МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

ЛАБОРАТОРНА РОБОТА №6

З дисципліни «Програмування на основі Java технологій»

Тема: «Система введення-виведення у мові Java»

Виконав студент групи ІПЗ-21-2
Губарєв Р.В.

Перевірив викладач
Котов І.А.
Гриценко А.М.
Карабут Н.О.

Кривий Ріг
2022

1. Основні відомості про консольну систему вводу-виводу в мові Java

У мові програмування Java ввід/вивід інформації базується на понятті потоку. Потік – це абстрактне поняття, яке символізує джерело або приймач даних, що може передавати або отримувати деяку інформацію. Будь-який потік приховує операції над даними, що виконуються на нижчих рівнях безпосередньо в пристроях вводу/виводу.

Відповідно до призначення потоків, класифікуються і класи в мові Java. Одні класи реалізують операції вводу, інші реалізують операції виводу. Щоб використовувати класи потоків вводу/виводу потрібно імпортувати пакет java.io

У мові Java розрізняють два види потоків:

- *байтові потоки*. Це аналог потоків двійкових даних, які дозволяють компактно зберігати інформацію;
- *символьні потоки*. Це потоки, що представлені зручним способом (для людей) кодування інформації у вигляді зрозумілих текстових символів. У багатьох мовах програмування символьні потоки асоціюються з текстовим форматом представлення інформації.

Класи, що реалізують байтові потоки вводу успадковані від абстрактного класу InputStream:

- **InputStream** – абстрактний клас, що описує потік вводу. Даний клас є базовим для всіх інших класів системи вводу;
- **BufferedInputStream** – клас, що описує буферизований потік вводу;
- **ByteArrayInputStream** – клас, що описує потік вводу, який читає байти з масиву;
- **DataInputStream** –клас, що реалізує методи для читання даних стандартних типів, визначених у Java (int, double, float і т.д.);
- **FileInputStream** – клас, що реалізує потік вводу, який читає дані з файлу;
- **FilterInputStream** – це є реалізація абстрактного класу InputStream;
- **ObjectInputStream** – клас, що реалізує потік вводу об’єктів;
- **PipedInputStream** – клас, що відповідає каналу вводу;
- **PushbackInputStream** – клас, що відповідає потоку вводу, який підтримує повернення одного байту назад в потік вводу;
- **SequenceInputStream** – клас, що реалізує потік вводу, який складається з двох або більше потоків вводу, дані з яких читаються по черзі.

Класи, що реалізують байтові потоки виводу успадковані від абстрактного класу OutputStream:

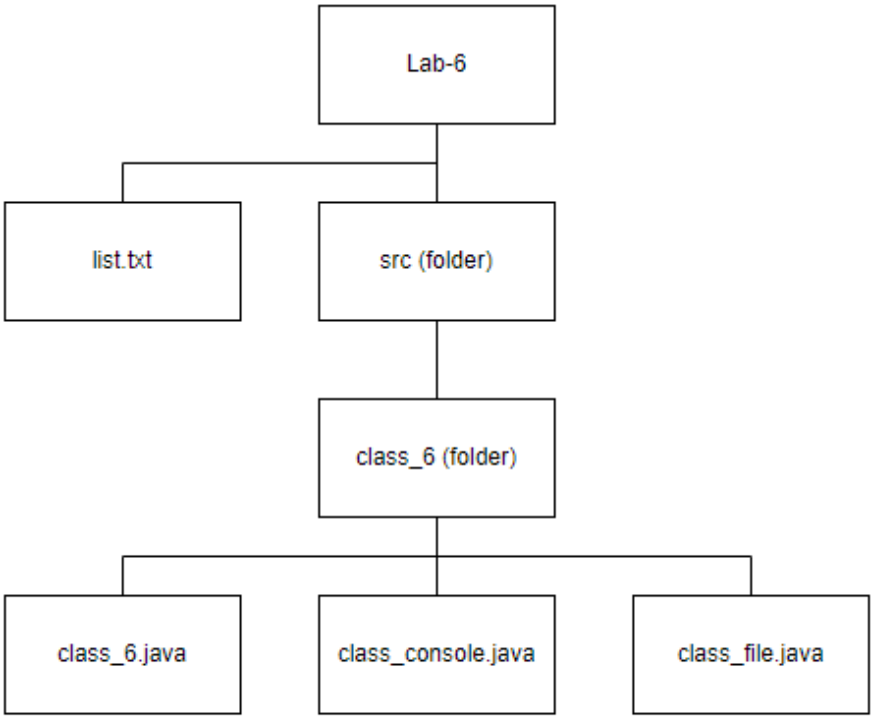
- **OutputStream** – абстрактний клас, що описує потік виводу. Усі інші класи системи виводу є підкласами класу OutputStream;
- **BufferedOutputStream** – клас, що імплементує буферизований потік виводу;

- **ByteArrayOutputStream** – клас, що реалізує потік виводу, який записує байти в масив;
- **DataOutputStream** – клас, що реалізує потік виводу, який містить методи для читання даних стандартних типів, визначених у Java (int, float, double тощо);
- **FileOutputStream** – клас, що відповідає потоку виводу, який записує дані у файл;
- **FilterOutputStream** – клас, який реалізує абстрактний клас OutputStream;
- **ObjectOutputStream** – клас, що відповідає потоку виводу об’єктів;
- **PipedOutputStream** – клас, що асоціюється з каналом виводу;
- **PrintStream** – клас, що представляє собою потік виводу, який містить методи print() та println().

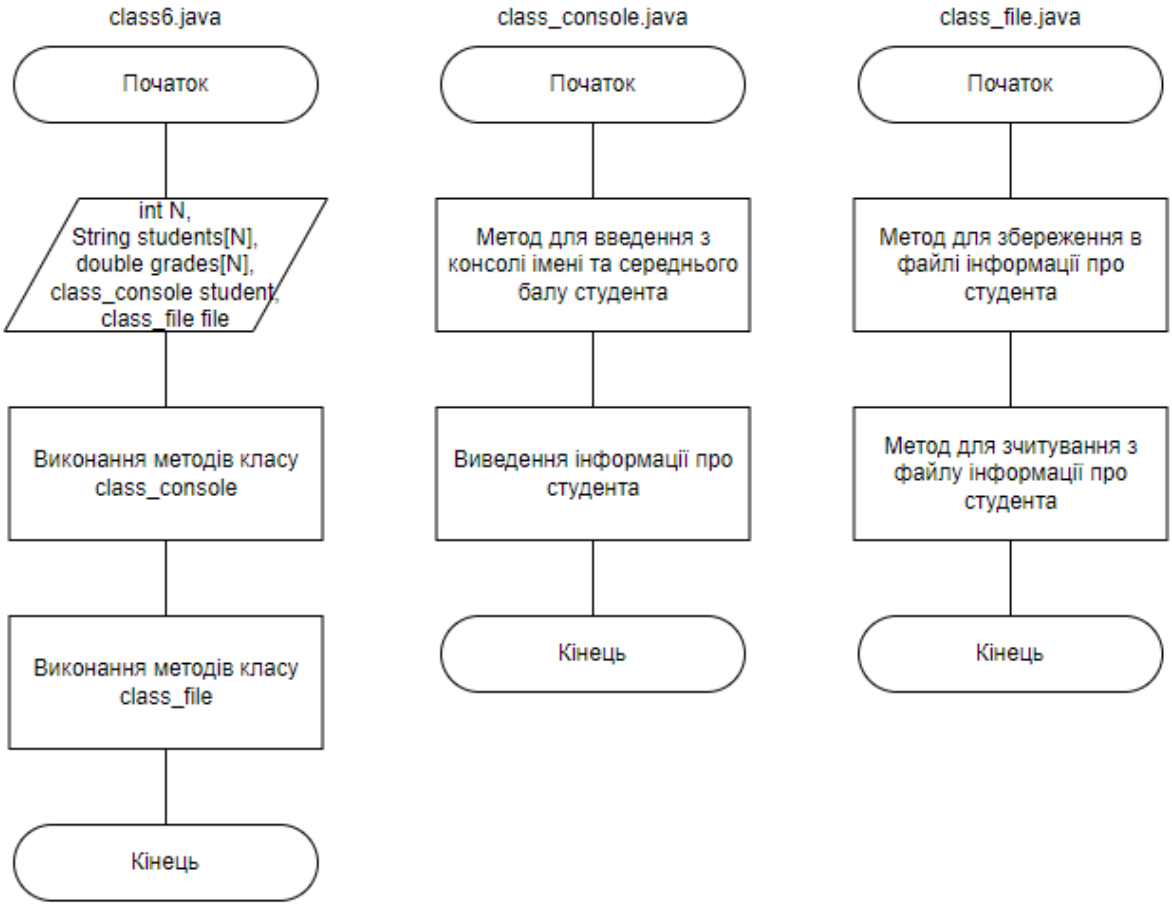
2. Основні відомості про файлову систему вводу-виводу у мові Java

Читання вмісту файлу по байтах не дуже хороша ідея, якщо файл доволі великий. Адже це зайве навантаження на обчислювальні ресурси комп'ютера. Тому більш кращим варіантом є читання тексту цілими блоками. Наприклад, рядками. Рядки у файлах прийнято завершувати символом нового рядка("\n") та символом переходу на новий рядок("\r"). Може бути присутній як один з цих символів так і обидва ("\r\n"), в залежності від того хто і яким чином створював файл.
Читання блоків файлу відбувається через так звані буферизовані потоки, що працюють через буфер в пам'яті комп'ютера. При читанні даних з файлу, дані передаються в програму коли буфер буде порожнім, при записі у файл буфер спочатку повинен заповнитись. Для буферизованого вводу/виводу існує чотири класи. Для буферизованих байтових потоків: BufferedInputStream та BufferedOutputStream. Для буферизованих символьних потоків: BufferedReader та BufferedWriter. Інколи корисно вивільнити дані з буфера до повного його заповнення у окремих критичних точках,це можна зробити з допомогою методу flush.

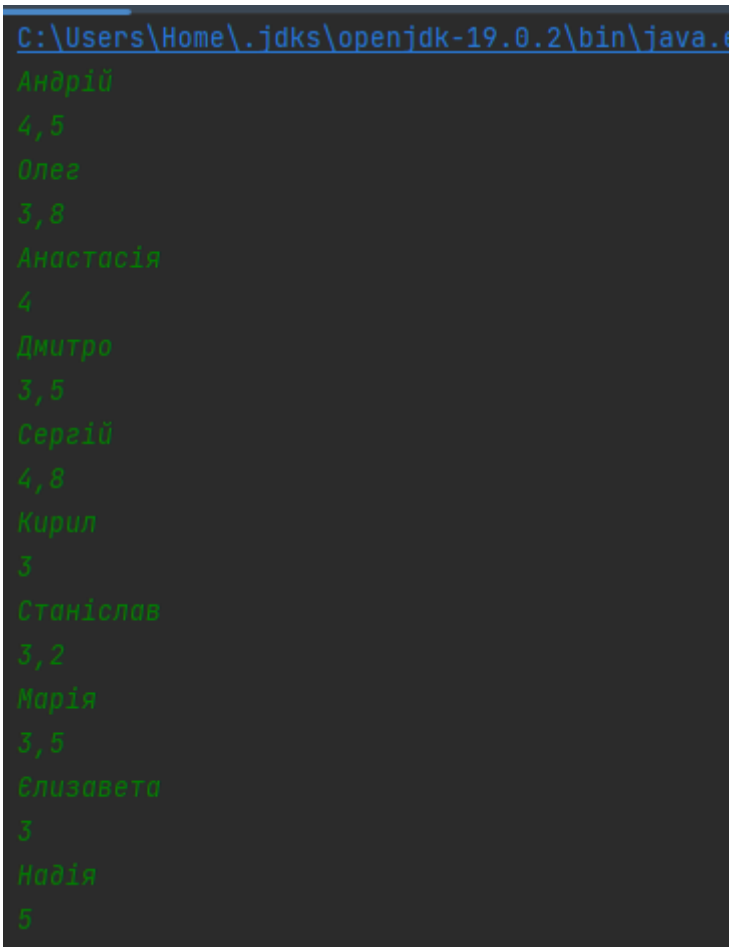
3. Розгорнуту структуру програмного проекту у вигляді деревоподібної схеми



4. Блок-схеми алгоритмів роботи методів класів



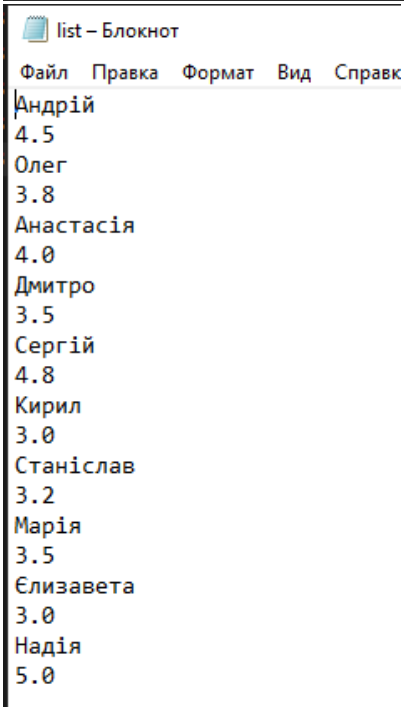
5. Скріншот екрану програми з результатом роботи програми



Студент	Середній бал
Андрій	4.5
Олег	3.8
Анастасія	4.0
Дмитро	3.5
Сергій	4.8
Кирил	3.0
Станіслав	3.2
Марія	3.5
Єлизавета	3.0
Надія	5.0

```
Файл існує
Повний шлях до файлу: H:\University\2 курс\2 семестр\Java програмування\Код\Lab-6\list.txt
Розмір файлу: 178 байт

Process finished with exit code 0
```



6. Текст вихідних кодів програми

class6.java

```
package class6;
import java.io.*;
public class class6 {
    public static void main(String[] args) {
        int N = 10;
        String[] students = new String[N];
        double[] grades = new double[N];

        class_console student = new class_console();
        student.getStudent(students, grades, N);
        student.printStudent(students, grades, N);
        System.out.println("\n");

        class_file file = new class_file();
        file.writeFile(students, grades, N);
        file.readFile(students, grades, N);
        student.printStudent(students, grades, N);

        String filePath = "list.txt";
        file_info info = new file_info(filePath);
        if (info.checkFileExists()) {
            System.out.println("Файл існує");
        } else {
            System.out.println("Файл не знайдено");
        }

        info.path_finder();
        info.size_calculator();
    }
}
```

class_console.java

```
package class6;

import java.io.InputStream;
import java.util.Scanner;

public class class_console {
    public void getStudent(String[] student, double[] grade, int N){
        Scanner in = new Scanner(System.in);

        for (int i = 0; i < N; i++){
            System.out.println("Введіть ім'я студента: ");
            student[i] = in.next();
            System.out.println("Введіть середній бал студента: ");
            grade[i] = in.nextDouble();
        }
    }
    public void printStudent(String[] student, double[] grade, int N){
        int x = 27;
        System.out.println("- - - - - - - - - - - - - - - - - - -");
        System.out.println("|    Студент    |" + "  Середній бал   |");
        for (int i = 0; i < N; i++){
            x = x - student[i].length();
            System.out.println("- - - - - - - - - - - - - - - - - - -");
            System.out.printf("|");
            System.out.printf("%-15s", student[i]);
            System.out.println("|        " + grade[i] + "        |");
        }
        System.out.println("- - - - - - - - - - - - - - - - - - -");
```

```
        }
}
```

class_file.java

```java
package class6;
import java.io.*;
public class class_file {
    public void writeFile(String[] student, double[] grade, int N){
        try {
            FileWriter writer = new FileWriter("list.txt", false);
            for (int i = 0; i < N; i++){
                writer.append(student[i]);
                writer.append('\n');
                writer.append(grade[i]+"\n");
            }
            writer.flush();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
    public void readFile(String[] student, double[] grade, int N){
        try(FileReader r = new FileReader("list.txt"))
        {
            BufferedReader reader = new BufferedReader(r);
            String line = reader.readLine();
            student[0] = line;
            int i = 1;
            int j = 0;

            while (j < N) {
                if (i % 2 == 0){
                    line = reader.readLine();
                    student[j] = line;
                    i++;
                }
                else {
                    line = reader.readLine();
                    grade[j] = Double.parseDouble(line);
                    i++;
                    j++;
                }
            }
        }
        catch(IOException ex){
            System.out.println(ex.getMessage());
        }
    }
}
```

file_info.java

```java
package class6;
import java.io.File;

public class file_info {
    private String filePath;

    public file_info(String filePath) {
        this.filePath = filePath;
    }

    public boolean checkFileExists() {
        File file = new File(filePath);
        return file.exists() && file.isFile();
    }
    public void path_finder(){
        File file = new File(filePath);
        if (file.exists()) {
            String absolutePath = file.getAbsolutePath();
            System.out.println("Повний шлях до файлу: " + absolutePath);
        }
        else {
            System.out.println("Файл не знайдено");
        }
    }
    public void size_calculator(){
        File file = new File(filePath);
        if (file.exists()) {
            long fileSizeBytes = file.length();
            System.out.println("Розмір файлу: " + fileSizeBytes + "
байт");
        } else {
            System.out.println("Файл не знайдено");
        }
    }

}
```

7. Висновки

В цій лабораторній роботі я дізнався про систему введення і виведення у мові Java, а також познайомився з пакетом Java.io.

8. Список використаних джерел

- https://uk.wikibooks.org/wiki/%D0%9E%D1%81%D0%B2%D0%BE%D1%8E%D1%94%D0%BC%D0%BE_Java/%D0%9F%D0%BE%D1%82%D0%BE%D0%BA%D0%B8_%D0%B2%D0%B2%D0%BE%D0%B4%D1%83-%D0%B2%D0%B8%D0%B2%D0%BE%D0%B4%D1%83