Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

з лабораторної роботи №6

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Файли»

Виконав: ст.гр. КІ-34

Турчин П.Р.

Прийняв:

викл. каф. ЕОМ

Іванов Ю. С.

**Львів 2022**

**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 №5. Написати програму для тестування коректності роботи розробленого класу.
2. Для розробленої програми згенерувати документацію.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагмент згенерованої документації.
4. Дати відповідь на контрольні запитання.

**Варіант 23**

*ctg(8x)/x*

**Лістинг програми:**

**Файл App.java**

*/\*\*  
 \* lab 6 package  
 \*/*package KI34.Turchyn.Lab6;  
  
import java.util.Scanner;  
import java.io.\*;  
  
import static java.lang.System.*out*;  
  
*/\*\*  
 \* Class <code>App</code> Implements driver for App class  
 \** ***@author*** *Pavlo Turchyn  
 \** ***@version*** *1.0  
 \*/*public class App {  
 */\*\*  
 \** ***@param*** *args function's parameters  
 \*/* public static void main(String[] args) throws IOException {  
 try {  
 *out*.print("Enter file name: ");  
 Scanner in = new Scanner(System.*in*);  
 String fName = in.nextLine();  
 PrintWriter fout = new PrintWriter(new File(fName));  
 try {  
 try {  
 CalcWFio obj = new CalcWFio();  
  
 Equations eq = new Equations();  
 *out*.print("Enter X: ");  
 int step =in.nextInt();  
  
  
 for(int i=1; i<=5; i++) {  
 double result = eq.calculate(step);  
  
 fout.printf("ctg(%d) = %f\n ",step,result);  
  
 step = step+25;  
 System.*out*.println("Result is: " + result);  
 obj.writeResTxt("textRes.txt", result);  
 obj.writeResBin("BinRes.bin", result);  
  
 obj.readResBin("BinRes.bin");  
 obj.readResTxt("textRes.txt");  
 }  
  
 } finally {  
  
 fout.flush();  
 fout.close();  
 }  
 } catch (CalcException ex) {  
  
  
 *out*.print(ex.getMessage());  
 }  
 } catch (FileNotFoundException ex) {  
  
  
 *out*.print("Exception reason: Perhaps wrong file path");  
 }  
 }  
}

**Файл CalcWFio.java**

package KI34.Turchyn.Lab6;  
import java.io.\*;  
import java.util.Scanner;  
  
*/\*\*  
 \* Class <code>CalcWFio</code> implements method for ctg(8x)/x expression calculation and methods for writing and reading files  
 \*  
 \** ***@author*** *Andriy Pavlo Turchyn  
 \** ***@version*** *1.0  
 \*/*class CalcWFio {  
 //  
  
 */\*\*  
 \* Method writes txt file  
 \*  
 \** ***@param*** *fName File's name  
 \** ***@throws*** *FileNotFoundException error  
 \*/* public void writeResTxt(String fName, double result) throws FileNotFoundException {  
 PrintWriter f = new PrintWriter(fName);  
 f.printf("%f ", result);  
 f.close();  
 }  
 */\*\*  
 \* Method read txt file  
 \*  
 \** ***@param*** *fName File's name  
 \*/* public void readResTxt(String fName) {  
 try {  
 File f = new File(fName);  
 if (f.exists()) {  
 Scanner s = new Scanner(f);  
 result = s.nextDouble();  
 s.close();  
 } else  
 throw new FileNotFoundException("File " + fName + "not found");  
  
 } catch (FileNotFoundException ex) {  
 System.*out*.print(ex.getMessage());  
 }  
 }  
 */\*\*  
 \* Method writes bin file  
 \*  
 \** ***@param*** *fName File's name  
 \** ***@throws*** *IOException error  
 \*/* public void writeResBin(String fName, double result) throws FileNotFoundException, IOException {  
  
 DataOutputStream f = new DataOutputStream(new FileOutputStream(fName));  
 f.writeDouble(result);  
 f.close();  
  
 }  
 */\*\*  
 \* Method reads bin file  
 \*  
 \** ***@param*** *fName File's name  
 \** ***@throws*** *IOException error  
 \*/* public void readResBin(String fName) throws FileNotFoundException, IOException {  
  
 DataInputStream f = new DataInputStream(new FileInputStream(fName));  
 result = f.readDouble();  
 f.close();  
 }  
  
  
 */\*\*  
 \* Method get the result of the expression  
 \*/* private double result;  
}

*/\*\*  
 \* Method reads bin file  
 \*  
 \** ***@param*** *fName File's name  
 \** ***@throws*** *IOException error  
 \*/* public void readResBin(String fName) throws IOException {  
 DataInputStream f = new DataInputStream(new FileInputStream(fName));  
 result = f.readDouble();  
 f.close();  
 }  
  
 */\*\*  
 \* Method calculates the sin(3x-5)/ctg(2x) expression  
 \*  
 \** ***@param*** *x Angle in degrees  
 \** ***@throws*** *CalcException  
 \*/* public void calculate(int x) throws CalcException {  
 double radSin, radCtg;  
 radSin = (3 \* x - 5) \* Math.*PI* / 180.0;  
 radCtg = 2 \* x \* Math.*PI* / 180.0;  
 try {  
 result = Math.*sin*(radSin) / (1.0 / Math.*tan*(radCtg));  
  
 // Якщо результат не є числом, то генеруємо виключення  
 if (result == Double.*NaN* || result == Double.*NEGATIVE\_INFINITY* ||  
 result == Double.*POSITIVE\_INFINITY* || x % 90 == 0 || x == 45 || (2 \* x) % 270.0 == 0)  
 throw new ArithmeticException();  
 } catch (ArithmeticException ex) {  
 // створимо виключення вищого рівня з поясненням причини  
 // виникнення помилки  
 if (radCtg % Math.*PI* / 2.0 == 0)  
 throw new CalcException("Exception reason: Illegal value of X for cotangent calculation");  
 else if (x == 45 || (2 \* x) % 270.0 == 0)  
 throw new CalcException("Exception reason: X = 0");  
 else  
 throw new CalcException("Unknown reason of the exception during exception calculation");  
 }  
  
 }  
  
 */\*\*  
 \* Method get the result of the expression  
 \*/* public double getResult() {  
 return result;  
 }  
}

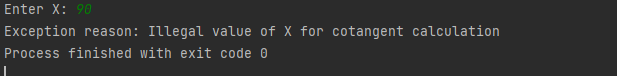
**Файл CalcException.java**

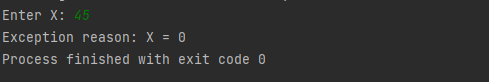
*/\*\*  
 \* lab 6 package  
 \*/*package KI34.Turchyn.Lab6;  
*/\*\*  
 \* Class <code>CalcException</code> more precises ArithmeticException  
 \*  
 \** ***@author*** *Turchyn Pavlo  
 \** ***@version*** *1.0  
 \*/*class CalcException extends ArithmeticException  
{  
 public CalcException(){}  
  
 public CalcException(String cause)  
 {  
 super(cause);  
 }  
}

**Файл Equations.java**

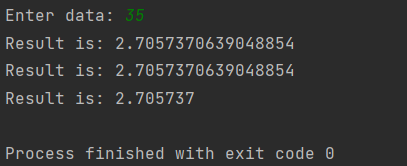
package KI34.Turchyn.Lab6;  
*/\*\*  
 \** ***@version*** *1.0  
 \*/*public class Equations {  
 */\*\*  
 \* Method calculates the ctg(8x)/x expression  
 \*  
 \** ***@param*** *x Angle in degrees  
 \** ***@throws*** *CalcException  
 \*/* public double calculate(double x) throws CalcException {  
 double y, radCtg;  
 radCtg = 8 \* x \* Math.*PI* / 180.0;  
 try {  
 y = (1.0 / Math.*tan*(radCtg) / x);  
  
 // Якщо результат не є числом, то генеруємо виключення  
 if (y == Double.*NaN* || y == Double.*NEGATIVE\_INFINITY* ||  
 y == Double.*POSITIVE\_INFINITY* || x % 180 == 0 || x == 0 || (8 \* x) % 180.0 == 0)  
 throw new ArithmeticException();  
 } catch (ArithmeticException ex) {  
 // створимо виключення вищого рівня з поясненням причини  
 // виникнення помилки  
 if (radCtg % Math.*PI* / x == 0.000000000)  
 throw new CalcException("Exception reason: Illegal value of X for cotangent calculation");  
 else if(x == 0)  
 throw new CalcException("Exception reason: X = 0");  
 else if (x == 45 || (8 \* x) % 180.0 == 0)  
 throw new CalcException("Exception reason: X = 0");  
 else  
 throw new CalcException("Unknown reason of the exception during exception calculation");  
 }  
  
 return y;  
 }  
  
}

**Результат виконання програми:**

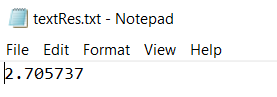


*Виключення для неприпустимого значення для котангеса* 

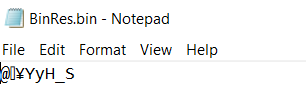
*Виключення при діленні на нуль*



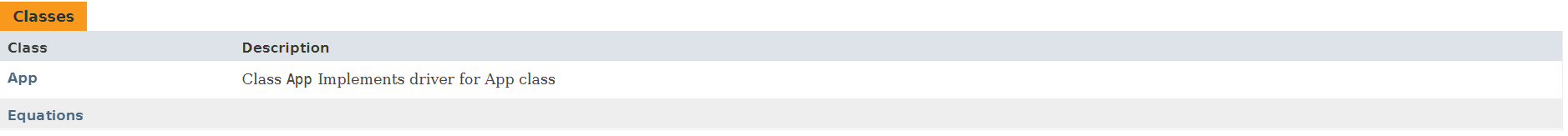
*Успішне виконання програми*



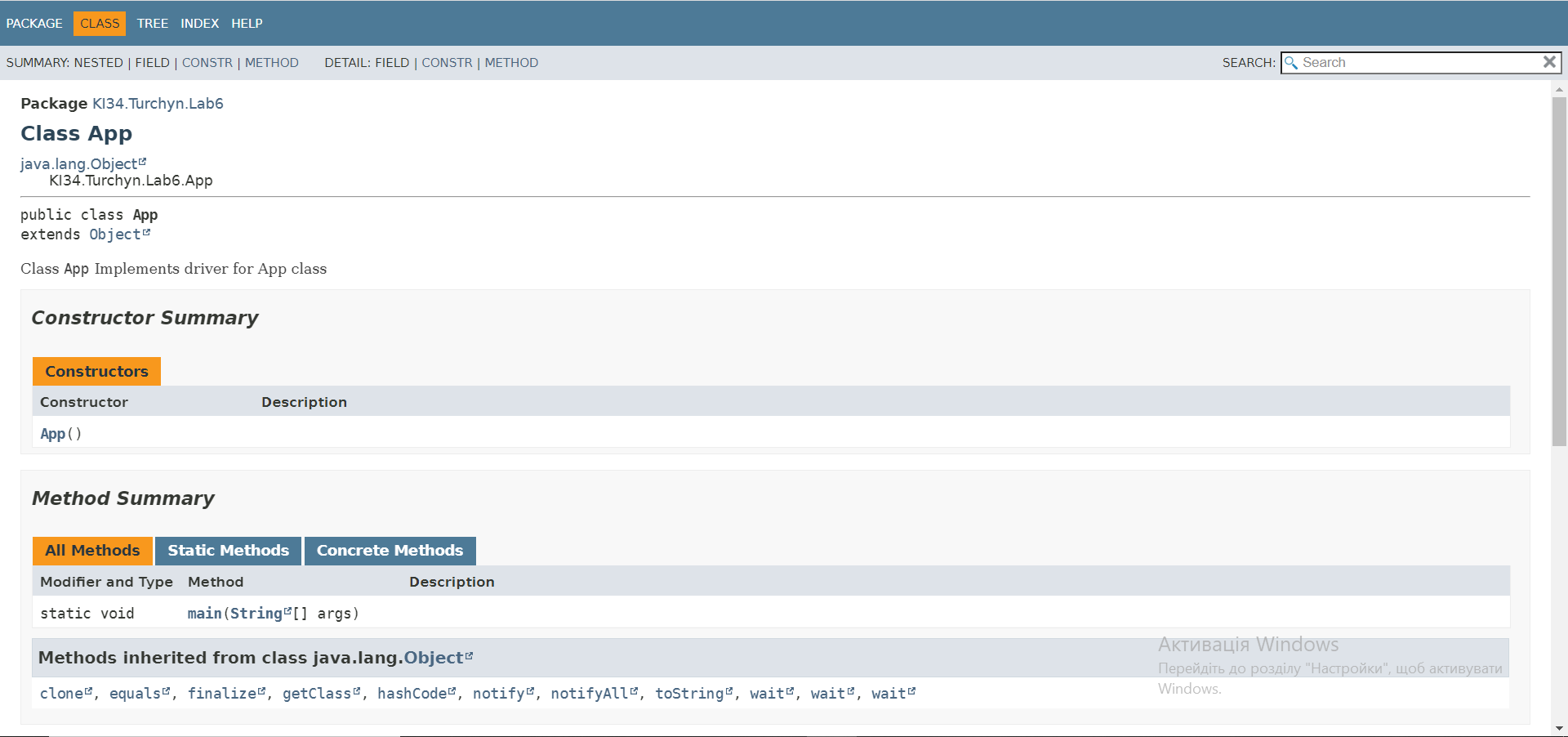
*Успішний запис результату у текстовий файл*



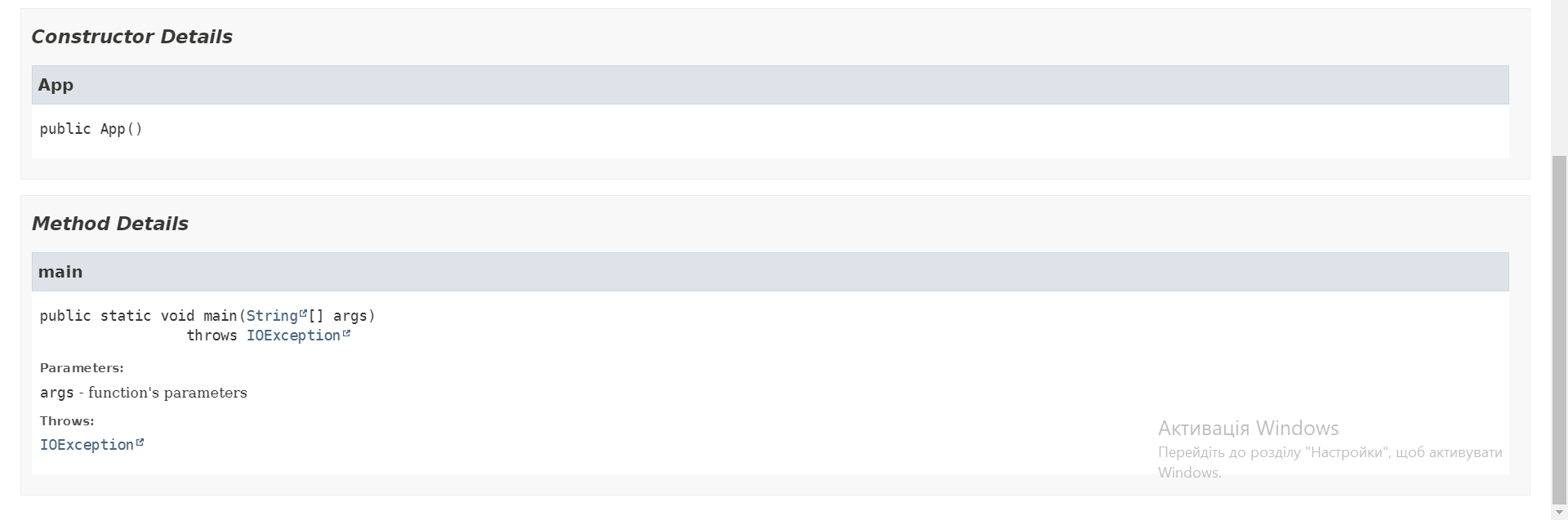
*Успішний запис результату у бінарний файл*

**

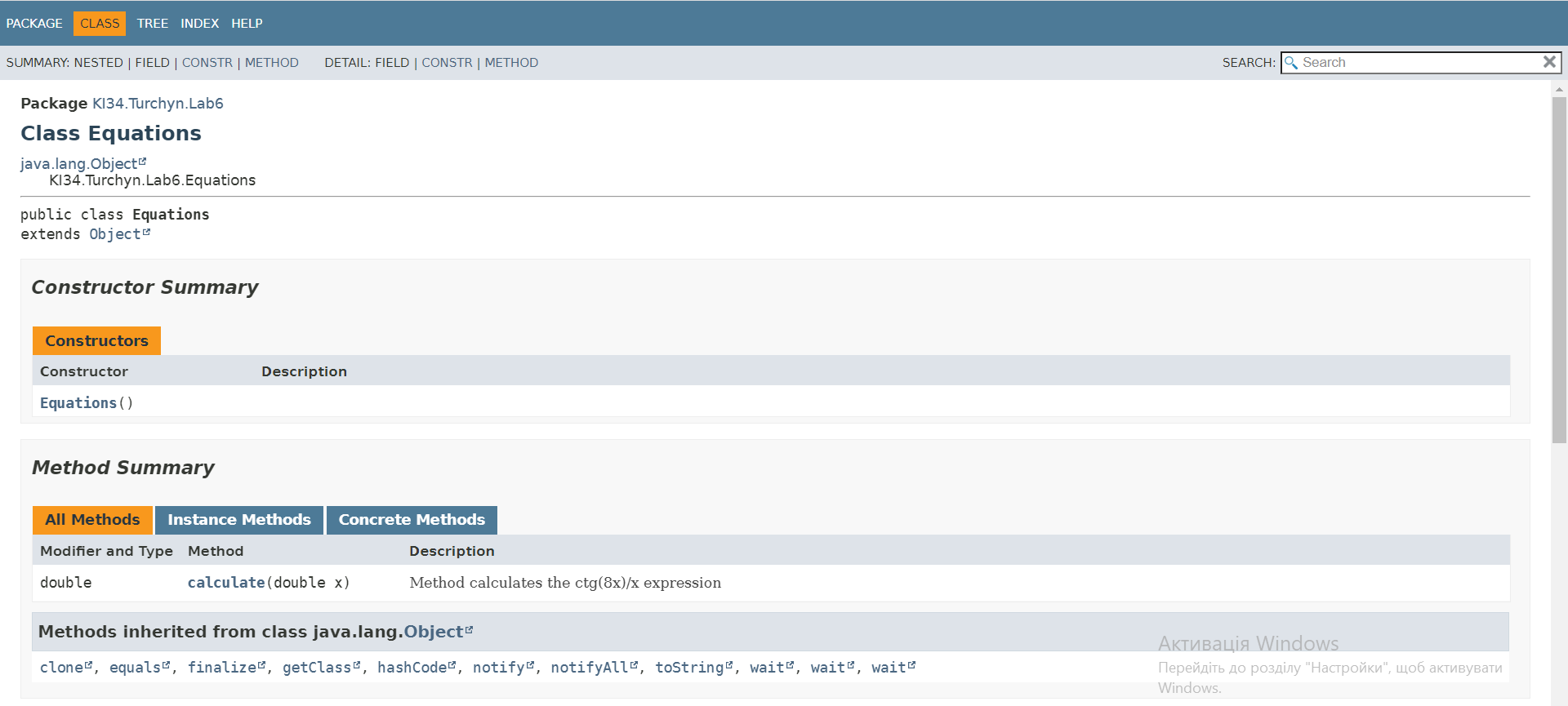
*Згенерована документація*

**

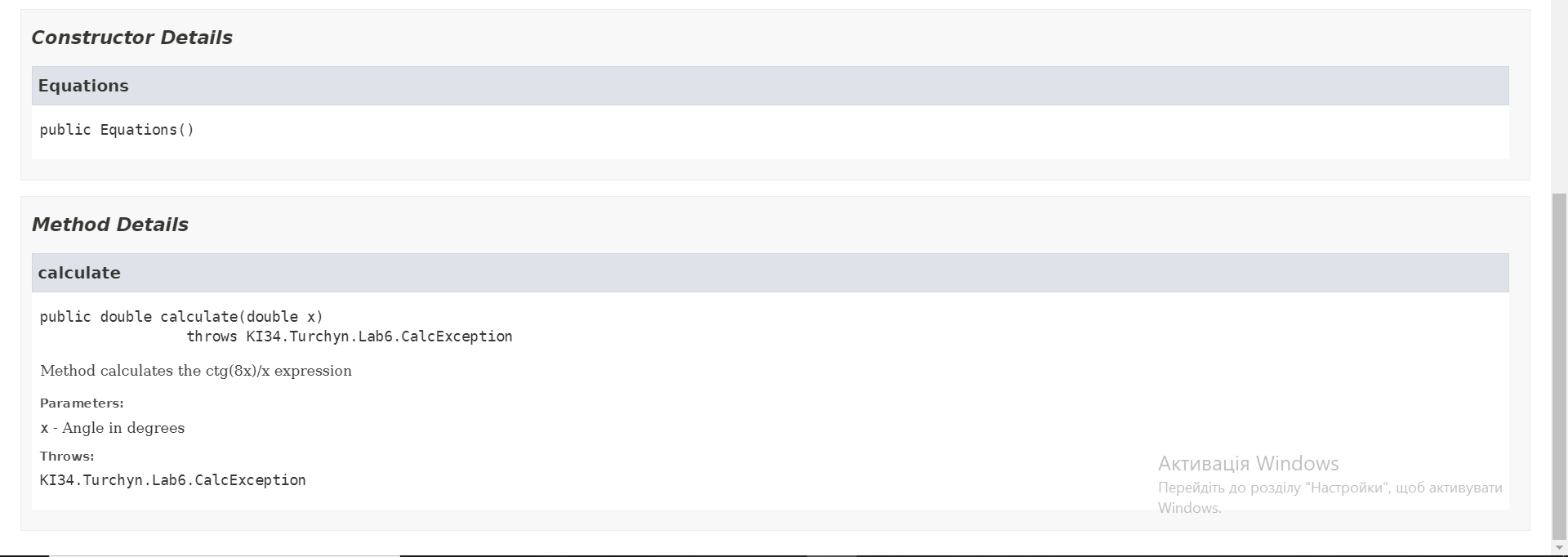
*Інформація про клас App*

**

*Інформація про клас App*

**

*Інформація про клас Equations*

**

*Інформація про клас Equations*

**Відповіді на 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.

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.

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

Приклад читання даних за допомогою класу Scanner з стандартного потоку вводу:

Scanner sc = new Scanner(System.in);

int i = sc.nextInt();

Приклад читання даних за допомогою класу Scanner з текстового файлу:Scanner sc = new Scanner(new File("myNumbers"));

while (sc.hasNextLong()) {

long aLong = sc.nextLong();

}

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

Для буферизованого запису у текстовий потік найкраще використовувати клас PrintWriter.

1. **Охарактеризуйте клас PrintWriter.**
2. **Розкрийте методи читання/запису двійкових даних засобами мови Java.**

Читання двійкових даних примітивних типів з потоків здійснюється за допомогою класів, що реалізують інтерфейс DataInput, наприклад класом DataInputStream. Інтерфейс DataInput визначає такі методи для читання двійкових даних:

* readByte;
* readInt;
* readShort;
* readLong;
* readFloat;
* readDouble;
* readChar;
* readBoolean;
* readUTF.

Запис двійкових даних примітивних типів у потоки здійснюється за допомогою класів, що реалізують інтерфейс DataOutput, наприклад класом DataOutputStream. Інтерфейс DataOutput визначає такі методи для запису двійкових даних:

* writeByte;
* writeInt;
* writeShort;
* writeLong;
* writeFloat;
* writeDouble;
* writeChar;
* writeBoolean;
* writeUTF.

1. **Призначення класів DataInputStream і DataOutputStream.**

Класи DataOutputStream і DataInputStream дозволяють записувати і зчитувати дані примітивних типів.

Клас DataOutputStream представляє потік виведення і призначений для запису даних примітивних типів, таких, як int, double і т.д. Для запису кожного з примітивних типів призначений свій метод.

Клас DataInputStream діє протилежним чином - він зчитує з потоку дані примітивних типів. Відповідно для кожного примітивного типу визначено свій метод для зчитування.

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

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

1. **Охарактеризуйте клас RandomAccessFile.**

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

Файли, що керуються класом RandomAccessFile, оснащені вказівником на позицію наступного байту, що має читатися або записуватися. Для того, щоб перемістити даний вказівник на довільну позицію в межах файлу використовується метод void seek(long pos). Параметр long pos визначає номер байту, що має читатися або записуватися.

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

Запис двійкових даних примітивних типів у потоки здійснюється за допомогою класів, що реалізують інтерфейс DataOutput, наприклад класом DataOutputStream.

**Висновок:**

На цій лабораторній роботі я оволодів навиками використання засобів мови Java для роботи з потоками і файлами.