " + e.getMessage());
    }
}

public static void writeInputToTempFile(String fileName)
{
    try {
        BufferedWriter writer = new BufferedWriter(new FileWriter(fileName));
        Scanner scanner = new Scanner(System.in);
        String line;

        while (!(line = scanner.nextLine()).isEmpty())
```

```
        {
            writer.write(line);
            writer.newLine();
        }
        writer.close();
        scanner.close();
    } catch (IOException e) {
        System.out.println("Ошибка при записи входных данных: " +
e.getMessage());
    }
}

public static String formatLineNumber(int number, String format)
{
    switch (format)
    {
        case "ln":
            return String.valueOf(number);
        case "rn":
            return String.valueOf(number).trim();
        case "rz":
            return String.format("%02d", number);
        default:
            return String.valueOf(number);
    }
}
```

```
case 2:
    System.out.println("Введите аргументы для команды nl (пример: -i 2 -l 0 -
n ln in.txt): ");
    input = in.nextLine();
    runNlCommand(input);
```

Пример работы программы:

```
1. Задание 1
2. Задание 2
3. Выход

Выберите необходимое:  2

========================================
Введите аргументы для команды nl (пример: -i 2 -l 0 -n ln in.txt):
-l 0
1 This is line 1
2 This is line 2
3 This is line 3
4 This is line 5
========================================
```

Main.java | in.txt

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

```
========================================
Введите аргументы для команды nl (пример: -i 2 -l 0 -n ln in.txt):
-i 5
1 This is line 1
6 This is line 2
11 This is line 3
16
21 This is line 5
========================================
```

Main.java | in.txt

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

Вывод: получил практические базовые навыки работы с файловой системой в языке программирования Java при решении практических задач.