МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №2

Специальность ПО9(з)

Выполнил
Д. Н. Кухарев,
студент группы ПО9

Проверил
А. А. Крощенко,
ст. преп. кафедры ИИТ,
«___k_____2024 г.

Брест 2024

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

Вариант 9

Задание 1. Напишите программу, которая использует генерацию случайных чисел для создания предложений.

Программа должна использовать 4 массива строк, называемые noun (существительные), adjective (прилагательные), verb (глаголы) и preposition (предлоги). Указанные массивы должны считываться из файла.

Программа должна создавать предложение, случайно выбирая слова из каждого массива в следующем порядке: noun, verb, preposition, adjective, noun.

Слова должны быть разделены пробелами. При выводе окончательного предложения, оно должно начинаться с заглавной буквы и заканчиваться точкой. Программа должна генерировать 20 таких предложений.

Выполнение:

Код программы

Main.java:

```
public class Main {
   public static void main(String[] args) {
      String[] types = {"noun", "adjective", "verb", "preposition"};
      String formated_string = Const.CLEAR, output = Const.CLEAR;
      StringDynamicArray[] type_array_navp = new StringDynamicArray[types.length];
      IFile workWith_txt = new IFile();
      try{
         output = args[Const.first_element];
      } catch(Exception ex){
         System.out.println("Arguments are empty, using default output file");
         output = "output";
      }

      workWith_txt.ClearFile(output);

      for(int i = 0; i < types.length; ++i){
         type_array_navp[i] = workWith_txt.ReadStrings(types[i]);
      }
      for(int i = 0; i < Const.sentences_amount; ++i){
         formated_string = type_array_navp[Const.noun].getRandom() + ' '
               + type_array_navp[Const.verb].getRandom() + ' ' + type_array_navp[Const.preposition].getRandom()
+ ' '
               + type_array_navp[Const.adjective].getRandom() + ' ' + type_array_navp[Const.noun].getRandom() +
".\n";

         formated_string = Character.toUpperCase(formated_string.charAt(Const.first_element))
```

```
                + formated_string.substring(Const.lower_case_start);
            workWith_txt.WriteString(formated_string, output);
            formated_string = Const.CLEAR;
        }
    }
}
```

IFile.java:

```
import java.io.BufferedReader;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;

public class IFile {
    static String read_string = "";
    public static StringDynamicArray ReadStrings(String filename){
        StringDynamicArray file_words_array = new StringDynamicArray();
        try(BufferedReader reader = new BufferedReader(new FileReader(Paths.PATH + filename + ".txt"))){
            read_string = reader.readLine();
            if(read_string != null){
                file_words_array.add(read_string);
            }
            while(read_string != null){
                read_string = reader.readLine();
                if(read_string != null){
                    file_words_array.add(read_string);
                }
            }
            reader.close();
        }catch(IOException ex){
            System.out.println(ex.getMessage());
        }
        return file_words_array;
    }
    public static void WriteString(String str, String file_name){
        FileOutputStream fout = null;
        try{
            fout = new FileOutputStream(Paths.PATH + file_name + ".txt", true);
            byte[] buffer = str.getBytes();

            fout.write(buffer, 0, buffer.length);
            System.out.println(str);
        }catch(IOException ex){
            System.out.println(ex.getMessage());
        }finally {
            try{
```

```
            if(fout != null){
                fout.close();
            }
        }catch(IOException ex){
            System.out.println(ex.getMessage());
        }
    }
}
public static void ClearFile(String filename){
    FileOutputStream fout = null;
    try{
        fout = new FileOutputStream(Paths.PATH + filename + ".txt");
        byte[] buffer = Const.CLEAR.getBytes();
        fout.write(buffer, 0, buffer.length);
        System.out.println("The file was cleaned");
    }catch(IOException ex){
        System.out.println(ex.getMessage());
    } finally {
        try{
            if(fout != null){
                fout.close();
            }
        }catch(IOException ex){
            System.out.println(ex.getMessage());
        }
    }
}
}
```

StringDynamicArray.java:

```
public class StringDynamicArray {
    private String[] array;
    private int size;
    private int capacity;
    public StringDynamicArray(){
        capacity = 10;
        array = new String[capacity];
        size = 0;
    }
    public int size(){
        return size;
    }
    public void resize(){
        capacity *= 2;
        String[] newArray = new String[capacity];
        System.arraycopy(array, 0, newArray, 0, size);
```

```
        array = newArray;
    }
    public void add(String str){
        if(size == capacity){
            resize();
        }
        array[size] = str;
        ++size;
    }
    public void remove(int index){
        if(index < 0 || index >= size){
            throw new IndexOutOfBoundsException("Index out of bounds: " + index);
        }
        for(int i = index; i < size-1; ++i){
            array[i] = array[i+1];
        }
        --size;
    }
    public String get(int index){
        if(index < 0 || index >= size){
            throw new IndexOutOfBoundsException("Index out of bounds: " + index);
        }
        return array[index];
    }
    public void show(){
        System.out.print('[');
        for(int i = 0; i < size; ++i){
            if(i < size-1){
                System.out.print(array[i] + ", ");
            }
            else{
                System.out.print(array[i]);
            }
        }
        System.out.print(']'); System.out.println();
    }
    public String getRandom(){
        String result = "";
        result = array[(int) (Math.random()*size)];

        return result;
    }
}
```

Const.java:

```
    public class Const {
```

```
    final static String CLEAR = "";
    final static int first_element = 0;
    final static int lower_case_start = 1;
    final static int noun = 0;
    final static int adjective = 1;
    final static int verb = 2;
    final static int preposition = 3;
    final static int sentences_amount = 20;
}
```

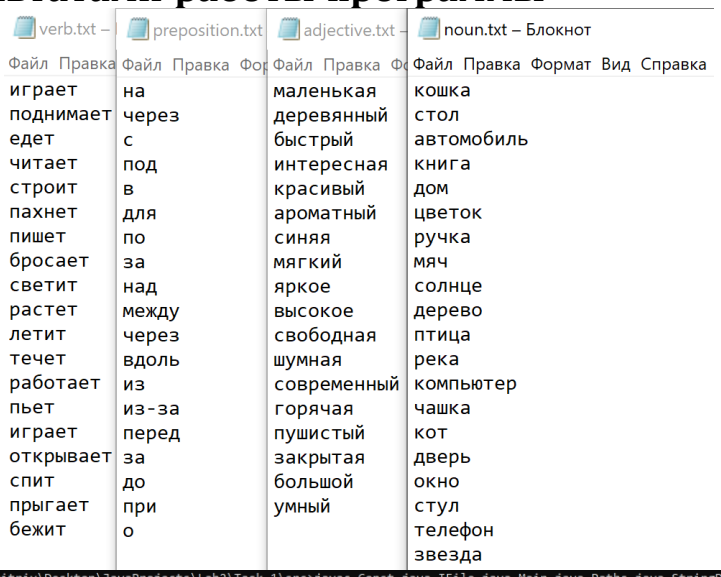
Спецификация ввода

>java Main <число1> <число2> <число3>…<числоN>

Пример

>java Main 1 2 23 123 324 21 2131

Рисунки с результатами работы программы



verb.txt –

Файл Правка
играет
поднимает
едет
читает
строит
пахнет
пишет
бросает
светит
растет
летит
течет
работает
пьет
играет
открывает
спит
прыгает
бежит

preposition.txt

Файл Правка Фор
на
через
с
под
в
для
по
за
над
между
через
вдоль
из
из-за
перед
за
до
при
о

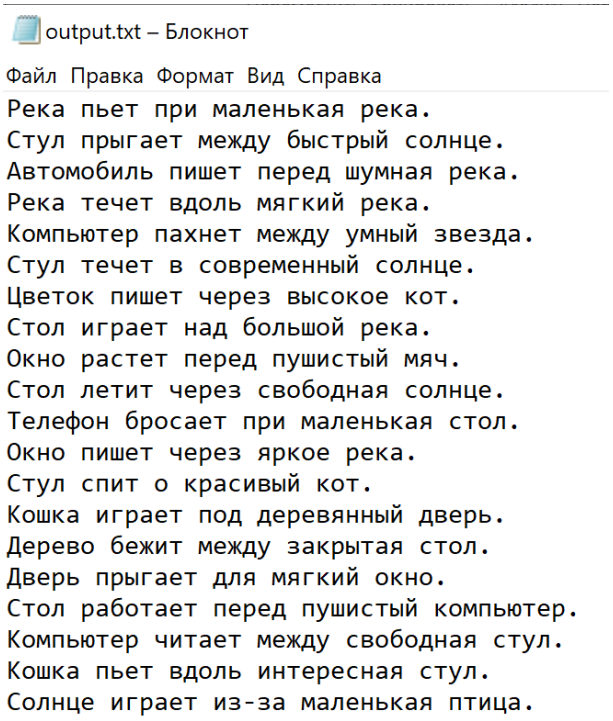
adjective.txt –

Файл Правка Фо
маленькая
деревянный
быстрый
интересная
красивый
ароматный
синяя
мягкий
яркое
высокое
свободная
шумная
современный
горячая
пушистый
закрытая
большой
умный

noun.txt – Блокнот

Файл Правка Формат Вид Справка
кошка
стол
автомобиль
книга
дом
цветок
ручка
мяч
солнце
дерево
птица
река
компьютер
чашка
кот
дверь
окно
стул
телефон
звезда

```
C:\Users\Dmitriy\Desktop\JavaProjects\Lab2\Task_1\src>javac Const.java IFile.java Main.java Paths.java StringDynamicArra
y.java

C:\Users\Dmitriy\Desktop\JavaProjects\Lab2\Task_1\src>java Main
Arguments are empty, using default output file
The file was cleaned
Река пьет при маленькая река.

Стул прыгает между быстрый солнце.

Автомобиль пишет перед шумная река.

Река течет вдоль мягкий река.

Компьютер пахнет между умный звезда.

Стул течет в современный солнце.

Цветок пишет через высокое кот.

Стол играет над большой река.

Окно растет перед пушистый мяч.

Стол летит через свободная солнце.

Телефон бросает при маленькая стол.

Окно пишет через яркое река.

Стул спит о красивый кот.

Кошка играет под деревянный дверь.

Дерево бежит между закрытая стол.

Дверь прыгает для мягкий окно.

Стол работает перед пушистый компьютер.

Компьютер читает между свободная стул.

Кошка пьет вдоль интересная стул.

Солнце играет из-за маленькая птица.
```



Задание 2. Написать консольную утилиту, обрабатывающую ввод пользователя и дополнительные ключи. Проект упаковать в jar-файл, написать bat-файл для запуска.

Утилита join объединяет строки двух упорядоченных текстовых файлов на основе наличия общего поля. По своему функционалу схоже с оператором JOIN , используемого в языке SQL для реляционных баз данных, но оперирует с текстовыми файлами.

Команда join принимает на входе два текстовых файла и некоторое число аргументов. Если не передаются никакие аргументы командной строки, то данная команда ищет пары строк в двух файлах, обладающие совпадающим первым полем (последовательностью символов, отличных от пробела), и выводит строку, состоящую из первого поля и содержимого обоих строк.

Ключами -1 или -2 задаются номера сравниваемых полей для первого и второго файла, соответственно. Если в качестве одного из файлов указано – (но не обоих сразу!), то в этом случае вместо файла считывается стандартный ввод.

Формат использования:

join [-1 номер_поля] [-2 номер_поля] файл1 файл2 [файл3]

Параметры:

- **- 1 fiedl_num Задает номер поля в строке для первого файла, по которому будет выполняться соединение.**
- **- 2 field_num Задает номер поля в строке для второго файла, по которому будет выполняться соединение.**

Аргументы:

- **файл1, файл2 – входные файлы**
- **файл3 – выходной файл, куда записывается результат работы программы.**

Примеры использования:

Пусть задан файл 1.txt со следующим содержимым:
1 abc
2 lmn
3 pqr
и файл 2.txt со следующим содержимым:
1 abc
3 lmn
9 opq
Тогда, выполнение команды join 1.txt 2.txt даст следующий результат:
1 abc abc
3 pqr lmn
Поскольку в обоих файлах есть строки, чьё первое поле совпадает (1, 3), выполнение команды
join -1 2 -2 2 1.txt 2.txt даст результат
abc 1 1
lmn 2 3

поскольку теперь сравнение выполняется по 2-му полю для первого и второго файла соответственно.

Выполнение:

Код программы

Main.java:

```
import java.util.Scanner;


public class Main {
   public static void main(String[] args) {
      GetCommand(args);
   }
   private static void GetCommand(String[] args){
      boolean isValide = false;
      try{
         if(args[Const.FIRST].equals("help")){
            ShowHelp();
         }else if(!args[Const.command].equals("join")){
            throw new IllegalArgumentException("Command '" + args[Const.command] +"' doesn't exist\nUse
'help' to get more info");
         }else{
            isValide = true;
         }
      }catch (IllegalArgumentException ce){
         System.out.println(ce.getMessage());
      }
```

```
        catch (Exception ex){
            System.out.println("Parameters can't be empty\nUse 'help' to get more info");
        } finally {
            if(isValide){
                ParseArgs(args);
            }else{
                System.exit(Const.ALL_PROCESSED);
            }
        }
    }
    private static void ShowHelp(){
        System.out.println("Format: join [-1 field_number] [-2 field_number] file1 file2 [file3]\n" +
                "Parameters:\n\t- 1 fiedl_num Specifies the field number in the line for the first file on which the execution will be performed;\n" +
                "\t- 2 field_num Specifies the field number in the line for the second file on which the execution will be performed;\n" +
                "Arguments:\n\t- file1, file2 – input files\n\t- file3 – output file where the result of the program is written.\n\n" +
                "'exit' to quit the program.");
    }
    private static void ParseArgs(String[] args){
        IntDynamicArray Parameters = new IntDynamicArray();
        StringDynamicArray Arguments = new StringDynamicArray();
        StringDynamicArray[] input_files = new StringDynamicArray[Const.files_amount];
        StringDynamicArray[] columns = new StringDynamicArray[Const.columns_amount];
        DoubleString[] tmp;

        for(int i = Const.INITIALIZE_INT; i < Const.columns_amount; ++i){
            columns[i] = new StringDynamicArray();
        }
        int input_num = Const.NO_RES, column_num = Const.INITIALIZE_INT;
        boolean isRead = false;

        for(int i = Const.parse_position; i < args.length; ++i){
            try{
                if(i < Const.params_amount + Const.next_num){
                    Parameters.add(Integer.parseInt(args[i]));
                }else{
                    throw new NumberFormatException();
                }
            }catch(NumberFormatException nex){
```

```
            Arguments.add(args[i]);
        }
    }
    Parameters = toNormalForm(Parameters);
    try{
        CheckParams(Parameters);
    }catch(IllegalArgumentException ex){
        System.out.println(ex.getMessage());
        System.exit(Const.ALL_PROCESSED);
    }
    try{
        if(Arguments.size() > Const.args_amount){
            throw new IllegalArgumentException("Too much arguments");
        }
    }catch(IllegalArgumentException ex){
        System.out.println(ex.getMessage());
        System.exit(Const.ALL_PROCESSED);
    }

    input_num = Arguments.find(Const.HYPHEN);
    isRead = (input_num != Const.NO_RES);

    for(int i = Const.INITIALIZE_INT; i < Const.files_amount; ++i){
        if(isRead && i == input_num){
            input_files[input_num] = ReadInput();
        }else{
            input_files[i] = ReadFile(Arguments.get(i));
        }
    }
    for(int i = Const.cycle_null; i < Const.files_amount; ++i){
        tmp = new DoubleString[input_files[i].size()];
        tmp = SplitFile(input_files[i]);
        for(int j = Const.INITIALIZE_INT; j < input_files[i].size(); ++j){
            columns[column_num].add(tmp[j].getFirst());
            columns[column_num+Const.next_num].add(tmp[j].getSecond());
        }
        column_num+=Const.columns_in_file;
    }
    MakeOutputFile(Parameters, Arguments, columns);
}
```

```
    private static DoubleString[] SplitFile(StringDynamicArray initial){
        DoubleString[] tmp = new DoubleString[initial.size()];
        for(int i = Const.cycle_null; i < initial.size(); ++i){
            tmp[i] = DivideString(initial.get(i));
        }
        return tmp;
    }
    private static DoubleString DivideString(String str){
        DoubleString parsed = new DoubleString();
        boolean isFirst = true;
        char symb;
        for(int i = Const.cycle_null; i < str.length(); ++i){
            symb = str.charAt(i);
            if(symb != Const.SPACE){
                if(isFirst){
                    parsed.setFirst(parsed.getFirst() + symb);
                }else{
                    parsed.setSecond(parsed.getSecond() + symb);
                }
            }else{
                isFirst = !isFirst;
            }
        }
        return parsed;
    }
    private static StringDynamicArray ReadFile(String file_name){
        if(!findType(file_name)){
            file_name += Const.default_file_type;
        }
        return IFile.ReadStrings(file_name);
    }
    private static void MakeOutputFile(IntDynamicArray params, StringDynamicArray args,
StringDynamicArray[] columns){
        String output_file_name, output_string = Const.CLEAR;
        int max_col_size = Const.INITIALIZE_INT;
        int[] col_fields = new int[params.size()];
        int second_file_col_pos = Const.INITIALIZE_INT;
        col_fields = SetColumnsNums(params);

        for(int i = Const.cycle_null; i < columns.length; ++i){
```

```
            max_col_size = columns[i].size();
            if(columns[i].size() > max_col_size){
                max_col_size = columns[i].size();
            }
        }
        if(args.size() < Const.args_amount){
            output_file_name = Const.default_file_name;
        }else{
            output_file_name = args.get(Const.args_amount+Const.sizeToPos);
        }
        if(!findType(output_file_name)){
            output_file_name += Const.default_file_type;
        }
        IFile.ClearFile(output_file_name);

        for(int i = Const.cycle_null; i < max_col_size; ++i){
            second_file_col_pos =
columns[col_fields[Const.cmp_field2]].find(columns[col_fields[Const.cmp_field1]].get(i));
            if(second_file_col_pos > Const.ISNEG){
                output_string = columns[col_fields[Const.cmp_field1]].get(i) + Const.SPACE
                    + columns[col_fields[Const.field_toWr1]].get(i) + Const.SPACE
                    + columns[col_fields[Const.field_toWr2]].get(second_file_col_pos) + Const.endl;
                IFile.WriteFile(output_string, output_file_name);
            }
        }
    }
    private static IntDynamicArray toNormalForm(IntDynamicArray params){
        IntDynamicArray normal_params = new IntDynamicArray();
        int rep_tmp = Const.INITIALIZE_INT;
        normal_params.add(Const.nparam1);
        normal_params.add(Const.nparam2);
        normal_params.add(Const.nparam3);
        normal_params.add(Const.nparam4);
        if(params.size() == Const.params_amount){
            normal_params = params;
        }
        if(params.size() == Const.ONEARG){
            if(params.get(Const.FIRST) == Const.nparam1){
                normal_params.replace(params.get(Const.FIRST), Const.FIRST);
            }else{
```

```
            normal_params.replace(params.get(Const.FIRST), Const.THIRD);
        }
    } else if(params.size() == Const.TWOARGS){
        if(params.get(Const.FIRST) < Const.ISNEG && params.get(Const.SECOND) < Const.ISNEG){
            normal_params.replace(params.get(Const.FIRST), Const.FIRST);
            normal_params.replace(params.get(Const.SECOND), Const.THIRD);
        }else{
            if(params.get(Const.FIRST) == Const.nparam1){
                normal_params.replace(params.get(Const.FIRST), Const.FIRST);
                normal_params.replace(params.get(Const.SECOND), Const.SECOND);
            }else{
                normal_params.replace(params.get(Const.FIRST), Const.THIRD);
                normal_params.replace(params.get(Const.SECOND), Const.FOURTH);
            }
        }
    }else if(params.size() == Const.THREEARGS){
        if(params.get(Const.SECOND) < Const.ISNEG){
            normal_params.replace(params.get(Const.FIRST), Const.FIRST);
            normal_params.replace(params.get(Const.SECOND), Const.THIRD);
            normal_params.replace(params.get(Const.THIRD), Const.FOURTH);
        }else{
            normal_params.replace(params.get(Const.FIRST), Const.FIRST);
            normal_params.replace(params.get(Const.SECOND), Const.SECOND);
            normal_params.replace(params.get(Const.THIRD), Const.THIRD);
        }
    }
    if(normal_params.get(Const.FIRST) == Const.nparam3){
        rep_tmp = params.get(Const.THIRD);
        normal_params.replace(params.get(Const.FIRST), Const.THIRD);
        normal_params.replace(rep_tmp, Const.FIRST);

        rep_tmp = params.get(Const.FOURTH);
        normal_params.replace(params.get(Const.SECOND), Const.FOURTH);
        normal_params.replace(rep_tmp, Const.SECOND);
    }
    return normal_params;
}
private static StringDynamicArray ReadInput(){
    StringDynamicArray console_stream = new StringDynamicArray();
    console_stream.add(Const.CLEAR);
```

```
    Scanner in = new Scanner(System.in);
    System.out.println("Welcome to the 'Input Reader'\nWrite lines in the next format '[str1] [str2]'\n" +
          "Space is required, line without space will end the input\nEnter:");
    String str = Const.CLEAR;
    do{
       str = in.nextLine();
       if(hasSpace(str)){
          console_stream.add(str);
       }
    }while(hasSpace(str));
    return console_stream;
}
private static boolean hasSpace(String str){
    for(int i = Const.INITIALIZE_INT; i < str.length(); ++i){
       if(str.charAt(i) == ' '){
          return true;
       }
    }
    return false;
}
private static int[] SetColumnsNums(IntDynamicArray params){
    int[] result = new int[params.size()];
    if(params.get(Const.firstLineNum) == Const.nparam2){
       result[Const.cmp_field1] = Const.FIRST;
       result[Const.field_toWr1] = Const.SECOND;
    }else{
       result[Const.cmp_field1] = Const.SECOND;
       result[Const.field_toWr1] = Const.FIRST;
    }
    if(params.get(Const.secondLineNum) == Const.nparam2){
       result[Const.cmp_field2] = Const.THIRD;
       result[Const.field_toWr2] = Const.FOURTH;
    }else{
       result[Const.cmp_field2] = Const.FOURTH;
       result[Const.field_toWr2] = Const.THIRD;
    }

    return result;
}
private static boolean findType(String str){
```

```
        for(int i = Const.INITIALIZE_INT; i < str.length(); ++i){
            if(str.charAt(i) == '.'){
                return true;
            }
        }
        return false;
    }
    private static void CheckParams(IntDynamicArray params){
        if(params.get(Const.FIRST) == params.get(Const.THIRD)){
            throw new IllegalArgumentException("Can't compare same file");
        }else if(params.get(Const.FIRST) > Const.nparam1 || params.get(Const.FIRST) < Const.nparam3 ||
                params.get(Const.THIRD) > Const.nparam1 || params.get(Const.THIRD) < Const.nparam3){
            throw new IllegalArgumentException("Wrong field number");
        } else if(params.get(Const.SECOND) > Const.max_param || params.get(Const.SECOND) <
Const.min_param ||
                params.get(Const.FOURTH) > Const.max_param || params.get(Const.FOURTH) <
Const.min_param){
            throw new IllegalArgumentException("Wrong field number");
        }
    }
}
```

Const.java:

```
public class Const {
    final static char SPACE = ' ';
    final static char endl = '\n';
    final static String CLEAR = "";
    final static String HYPHEN = "-";
    final static String default_file_name = "output.txt";
    final static String input = "input/";
    final static String output = "output/";
    final static String default_file_type = ".txt";
    final static int default_capacity = 10;
    final static int srcPos = 0;
    final static int destPos = 0;
    final static int OFF = 0;
    final static int inc_capacity = 2;
    final static int nparam1 = -1;
    final static int nparam2 = 1;
    final static int nparam3 = -2;
    final static int nparam4 = 1;
```

```
    final static int min_param = 1;
    final static int max_param = 2;
    final static int NO_RES = -1;
    final static int ISNEG = 0;
    final static int INITIALIZE_INT = 0;
    final static int cycle_null = 0;
    final static int next_num = 1;
    final static int size_chng = 1;
    final static int sizeToPos = -1;
    final static int parse_position = 1;
    final static int FIRST = 0;
    final static int SECOND = 1;
    final static int THIRD = 2;
    final static int FOURTH = 3;
    final static int files_amount = 2;
    final static int columns_amount = files_amount*2;
    final static int columns_in_file = 2;
    final static int params_amount = 4;
    final static int args_amount = 3;
    final static int firstLineNum = 1;
    final static int secondLineNum = 3;
    final static int cmp_field1 = 0;
    final static int field_toWr1 = 1;
    final static int cmp_field2 = 2;
    final static int field_toWr2 = 3;

    final static int FILE1 = 0;
    final static int FILE2 = 1;
    final static int file1_1 = 0;
    final static int file1_2 = 1;
    final static int file2_1 = 2;
    final static int file2_2 = 3;
    final static int command = 0;
    final static int NOARGS = 0;
    final static int ONEARG = 1;
    final static int TWOARGS = 2;
    final static int THREEARGS = 3;
    final static int ALL_PROCESSED = 0;
}
```

DoubleString.java:

```
public class DoubleString {
   String first = Const.CLEAR;
   String second = Const.CLEAR;
   public void setFirst(String str){
      first = str;
   }
   public void setSecond(String str){
      second = str;
   }
   public String getFirst(){
      return first;
   }
   public String getSecond(){
      return second;
   }
}
```

IFile.java:

```
import java.io.BufferedReader;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;


public class IFile {
   private static String read_string = "";
   public static StringDynamicArray ReadStrings(String filename){
      StringDynamicArray file_words_array = new StringDynamicArray();
      try(BufferedReader reader = new BufferedReader(new FileReader(Paths.PATH + Const.input +
filename))){
         file_words_array.add(Const.CLEAR);
         read_string = reader.readLine();
         if(read_string != null){
            file_words_array.add(read_string);
         }
         while(read_string != null){
            read_string = reader.readLine();
            if(read_string != null){
               file_words_array.add(read_string);
            }
         }
```

```
    }catch(IOException ex){
        System.out.println(ex.getMessage());
    }
    return file_words_array;
}
public static void WriteFile(String str, String filename){
    FileOutputStream fout = null;
    try{
        fout = new FileOutputStream(Paths.PATH + Const.output + filename, true);
        byte[] buffer = str.getBytes();
        fout.write(buffer, Const.OFF, buffer.length);
        System.out.println("The file was updated");
    }catch(IOException ex){
        System.out.println(ex.getMessage());
    } finally {
        try{
            if(fout != null){
                fout.close();
            }
        }catch(IOException ex){
            System.out.println(ex.getMessage());
        }
    }
}
public static void ClearFile(String filename){
    FileOutputStream fout = null;
    try{
        fout = new FileOutputStream(Paths.PATH + Const.output + filename);
        byte[] buffer = Const.CLEAR.getBytes();
        fout.write(buffer, Const.OFF, buffer.length);
        System.out.println("The file was cleaned");
    }catch(IOException ex){
        System.out.println(ex.getMessage());
    } finally {
        try{
            if(fout != null){
                fout.close();
            }
        }catch(IOException ex){
            System.out.println(ex.getMessage());
```

```
            }
        }
    }
}
```

IntDynamicArray.java:

```
public class IntDynamicArray {
    private int[] array;
    private int size;
    private int capacity;
    public IntDynamicArray(){
        capacity = Const.default_capacity;
        array = new int[capacity];
        size = Const.INITIALIZE_INT;
    }
    public int size(){
        return size;
    }
    public boolean find(int number){
        for(int i = Const.ISNEG; i < size; ++i){
            if(array[i] == number){
                return true;
            }
        }
        return false;
    }
    public void resize(){
        capacity *= Const.inc_capacity;
        int[] newArray = new int[capacity];
        System.arraycopy(array, Const.srcPos, newArray, Const.destPos, size);
        array = newArray;
    }
    public void add(int number){
        if(size == capacity){
            resize();
        }
        array[size] = number;
        ++size;
    }
    public int replace(int number, int index){
        if(index < size){
```

```
            array[index] = number;
            return Const.ALL_PROCESSED;
        }else{
            return Const.NO_RES;
        }
    }
    public void remove(int index){
        if(index < Const.ISNEG || index >= size){
            throw new IndexOutOfBoundsException("Index out of bounds: " + index);
        }
        for(int i = index; i < size-Const.size_chng; ++i){
            array[i] = array[i+Const.next_num];
        }
        --size;
    }
    public int get(int index){
        if(index < Const.ISNEG || index >= size){
            throw new IndexOutOfBoundsException("Index out of bounds: " + index);
        }
        return array[index];
    }
    public void show(){
        System.out.print('[');
        for(int i = Const.ISNEG; i < size; ++i){
            if(i < size-Const.size_chng){
                System.out.print(array[i] + ", ");
            }
            else{
                System.out.print(array[i]);
            }
        }
        System.out.print(']'); System.out.println();
    }
}
```

StringDynamicArray.java:

```
public class StringDynamicArray {
    private String[] array;
    private int size;
    private int capacity;
    public StringDynamicArray(){
```

```
    capacity = Const.default_capacity;
    array = new String[capacity];
    size = Const.INITIALIZE_INT;
}
public int size(){
    return size;
}
public void resize(){
    capacity *= Const.inc_capacity;
    String[] newArray = new String[capacity];
    System.arraycopy(array, Const.srcPos, newArray, Const.destPos, size);
    array = newArray;
}
public void add(String str){
    if(size == capacity){
        resize();
    }
    array[size] = str;
    ++size;
}
public void remove(int index){
    if(index < Const.ISNEG || index >= size){
        throw new IndexOutOfBoundsException("Index out of bounds: " + index);
    }
    for(int i = index; i < size-Const.size_chng; ++i){
        array[i] = array[i+Const.next_num];
    }
    --size;
}
public String get(int index){
    if(index < Const.ISNEG || index >= size){
        throw new IndexOutOfBoundsException("Index out of bounds: " + index);
    }
    return array[index];
}
public int find(String cmp){
    for(int i = Const.INITIALIZE_INT; i < size; ++i){
        if(array[i].equals(cmp)){
            return i;
        }
```

```
        }
        return Const.NO_RES;
    }
    public void show(){
        System.out.print('[');
        for(int i = Const.INITIALIZE_INT; i < size; ++i){
            if(i < size-Const.size_chng){
                System.out.print(array[i] + ", ");
            }
            else{
                System.out.print(array[i]);
            }
        }
        System.out.print(']'); System.out.println();
    }
    public String getRandom(){
        String result = "";
        result = array[(int) (Math.random()*size)];

        return result;
    }
}
```

run.bat:

```
@echo off
chcp 65001

setlocal enabledelayedexpansion

set "command="
set "exitCommand=exit"

:main
set /p "command= "

if "%command%"=="%exitCommand%" (
    goto :eof
)

java -jar Task2.jar !command!
```

```
goto :main
```

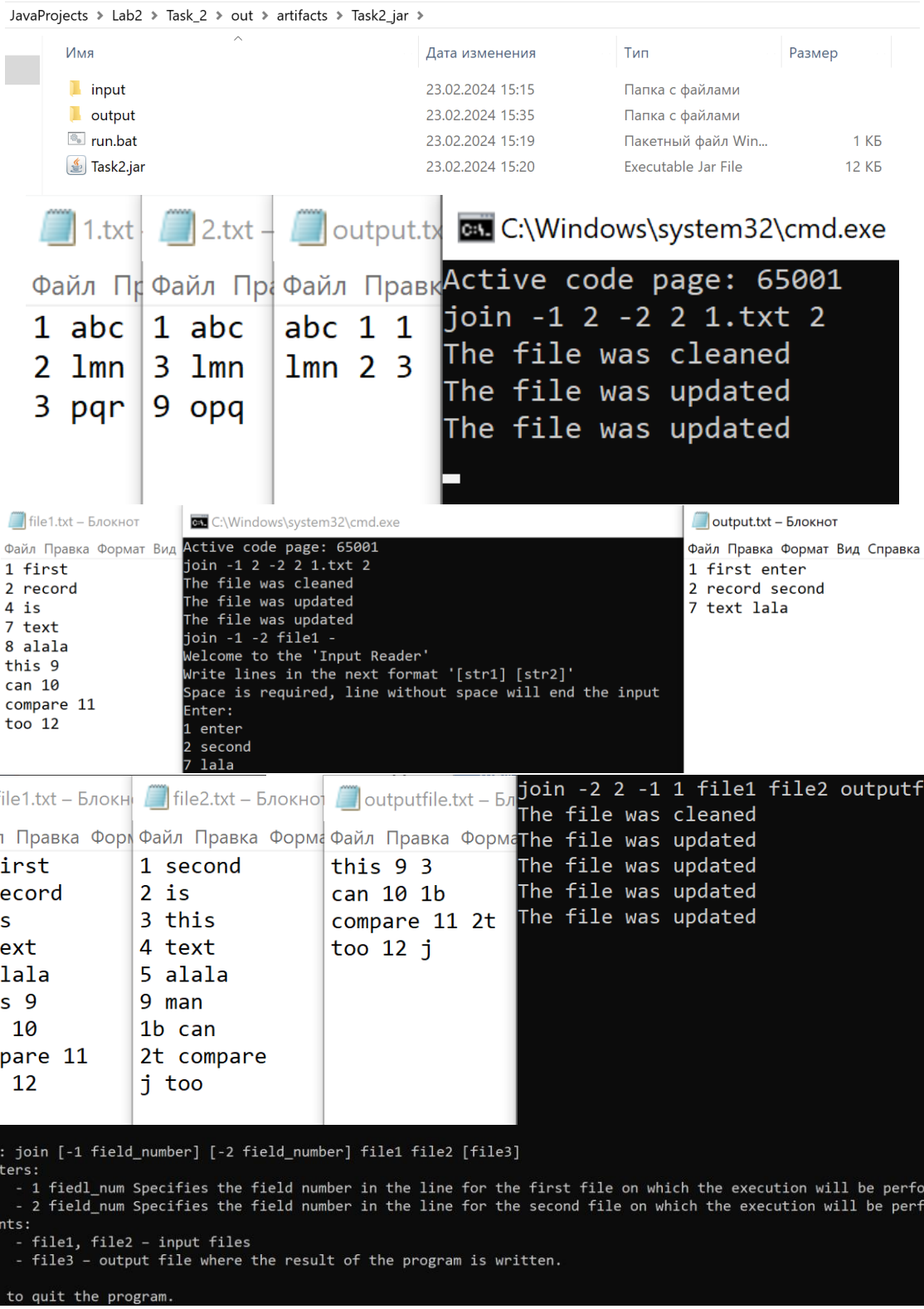
Спецификация ввода

> join [-1 номер_поля] [-2 номер_поля] файл1 файл2 [файл3]

Пример

>join -1 2 -1 2 file1.txt file2 output

Рисунки с результатами работы программы



Вывод: приобрели базовые навыки работы с файловой системой в Java.