Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

до лабораторної роботи №5

З дисципліни: «Кросплатформні засоби програмування»

На тему: «ФАЙЛИ У JAVA»

**Варіант 3**

Виконав: ст. гр. КІ-305

Гнідець В.М.

Прийняв:

Іванов Ю.С.

Львів 2023

**Мета роботи:** оволодіти навиками використання засобів мови Java для роботи з потоками і файлами.

**Завдання:**

1.Створити клас, що реалізує методи читання/запису у текстовому і двійковому форматах результатів роботи класу, що розроблений у лабораторній роботі №4. Написати програму для тестування коректності роботи розробленого класу.

2. Для розробленої програми згенерувати документацію.

3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

5. Дати відповідь на контрольні запитання.

**Варіант 3:** *y=sin(x)/cos(x)*

**Код програми**

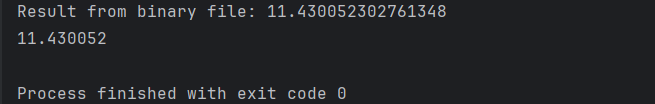
**Class EquationsWithIO**

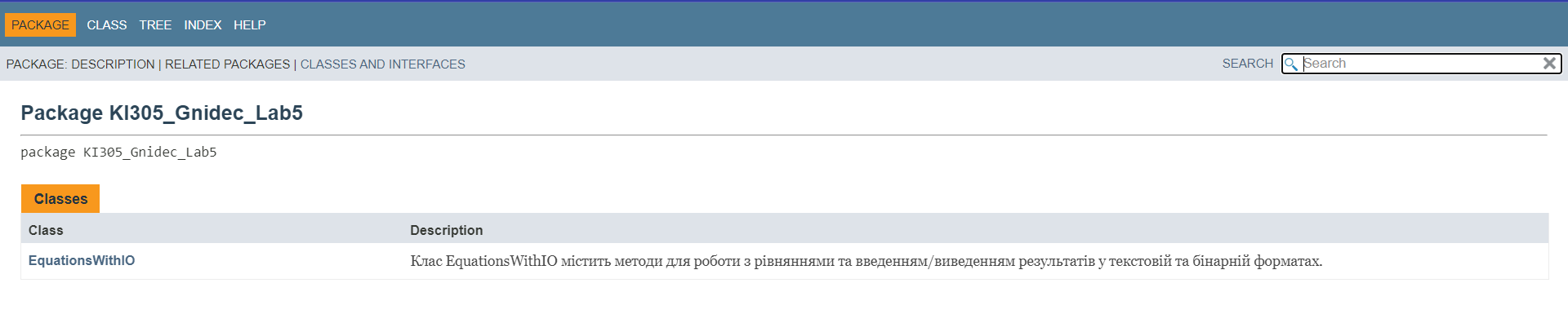
package KI305\_Gnidec\_Lab5;  
  
import java.io.\*;  
import java.io.RandomAccessFile;  
  
import java.util.Scanner;  
*/\*\*  
 \* Клас EquationsWithIO містить методи для роботи з рівняннями та  
 введенням/виведенням результатів у текстовій та бінарній форматах.  
 \*/*public class EquationsWithIO {  
 */\*\*  
 \* Записує результат обчислення рівняння у текстовий файл.  
 \*  
 \* @param fName ім'я файлу, у який записується результат  
 \* @param x значення для обчислення рівняння  
 \* @throws FileNotFoundException виникає, якщо файл не  
 знайдено  
 \*/* public void writeResultTxt(String fName, int x) throws FileNotFoundException {  
 PrintWriter f = new PrintWriter(fName);  
 Equations eq = new Equations();  
 try {  
 double result = eq.calculate(x);  
 f.printf("%f", result);  
 } catch (CalcException ex) {  
 f.print(ex.getMessage());  
 }  
 f.close();  
 }  
 */\*\*  
 \* Записує результат обчислення рівняння у бінарний файл.  
 \*  
 \* @param fName ім'я файлу, у який записується результат  
 \* @param x значення для обчислення рівняння  
 \* @throws IOException виникає, якщо сталася помилка  
 вводу/виводу  
 \*/* public void writeResultBin(String fName, int x) throws IOException {  
 Equations eq = new Equations();  
 try (DataOutputStream f = new DataOutputStream(new FileOutputStream(fName))) {  
 double result = eq.calculate(x);  
 f.writeDouble(result);  
 } catch (CalcException ex) {  
 System.*out*.println("Exception reason: " +  
 ex.getMessage());  
 }  
 }  
 */\*\*  
 \* Читає результат обчислення рівняння з бінарного файлу та  
 виводить його на консоль.  
 \*  
 \* @param fName ім'я файлу, з якого читається результат  
 \* @throws IOException виникає, якщо сталася помилка  
 вводу/виводу  
 \*/* public void readResultBin(String fName) throws IOException {  
 try (DataInputStream f = new DataInputStream(new FileInputStream(fName))) {  
 double result = f.readDouble();  
 System.*out*.println("Result from binary file: " + result);  
 }  
 }  
 */\*\*  
 \* Читає результат обчислення рівняння з текстового файлу та  
 виводить його на консоль.  
 \*  
 \* @param fName ім'я файлу, з якого читається результат  
 \* @throws FileNotFoundException виникає, якщо файл не  
 знайдено  
 \*/* public void readResultTxt(String fName) throws FileNotFoundException {  
 File file = new File(fName);  
 if (file.exists()) {  
 try (Scanner scanner = new Scanner(file)) {  
 if (scanner.hasNextDouble()) {  
 double result = scanner.nextDouble();  
 System.*out*.println("Result from text file: " + result);  
 } else {  
 System.*out*.println(scanner.nextLine());  
 }  
 }  
 } else {  
 throw new FileNotFoundException("File not found: " + fName);  
 }  
 }  
 */\*\*  
 \* Точка входу у програму. Демонструє використання методів  
 класу EquationsWithIO.  
 \*  
 \* @param args аргументи командного рядка  
 \*/* public static void main(String[] args) throws IOException {  
  
 RandomAccessFile raf = null;  
 try {  
 raf = new RandomAccessFile("C:\\Users\\admin\\Desktop\\Size1.txt", "r");  
 } catch (FileNotFoundException e) {  
 throw new RuntimeException(e);  
 }  
  
 raf.seek(0);  
  
 int read = raf.read();  
 StringBuilder contents = new StringBuilder();  
 contents.append((char) read); // Convert the byte to a character  
  
 System.*out*.println("Вміст: " + contents);  
  
 //System.out.print("Розмір масиву: " + new String (bytes) + "\n");  
  
 raf.close();  
  
  
  
 EquationsWithIO equationsWithIO = new EquationsWithIO();  
 try {  
 // Запис результату у текстовий файл  
  
 equationsWithIO.writeResultTxt("result.txt", 85);  
 // Запис результату у бінарний файл  
 equationsWithIO.writeResultBin("result.bin", 85);  
 // Читання результату з бінарного файлу  
 equationsWithIO.readResultBin("result.bin");  
 // Читання результату з текстового файлу  
 equationsWithIO.readResultTxt("result.txt");  
 } catch (IOException e) {  
 System.*out*.println("IO Exception: " + e.getMessage());  
 }  
 }  
}

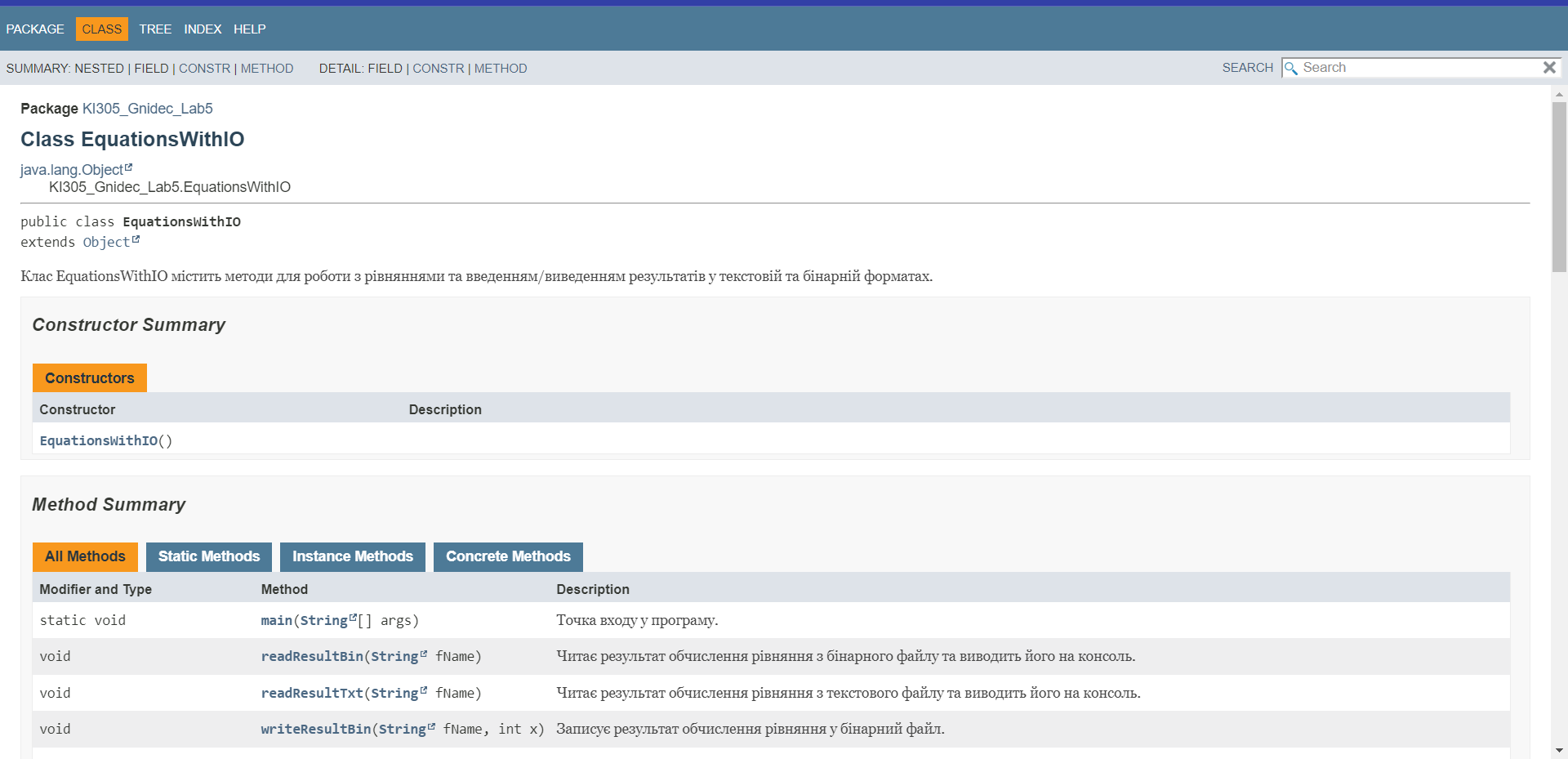
**Class CalcExeption**

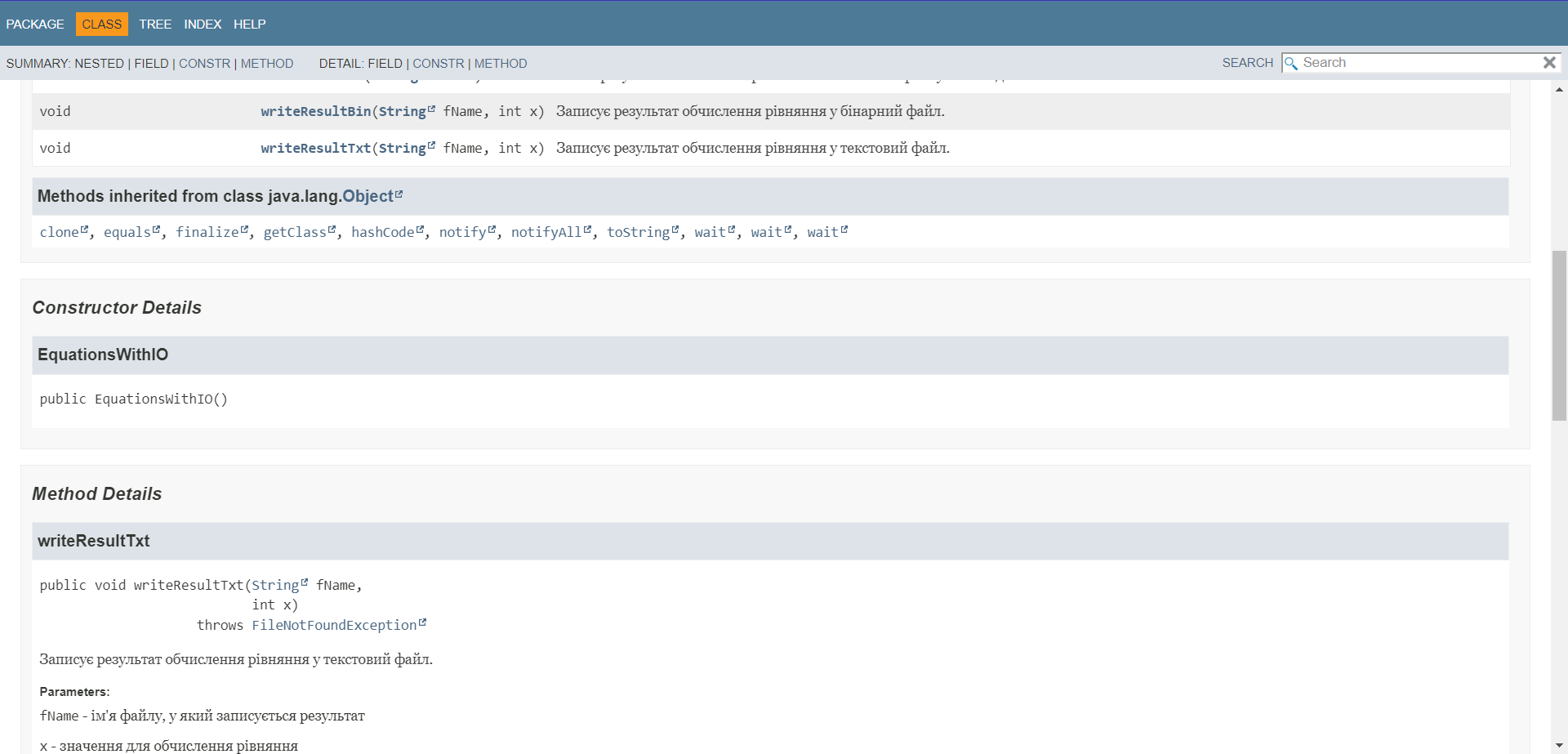
package KI305\_Gnidec\_Lab5;  
  
*/\*\*  
 \* Клас CalcException є підкласом ArithmeticException та  
 використовується для представлення винятків, пов'язаних з  
 обчисленнями.  
 \*/*class CalcException extends ArithmeticException {  
 */\*\*  
 \* Конструктор за замовчуванням.  
 \*/* public CalcException() {}  
 */\*\*  
 \* Конструктор, який приймає рядок, який вказує на причину  
 винятку.  
 \*  
 \* @param cause рядок, який містить причину винятку  
 \*/* public CalcException(String cause) {  
 super(cause);  
 }  
}  
*/\*\*  
 \* Клас Equations містить метод calculate для обчислення значення  
 рівняння.  
 \*/*class Equations {  
 */\*\*  
 \* Обчислює значення рівняння для заданого кута.  
 \*  
 \* @param x кут, для якого обчислюється рівняння  
 \* @return результат обчислення рівняння  
 \* @throws CalcException виникає, якщо сталася помилка під час  
 обчислення рівняння  
 \*/* public double calculate(int x) throws CalcException {  
 double y, rad;  
 rad = x \* Math.*PI* / 180.0;  
 try {  
 y = Math.*sin*(rad) / Math.*cos*(rad);  
 if (Double.*isNaN*(y) || Double.*isInfinite*(y) || x == 90  
 || x == -90) {  
 throw new ArithmeticException();  
 }  
 } catch (ArithmeticException ex) {  
 if (rad == Math.*PI* / 2.0 || rad == -Math.*PI* / 2.0) {  
 throw new CalcException("Exception reason: Illegal value of X for tangent calculation");  
 } else if (x == 0) {  
 throw new CalcException("Exception reason: X = 0");  
 } else {  
 throw new CalcException("Unknown reason of the exception during expression calculation");  
 }  
 }  
 return y;  
 }  
}

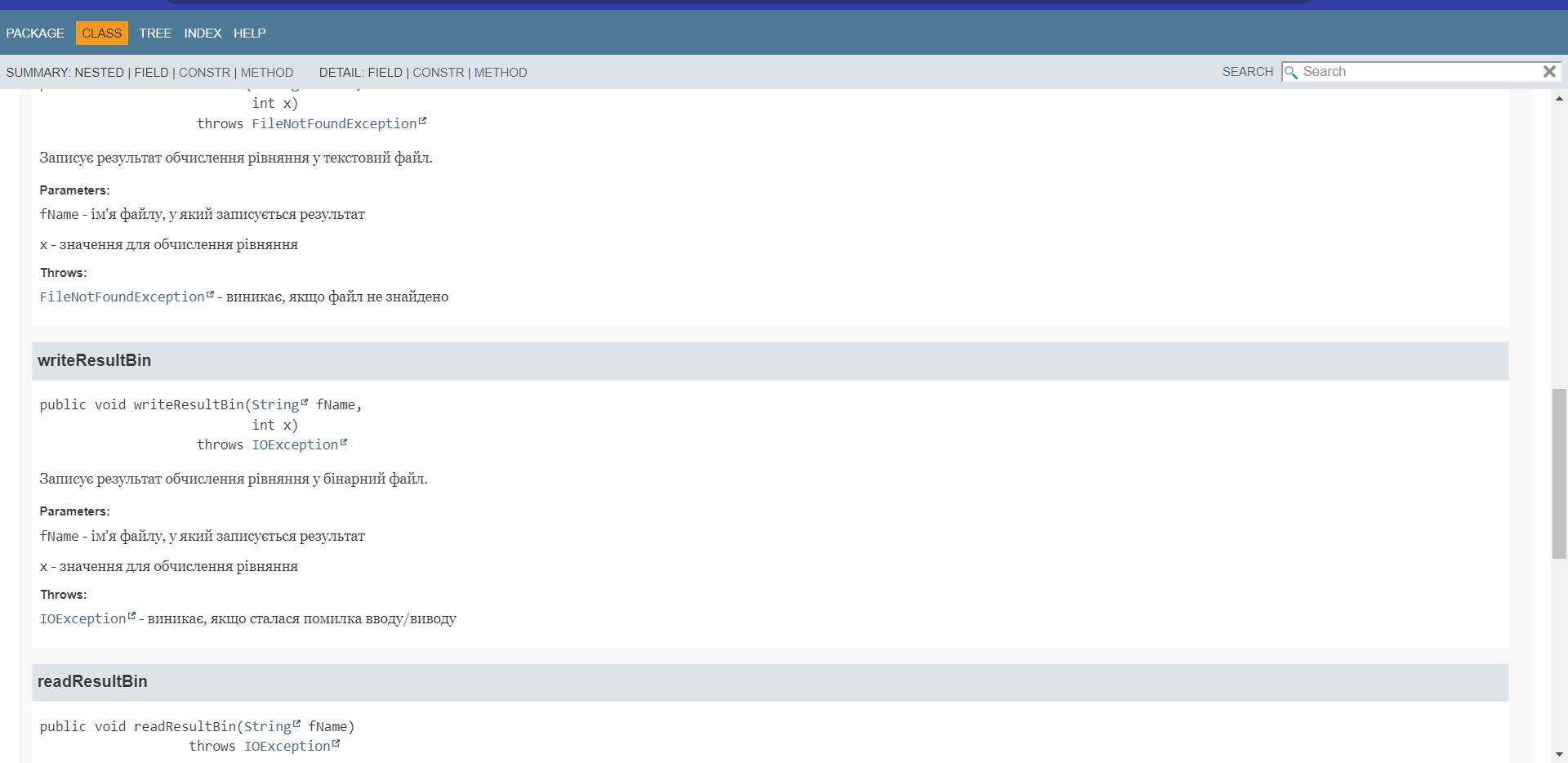
**Результат роботи програми:**

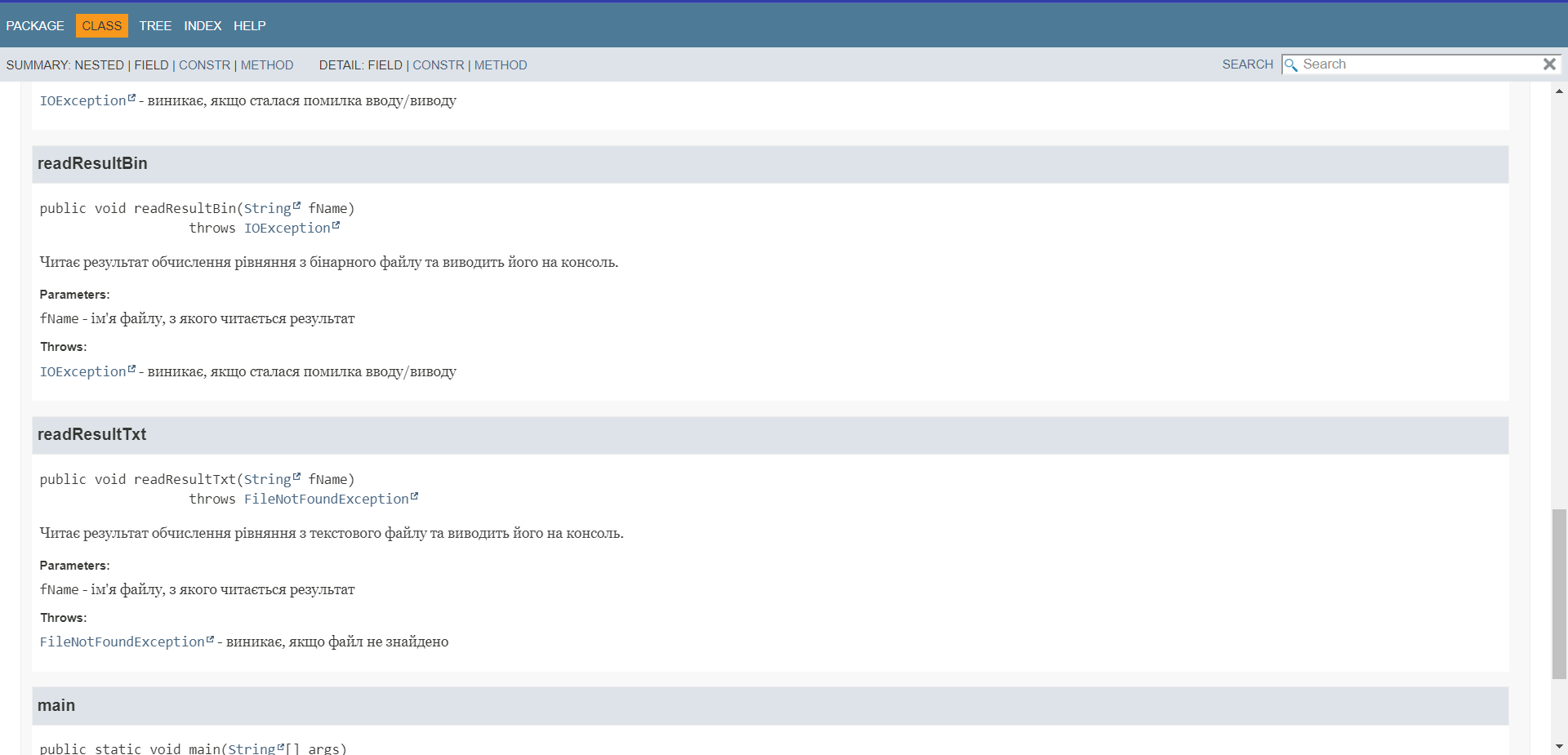
****

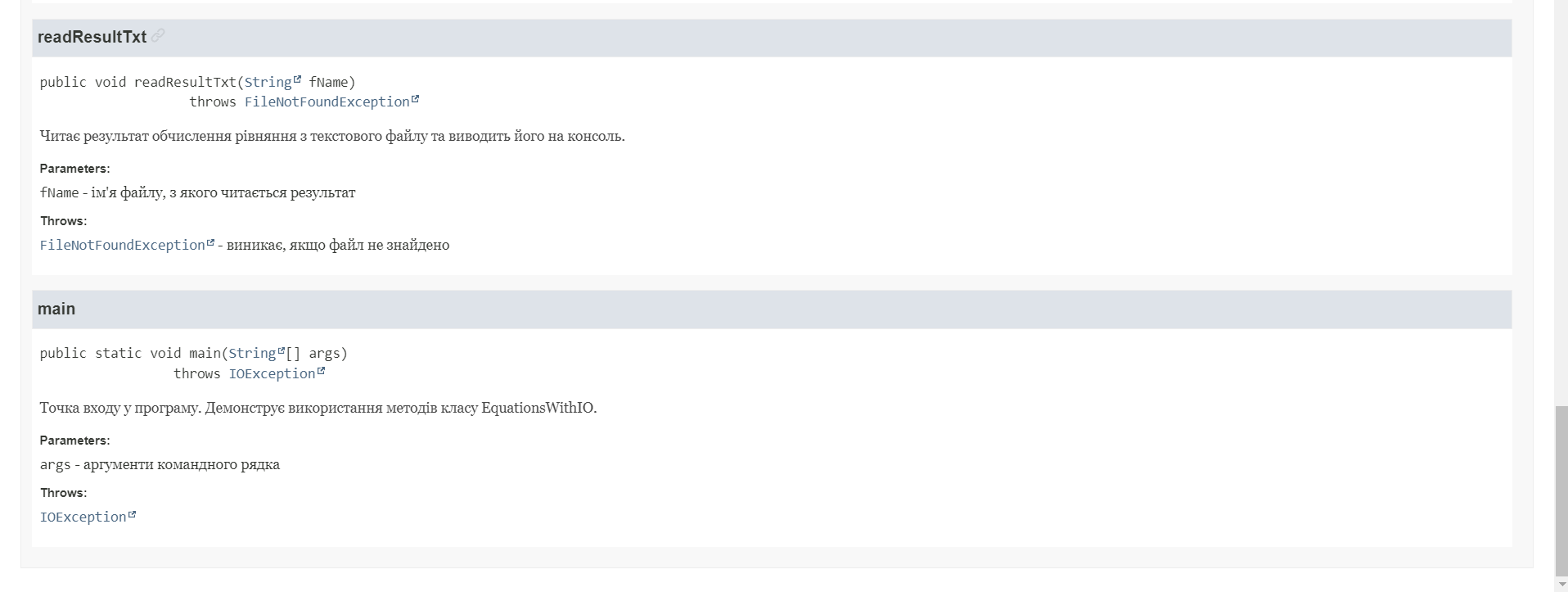
****

****

****



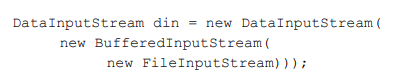




**Контрольні запитання**

1. ***Розкрийте принципи роботи з файловою системою засобами мови Java.***

Для створення файлових потоків і роботи з ними у Java є 2 класи, що успадковані від InputStream і OutputStream це - FileInputStream і FileOutputStream. Як і їх суперкласи вони мають методи лише для байтового небуферизованого блокуючого читання/запису даних та керуванням потоками. На відміну від, наприклад, мови програмування С, де для виконання усіх можливих операцій з файлами необхідно мати один вказівник на FILE у мові Java реалізовано інший набагато складніший і гнучкіший підхід, який дозволяє формувати такі властивості потоку, які найкраще відповідають потребам рішення конкретної задачі. Так у Java розділено окремі функціональні можливості потоків на різні класи. Компонуючи ці класи між собою і досягається необхідна кінцева функціональність потоку. Так одні класи, як FileInputStream, забезпечують елементарний доступ до файлів, інші, як PrintWriter, надають додаткової функціональності по високорівневій обробці даних, що пишуться у файл. Ще інші, наприклад, BufferedInputStream забезпечують буферизацію. Таким чином, наприклад, щоб отримати буферизований файловий потік для читання інформації у форматі примітивних типів (char, int, double,…) слід створити потік з одночасним сумісним використанням функціональності класів FileInputStream, 4 BufferedInputStream і DataInputStream. Для цього слід здійснити наступний виклик:

******

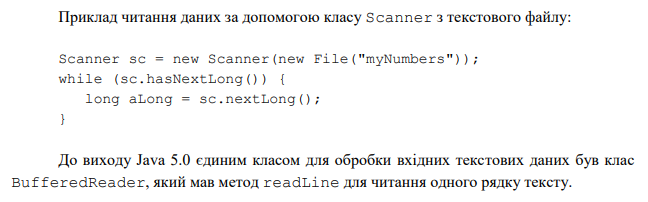
Класи типу BufferedInputStream, DataInputStream, PushbackInputStream (дозволяє читати з потоку дані і повертати їх назад у потік) успадковані від класу FilterInputStream. Вони виступають так званими фільтрами, що своїм комбінуванням забезпечують додаткову лише необхідну функціональність при читанні даних з файлу. Аналогічний підхід застосовано і при реалізації класів для обробки текстових даних, що успадковані від Reader і Writer.

1. ***Охарактеризуйте клас Scanner.***

Для читання текстових потоків найкраще підходить клас Scanner. На відміну від InputStreamReader і FileReader, що дозволяють лише читати текст, він має велику кількість методів, які здатні читати як рядки, так і окремі примітивні типи з подальшим їх перекодуванням до цих типів, робити шаблонний аналіз текстового потоку, здатний працювати без потоку даних та ще багато іншого. Приклад читання даних за допомогою класу Scanner з стандартного потоку вводу:

******

1. ***Наведіть приклад використання класу Scanner.***

******

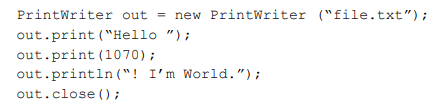
***4. За допомогою якого класу можна здійснити запис у текстовий потік?***

Для буферизованого запису у текстовий потік найкраще використовувати клас PrintWriter.

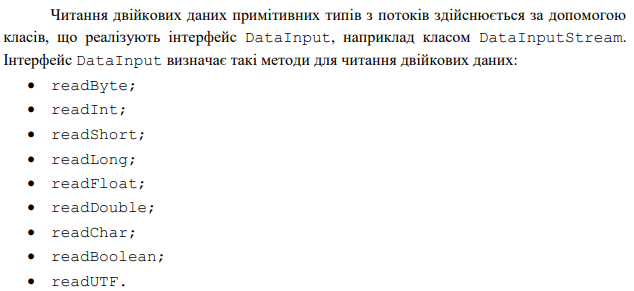
***5. Охарактеризуйте клас PrintWriter.***

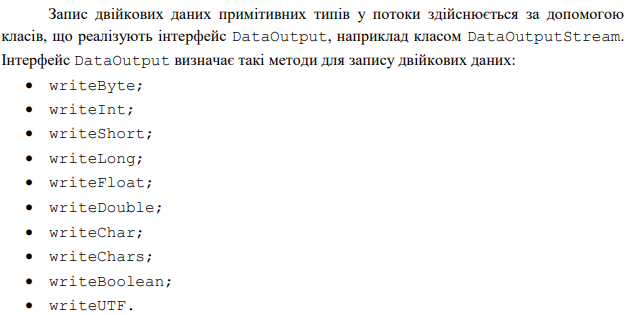
Цей клас має методи для виводу рядків і чисел у текстовому форматі: print, println, printf, - принцип роботи яких співпадає з аналогічними методами Systen.out.

Приклад використання класу PrintWriter:

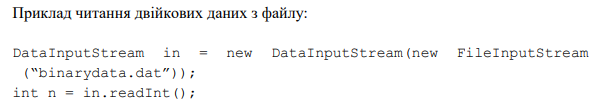
******

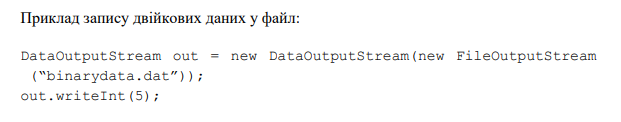
***6. Розкрийте методи читання/запису двійкових даних засобами мови Java.***

******

******

***7. Призначення класів DataInputStream і DataOutputStream.***

******

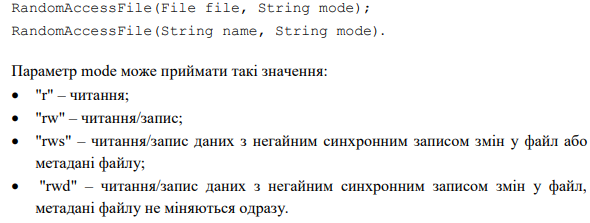
******

***8. Який клас мови Java використовується для здійснення довільного доступу до файлів.***

Керування файлами з можливістю довільного доступу до них здійснюється за допомогою класу RandomAccessFile.

***9. Охарактеризуйте клас RandomAccessFile.***

Відкривання файлу в режимі запису і читання/запису здійснюється за допомогою конструктора, що приймає 2 параметри – посилання на файл (File file) або його адресу (String name) та режим відкривання файлу (String mode):

******

***10. Який зв’язок між інтерфейсом DataOutput і класом DataOutputStream?***

Інтерфейс **DataOutput** та клас **DataOutputStream** в Java пов'язані один з одним і використовуються для запису даних у бінарний потік.

1. **Інтерфейс DataOutput**:
   * **DataOutput** є інтерфейсом в Java, який оголошує методи для запису примітивних типів даних (наприклад, чисел і логічних значень) та рядків в бінарний потік даних.
   * Інтерфейс **DataOutput** включає в себе методи, такі як **writeInt()**, **writeDouble()**, **writeBoolean()**, **writeUTF()**, і т.д., для запису даних певних типів.
2. **Клас DataOutputStream**:
   * **DataOutputStream** є класом, який реалізує інтерфейс **DataOutput**.
   * Клас **DataOutputStream** дозволяє вам створювати об'єкти для запису даних в бінарний потік. Ви можете використовувати методи цього класу для запису даних в форматі, який відповідає методам інтерфейсу **DataOutput**.
   * Наприклад, ви можете створити об'єкт **DataOutputStream**, оточити його навколо **FileOutputStream** або іншого виходу, і використовувати методи **writeInt()**, **writeDouble()**, тощо, для запису даних у бінарний файл або інший потік даних.

Отже, клас **DataOutputStream** інкапсулює логіку запису даних, визначену в інтерфейсі **DataOutput**, і дозволяє зручно записувати дані у бінарний формат.

**Висновок:** оволодів навиками використання засобів мови Java для роботи з потоками і файлами.