Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



**Звіт**

до лабораторної роботи №4

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Виключення»

**Варіант 3**

Виконав: ст. гр. КІ-305

Гнідець В.М.

Прийняв:

Іванов Ю.С.

Львів 2023

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

**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.

**Варіант 3:** y=sin(x)/cos(x)

**Код програми**

**Class Equations**

package KI305\_Gnidec\_Lab4;  
  
*/\*\*  
 \* Клас <code>Equations</code> реалізує метод для обчислення  
 виразу ((sin(x) / cos(x)).  
 \* Автор: [Гнідець Володимир]  
 \* Версія 1.2  
 \*/*public class Equations {  
 */\*\*  
 \* Метод обчислює вираз ((sin(x) / cos(x)).  
 \*  
 \* @param x Значення X, для якого проводиться обчислення.  
 \* @return Результат обчислення виразу ((sin(x) / cos(x)).  
 \* @throws CalcException якщо під час обчислення виникає  
 виняток.  
 \*/  
 /\*\*public double checkForNumber(int x) throws TypeException {  
 try {  
  
 if (x == 3)  
 {  
 return x;  
 }  
 else {  
 throw new ArithmeticException();  
 }  
 }  
 catch (ArithmeticException ex1) {  
 throw new CalcException("Причина винятку: Некоректне значення X. Він повинен бути числом");  
  
 }  
  
 }\*/* public double calculate(int x) throws CalcException, TestException {  
 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 (CalcException | TestException ex) {  
 if (rad == Math.*PI* / 2.0 || rad == -Math.*PI* / 2.0){  
 throw new CalcException("Error");  
 }  
 else {  
 throw new CalcException("Another error");  
 }  
  
 /\*if (rad == Math.PI / 2.0 || rad == -Math.PI / 2.0) {  
 throw new CalcException("Причина винятку: Некоректне значення X для обчислення тангенсу");  
 }  
 else {  
 throw new CalcException("Невідома причина винятку під час обчислення виразу");  
 }\*/  
 }  
 return y;  
 }  
}

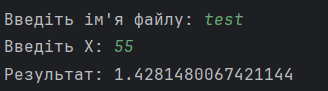
**Class EquationsApp**

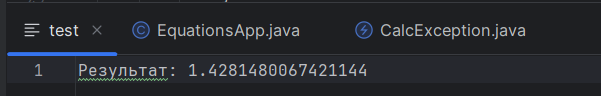
package KI305\_Gnidec\_Lab4;  
import java.util.Scanner;  
import java.io.\*;  
import static java.lang.System.*out*;  
*/\*\*  
 \* Клас <code>EquationsApp</code> реалізує головний драйвер для  
 класу Equations.  
 \* Версія 1.0  
 \*/*public class EquationsApp {  
 public static void main(String[] args) {  
  
 *out*.print("Введіть ім'я файлу: ");  
  
 Scanner in = new Scanner(System.*in*);  
 String fName = in.nextLine();  
 PrintWriter fout = null;  
  
 try {  
 fout = new PrintWriter(new File(fName));  
 Equations eq = new Equations();  
  
 *out*.print("Введіть X: ");  
  
 int x = in.nextInt();  
 double result = eq.calculate(x);  
  
 fout.print("Результат: " + result);  
 fout.flush();  
 fout.close();  
 *out*.println("Результат: " + result);  
 } catch (FileNotFoundException ex) {  
 *out*.print("Причина винятку: Файл не знайдено");  
 } catch (CalcException | TestException ex) {  
 *out*.print(ex.getMessage());  
 }  
 finally {  
 if (fout != null) {  
 fout.close();  
 }  
 }  
 }  
}

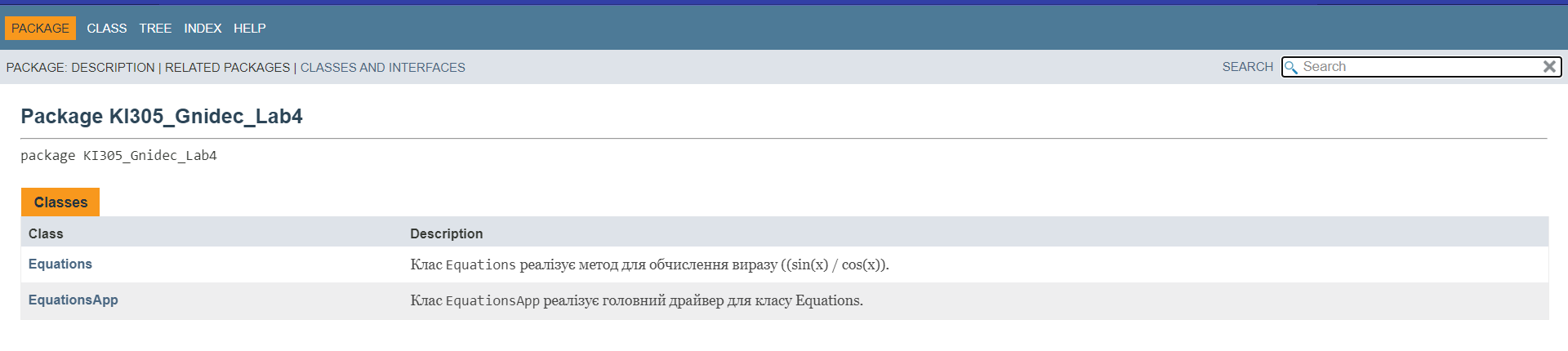
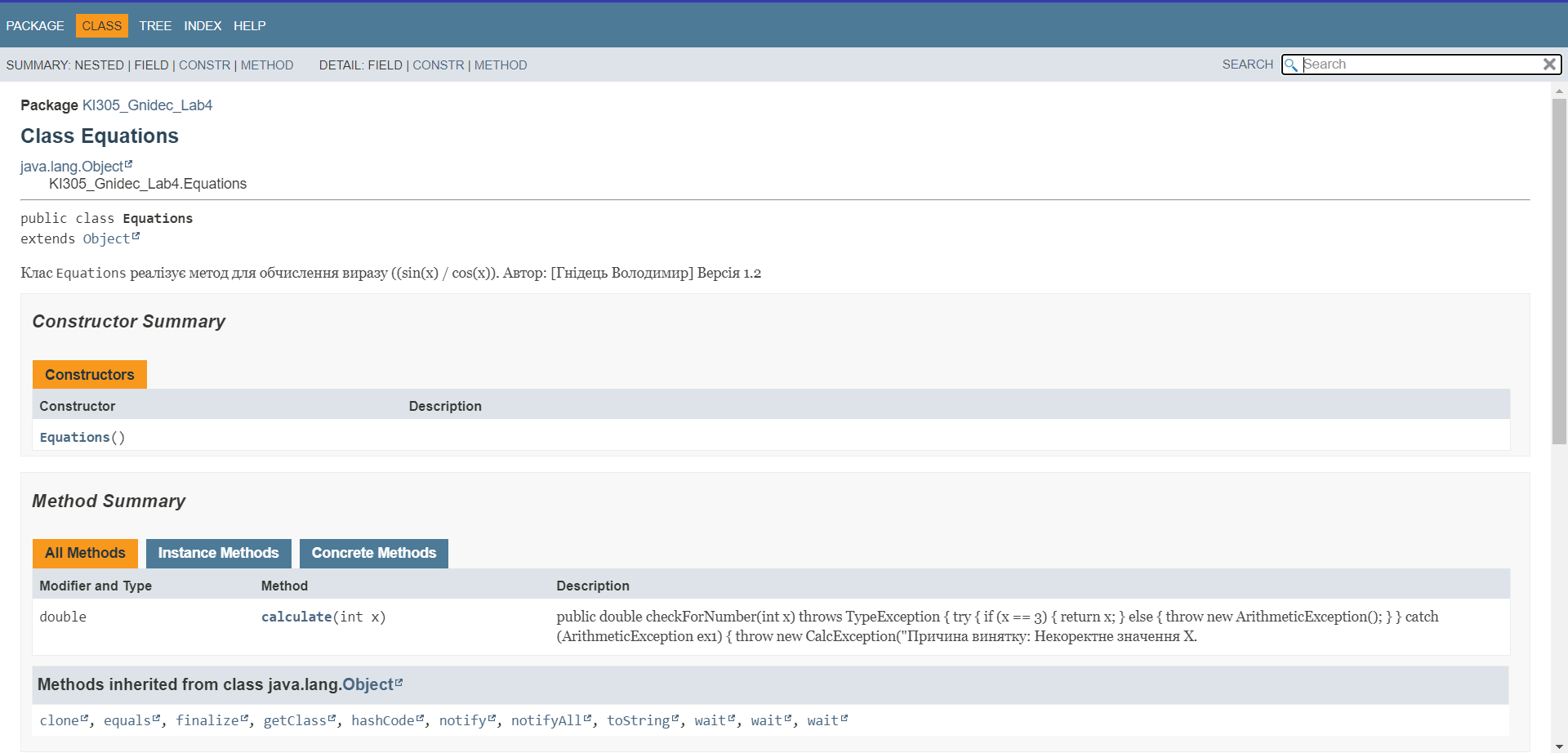
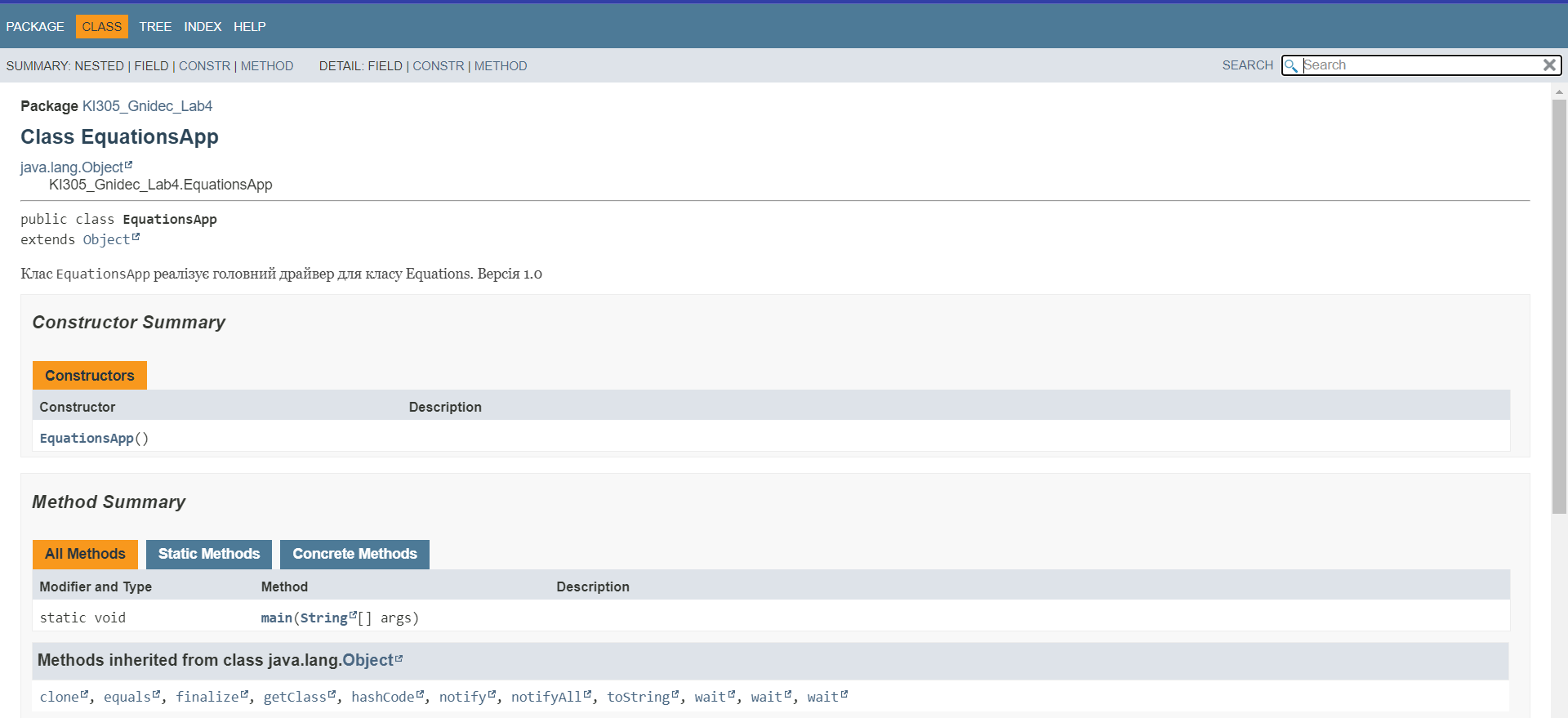
**Class CalcExeption**

package KI305\_Gnidec\_Lab4;  
  
class CalcException extends ArithmeticException {  
  
 public CalcException(String cause) {  
 super(cause);  
 }  
  
}

**Результат роботи програми:**

****

****

**** **** **Контрольні запитання**

1. ***Дайте визначення терміну «виключення».***

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

1. ***У яких ситуаціях використання виключень є виправданим?***

Генерація виключень застосовується при:

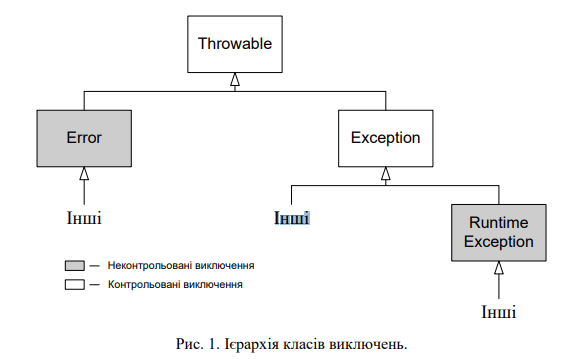
• помилках введення, наприклад, при введенні назви неіснуючого файлу або Інтернет адреси з подальшим зверненням до цих ресурсів, що призводить до генерації помилки системним програмним забезпеченням;

• збоях обладнання;

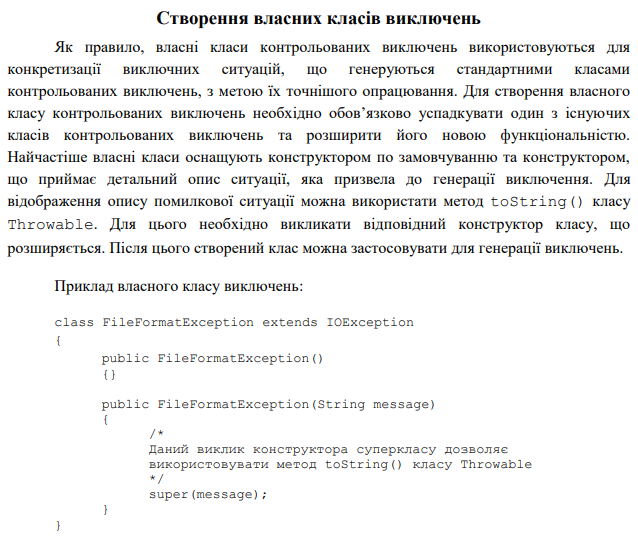
• помилках, що пов’язані з фізичними обмеженнями комп’ютерної системи, наприклад, при заповненні оперативної пам’яті або жорсткого диску;

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

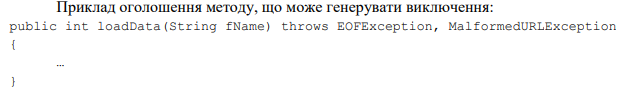
1. ***Яка ієрархія виключень використовується у мові Java?***

******

1. ***Як створити власний клас виключень?***

******

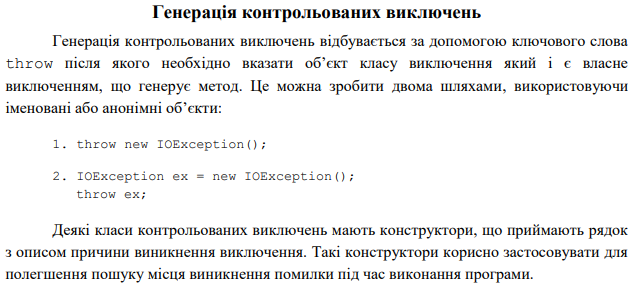
1. ***Який синтаксис оголошення методів, що можуть генерувати виключення?***

******

1. ***Які виключення слід вказувати у заголовках методів і коли?***

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.

1. ***Як згенерувати контрольоване виключення?***

******

1. ***Розкрийте призначення та особливості роботи блоку try.***

Основне призначення блоку try – це спроба виконати деяку частину коду, в якому може виникнути помилка.

Щоб виділити код, який потрібно відслідковувати на виникнення винятків, використовують ключове слово **try**. Після слова **try** пишеться блок, в якому розміщується програмний код, де можливе виникнення винятку(помилки виконання):

**try** {

  програмний\_код

}

1. ***Розкрийте призначення та особливості роботи блоку catch.***

Основне призначення блоку cath – це продовження виконання програми при виникненні помилки. В інших мовах програмування при виникненні виключної ситуації програма одразу завершується. Для одного блоку try може існувати довільна кількість блоків cath, оскільки один і той самий код, що розміщений в блоці try може генерувати різні види виключних ситуацій. Найбільш спеціалізовані блоки обробки виключних ситуацій повинні йти першими.

Після інструкції **try** обов'язково має бути інструкція перехоплення винятків, яка позначається ключовим словом **catch**. Після слова **catch** в дужках (параметри) вказується перехоплений об'єкт винятку, який був створений у блоці **try**. Після параметрів пишеться блок коду, який має виконатися при перехопленні винятку:

**try** { /\* всередині цього блоку пишеться програмний код, який буде відслідковуватися на

виникнення винятків \*/

  програмний\_код

}

**catch** (об'єкт\_винятку) { /\* всередині цього блоку пишеться код, який має виконатися при перехоленні вказаного в параметрах об'єкту винятку \*/

  програмний\_ код

}

***10. Розкрийте призначення та особливості роботи блоку finally.***

Блок **finally**виконується після try-catch незалежно від того, чи виник виняток. Це необов’язковий блок, але якщо немає блоку catch, то блок finally необхідний.

У цьому блоці можна, наприклад, закрити файл, який відкрито в блоці try, як у наведеному нижче коді.

import java.io.FileWriter;

import java.io.IOException;

public class FinallyTest {

public static void main(String[] args) {

FileWriter writer = null;

try {

writer = new FileWriter("out.txt");

writer.write("Writing to the file!");

System.out.println("Файл записаний вдало.");

} catch (IOException e) {

System.out.println("Помилка запису у файл.");

e.printStackTrace();

} finally {

if ( writer != null ){

try{

writer.close();

} catch (IOException e) {

System. out.println("Помилка закриття файла.");

e.printStackTrace();

}

}

}

}

}

**Висновок:** оволодів навиками використання механізму виключень при написанні програм мовою Java.