Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

З лабораторної роботи №3

Варіант – 24

З дисципліни: «Кросплатформні засоби програмування»

На тему: «ВИКЛЮЧЕННЯ»

Виконав: ст. гр. КІ-301

Чоланюк О.Ю.

Прийняв:

Майдан М.В.

Львів 2023

Мета: оволодіти навиками використання механізму виключень при написанні програм мовою Java.

ТЕОРЕТИЧНІ ВІДОМОСТІ

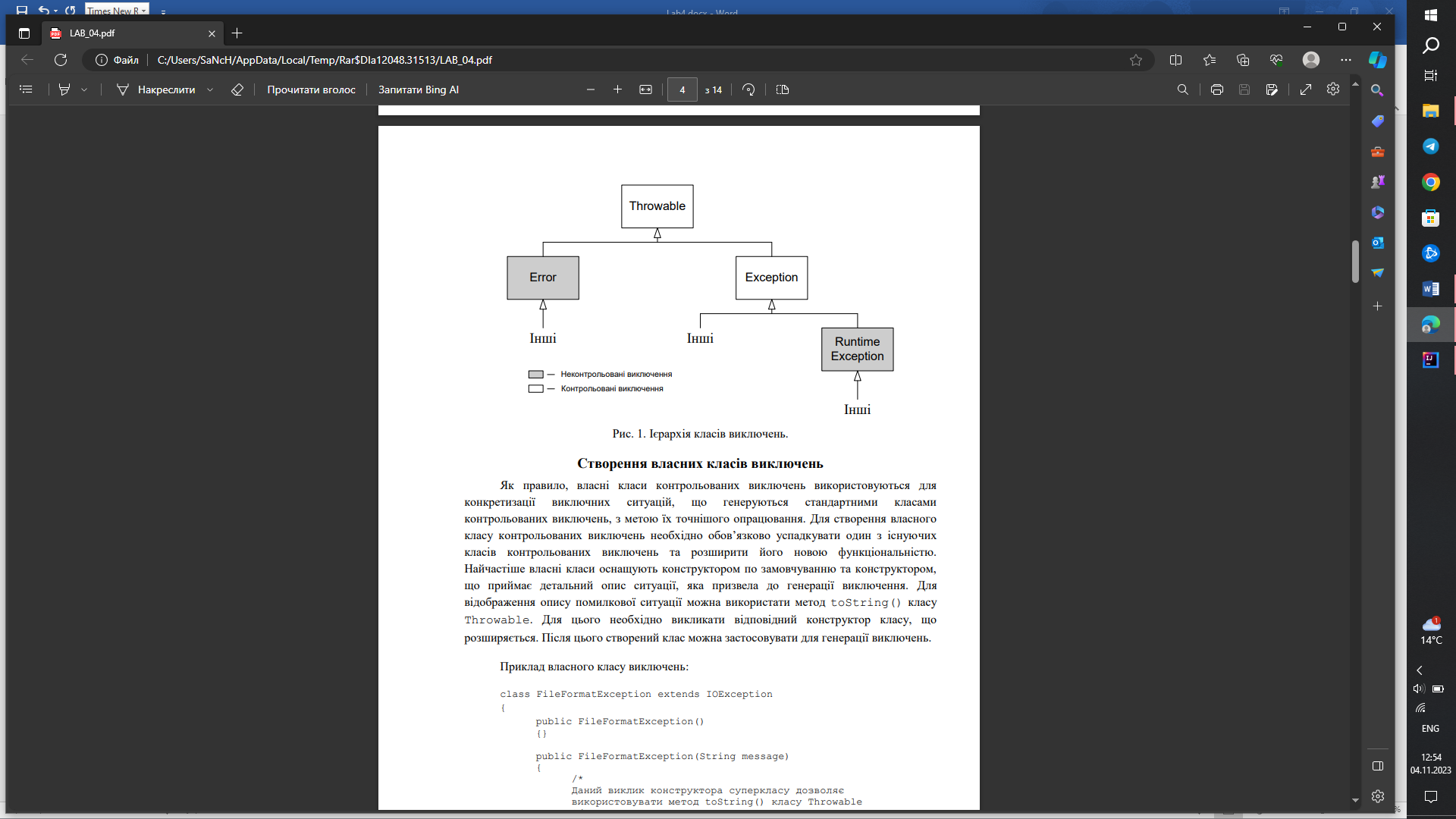
Виключення – це механізм мови Java, що забезпечує негайну передачу керування блоку коду опрацювання критичних помилок при їх виникненні уникаючи процесу розкручування стеку. Генерація виключень застосовується при:

• помилках введення, наприклад, при введенні назви неіснуючого файлу або Інтернет адреси з подальшим зверненням до цих ресурсів, що призводить до генерації помилки системним програмним забезпеченням;

• збоях обладнання;

• помилках, що пов’язані з фізичними обмеженнями комп’ютерної системи, наприклад, при заповненні оперативної пам’яті або жорсткого диску;

• помилках програмування, наприклад, при некоректній роботі методу, читанні елементів порожнього стеку, виходу за межі масиву тощо.



Створення власних класів виключень

Як правило, власні класи контрольованих виключень використовуються для конкретизації виключних ситуацій, що генеруються стандартними класами контрольованих виключень, з метою їх точнішого опрацювання. Для створення власного класу контрольованих виключень необхідно обов’язково успадкувати один з існуючих класів контрольованих виключень та розширити його новою функціональністю. Найчастіше власні класи оснащують конструктором по замовчуванню та конструктором, що приймає детальний опис ситуації, яка призвела до генерації виключення. Для відображення опису помилкової ситуації можна використати метод toString() класу Throwable. Для цього необхідно викликати відповідний конструктор класу, що розширяється. Після цього створений клас можна застосовувати для генерації виключень.

Оголошення контрольованих виключень

Виключення можуть генеруватися лише методами. Якщо метод може генерувати виключення певного класу, то назву цього класу слід вказати в заголовку методу після 5 ключового слова throws. Якщо метод може генерувати кілька видів виключень, то всі вони перелічуються через кому.

Генерація контрольованих виключень

Генерація контрольованих виключень відбувається за допомогою ключового слова throw після якого необхідно вказати об’єкт класу виключення який і є власне виключенням, що генерує метод. Це можна зробити двома шляхами, використовуючи іменовані або анонімні об’єкти:

1. throw new IOException();

2. IOException ex = new IOException(); throw ex;

Деякі класи контрольованих виключень мають конструктори, що приймають рядок з описом причини виникнення виключення. Такі конструктори корисно застосовувати для полегшення пошуку місця виникнення помилки під час виконання програми.

Перехоплення виключень

Перехоплення виключень здійснюється з метою коректного опрацювання критичних помилок повідомлення про які були згенеровані у процесі виконання програми. Якщо виключення виникає і ніде не перехоплюється, то програма припиняє свою роботу і виводить на консоль повідомлення про тип виключення та вміст стеку. Графічні програми 6 (аплети і прикладні програми) виводять те ж саме повідомлення, повертаючись після цього у цикл обробки користувацького інтерфейсу.

Створити клас, що реалізує метод обчислення виразу заданого варіантом. Написати на мові Java та налагодити програму-драйвер для розробленого класу. Результат обчислень записати у файл. При написанні програми застосувати механізм виключень для виправлення помилкових ситуацій, що можуть виникнути в процесі виконання програми. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету. 2. Автоматично згенерувати документацію до розробленого пакету. 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.

Завдання

1. Створити клас, що реалізує метод обчислення виразу заданого варіантом. Написати на мові Java та налагодити програму-драйвер для розробленого класу. Результат обчислень записати у файл. При написанні програми застосувати механізм виключень для виправлення помилкових ситуацій, що можуть виникнути в процесі виконання програми. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

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

Мій варіант: y=sin(x-9)/(x-cos(2x))

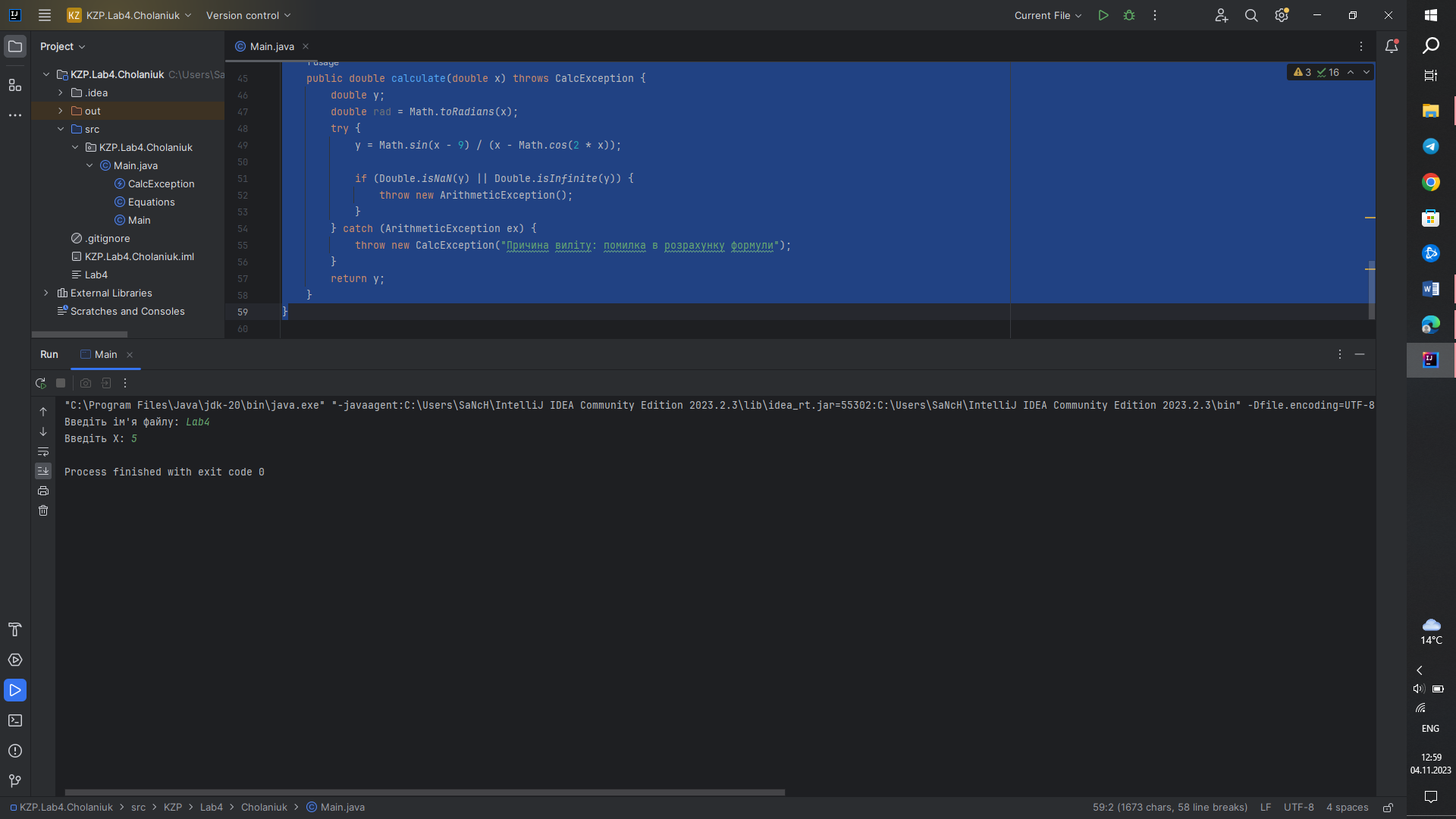
Код:

package KZP.Lab4.Cholaniuk;  
  
import java.util.\*;  
import java.io.\*;  
import static java.lang.System.*out*;  
  
public class Main {  
 public static void main(String[] args) {  
 try {  
 *out*.print("Введіть ім'я файлу: ");  
 Scanner in = new Scanner(System.*in*);  
 String fName = in.nextLine();  
 PrintWriter fout = new PrintWriter(new File(fName));  
 try {  
 try {  
 Equations eq = new Equations();  
 *out*.print("Введіть X: ");  
 double x = in.nextDouble();  
 double result = eq.calculate(x);  
 fout.print(result);  
 } finally {  
 fout.flush();  
 fout.close();  
 }  
 } catch (CalcException ex) {  
 *out*.print(ex.getMessage());  
 }  
 } catch (FileNotFoundException ex) {  
 *out*.print("Причина виліту: неправильний шлях до файлу");  
 }  
 }  
}

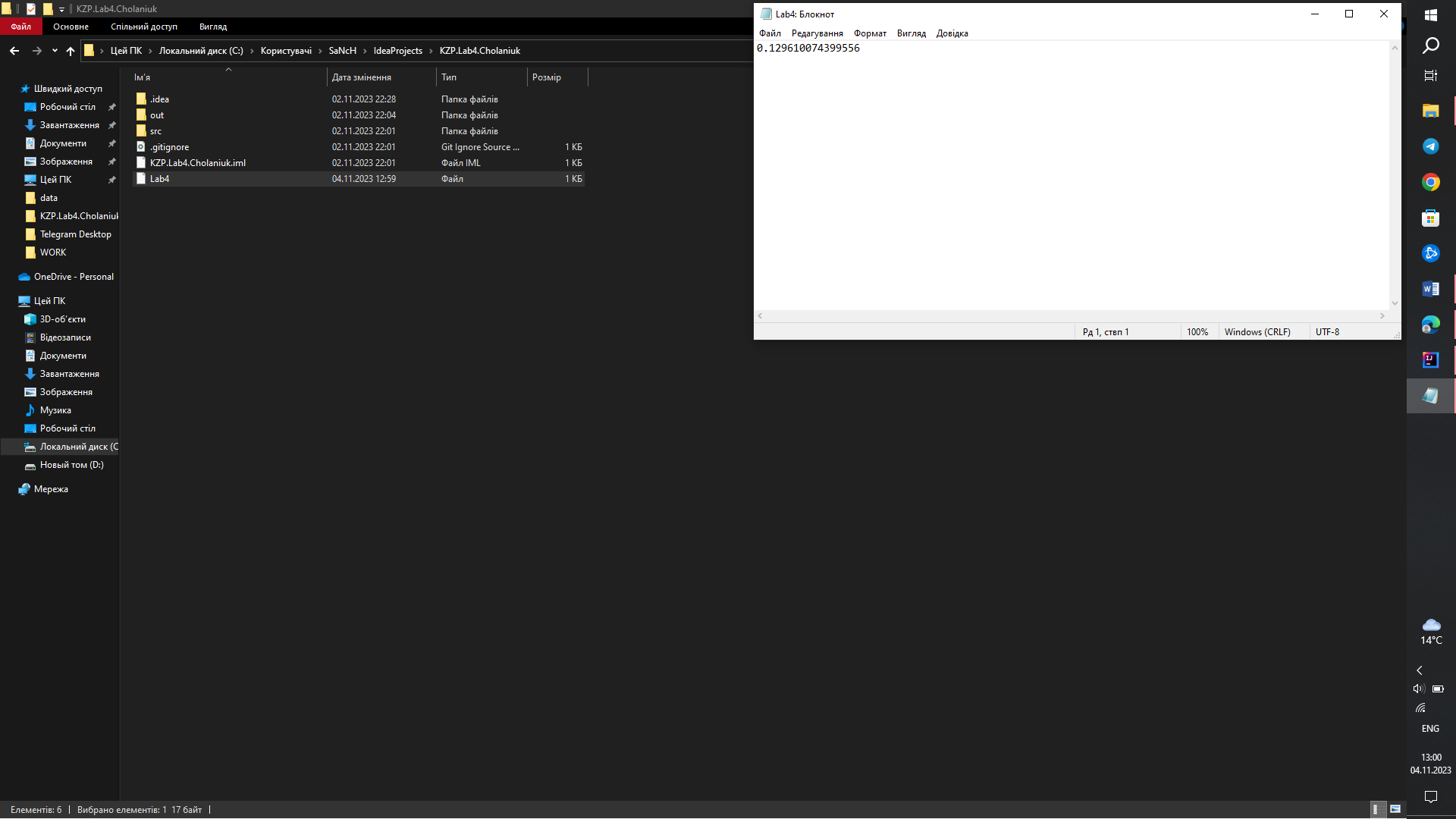
package KZP.Lab4.Cholaniuk;  
  
public class Equations {  
 public double calculate(double x) throws CalcException {  
 validateInput(x);  
  
 double y;  
 double rad = Math.*toRadians*(x);  
  
 try {  
 y = Math.*sin*(x - 9) / (x - Math.*cos*(2 \* x));  
 System.*out*.println(y);  
 if (Double.*isNaN*(y) || Double.*isInfinite*(y)) {  
 throw new ArithmeticException();  
 }  
 } catch (ArithmeticException ex) {  
 throw new CalcException("Причина виліту: помилка в розрахунку формули");  
 }  
 return y;  
 }  
  
 private void validateInput(double x) throws CalcException {  
 if (Double.*isInfinite*(x) || Double.*isNaN*(x)) {  
 throw new CalcException("Причина виліту: некоректне введення X");  
 }  
  
 if (x - Math.*cos*(2 \* x) == 0) {  
 throw new CalcException("Причина виліту: ділення на нуль");  
 }  
 }  
}

package KZP.Lab4.Cholaniuk;  
  
class CalcException extends ArithmeticException {  
 public CalcException() {  
 }  
  
 public CalcException(String cause) {  
 super(cause);  
 }  
}

Terminal:



Lab4.txt:



Перевірка:

