Національний університет «Львівська Політехніка»

Інститут комп’ютерних технологій, автоматики та метрології

Кафедра електронних обчислювальних машин



Звіт

Про виконання лабораторної роботи №4

З дисципліни «Кросплатформлені засоби програмування»

**Виконав:**

студент групи КІ-305

Панасюк Т.Ю.

**Перевірив:**

Доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

**Тема:** Виключення.

**Мета:** 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 розробленогопакету.  
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 згідно варіанту:**

**Варіант: 21**

**Виконання:**

**Код програми:**

**package** KI305.Panasiuk.Lab5;

/\*\*

\* Class that implements a method for calculating the expression sin(3x-5) / ctg(2x).

\*

\* **@author** Taras Panasiuk

\* **@version** 1.0

\*/

**public** **class** Equations {

/\*\*

\* Method to calculate the sin(3x-5) / ctg(2x) expression.

\*

\* **@param** x Angle in degrees.

\* **@throws** CalcException

\* **@return** Result of the calculation.

\*/

**public** **double** calculate(**double** x) **throws** CalcException {

**double** y;

**try** {

**double** division = (1.0 / Math.*tan*(Math.*toRadians*(2 \* x)));

// Check for division by zero

**if** (division == 0) {

**throw** **new** CalcException("Exception reason: ctg(2x) = 0");

}

y = Math.*sin*(Math.*toRadians*(3 \* x - 5)) / division;

// Check for NaN and Infinity

**if** (Double.*isNaN*(y) || Double.*isInfinite*(y) || Double.*isInfinite*(-y)) {

**throw** **new** ArithmeticException();

}

// Check for illegal values of X for ctg calculation

**if** (x % 90 == 0) {

**throw** **new** CalcException("Exception reason: Illegal value of X for ctg calculation");

}

} **catch** (ArithmeticException ex) {

**throw** **new** CalcException("Unknown reason for the exception during expression calculation");

}

**return** y;

}

}

@SuppressWarnings("serial")

**public** **class** CalcException **extends** ArithmeticException {

/\*\*

\* Default constructor.

\*/

**public** CalcException() {

}

/\*\*

\* Constructor.

\*

\* **@param** cause Explanatory message explaining why the exception occurred.

\*/

**public** CalcException(String cause) {

**super**(cause);

}

}

/\*\*

\* Class CalcExceptionDriver implements a driver for the given expression calculator.

\*

\* **@author** Taras Panasiuk

\* **@version** 1.0

\*/

**public** **class** CalcExceptionDriver {

/\*\*

\* The main method, the starting point of the program.

\*/

**public** **static** **void** main(String[] args) {

**try** {

Scanner input = **new** Scanner(System.***in***);

PrintWriter fout = **new** PrintWriter(**new** File("result.txt"));

**try** {

**try** {

Equations eq = **new** Equations();

System.***out***.print("Enter X in degrees: ");

**double** res = eq.calculate(input.nextDouble());

System.***out***.print("sin(3x-5) / ctg(2x) = " + res);

fout.print(res);

} **finally** {

fout.flush();

fout.close();

input.close();

}

} **catch** (CalcException ex) {

System.***out***.print(ex.getMessage());

}

} **catch** (FileNotFoundException ex) {

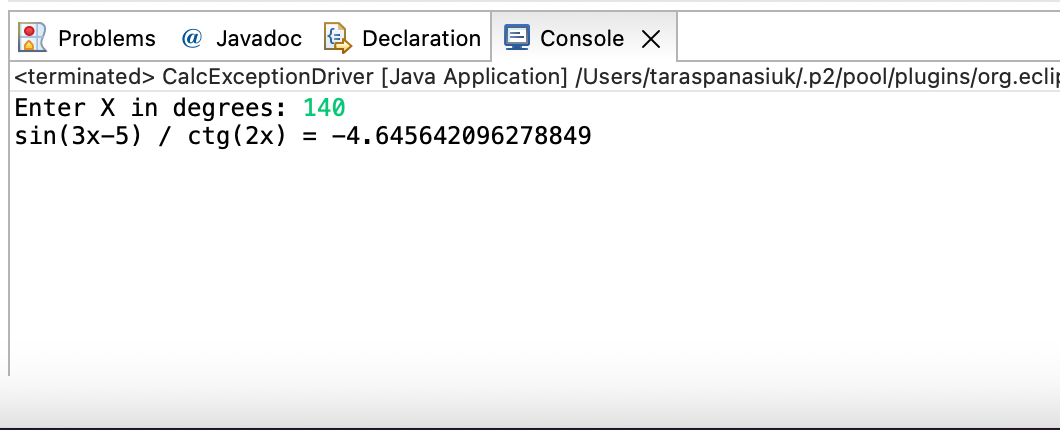
System.***out***.print("Exception reason: Perhaps wrong file path");

}

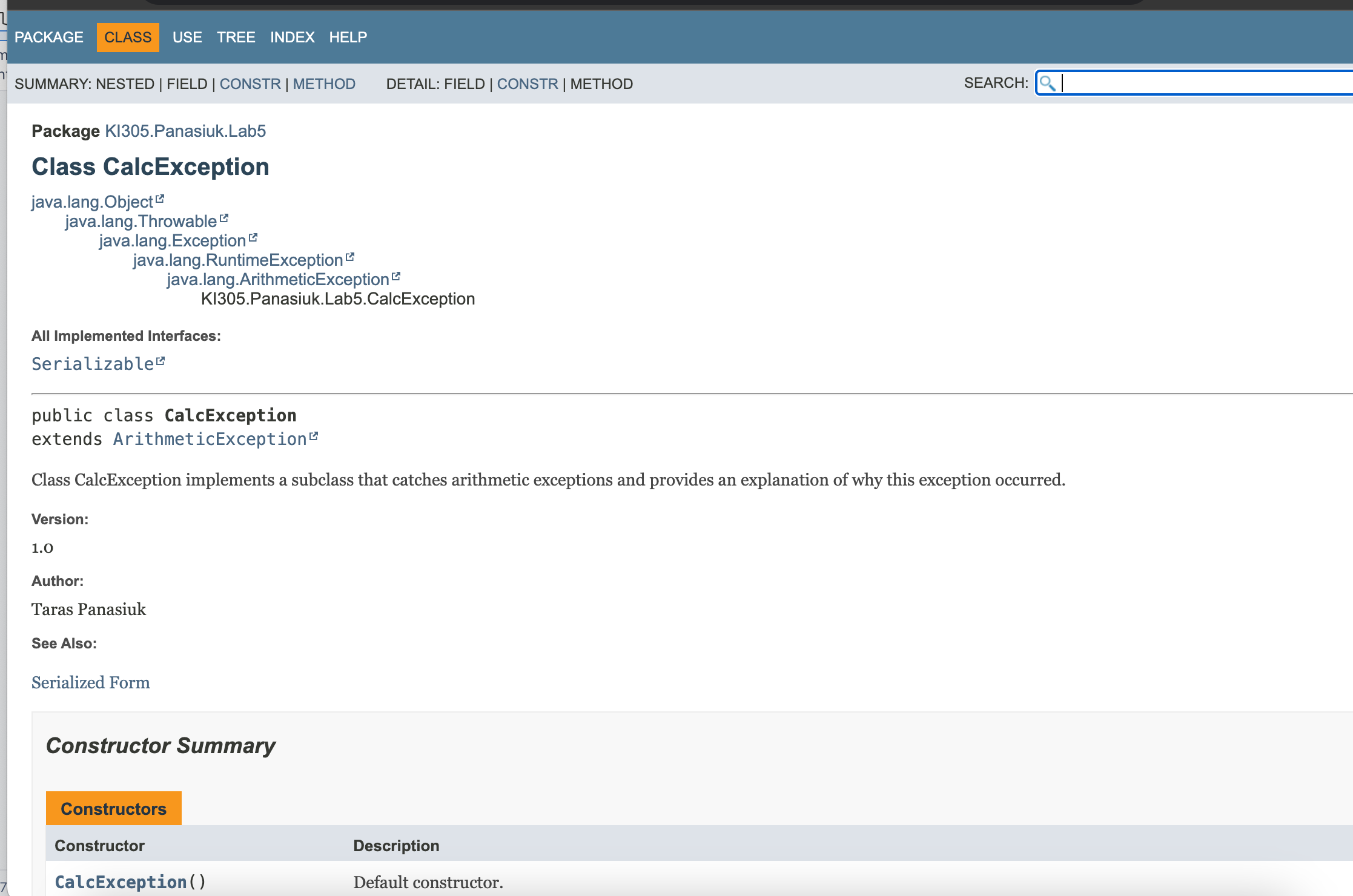
}

}

**Результат роботи програми у консолі та файлі:**



**Фрагмент генерації документації:**



**КОНТРОЛЬНІ ПИТАННЯ**

1. Дайте визначення терміну «виключення».

**Відповідь:**

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?

**Відповідь:**

Генерація виключень застосовується при:

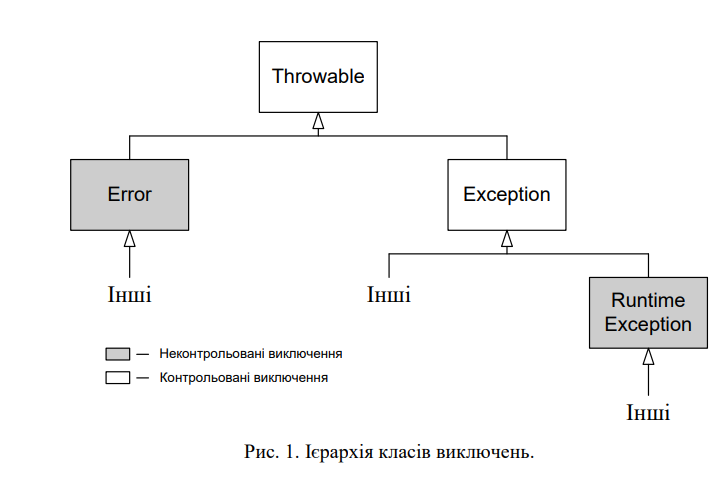
• помилках введення, наприклад, при введенні назви неіснуючого файлу або Інтернет адреси з подальшим зверненням до цих ресурсів, що призводить до генерації помилки системним програмним забезпеченням; • збоях обладнання;

• 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?

**Відповідь:**

****

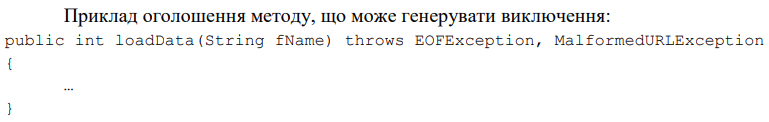
1. Як створити власний клас виключень?

**Відповідь:**

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.

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

**Відповідь:**

****

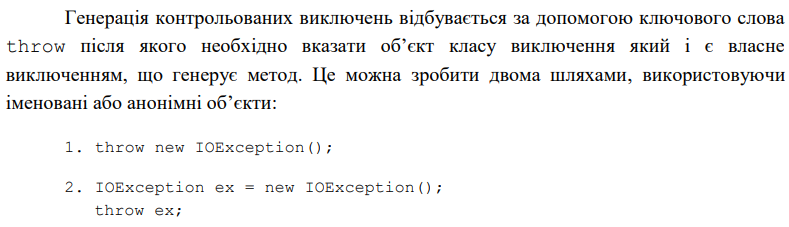
1. Які виключення слід вказувати у заголовках методів і коли?

**Відповідь:**

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?

**Відповідь:**

****

1. Розкрийте призначення та особливості роботи блоку try.

**Відповідь:**

Блок try використовується в Java для визначення частини коду, в якій може виникнути виняткова ситуація (виключення). Головна мета блоку try - це створення області коду, в якій можна перехоплювати виняткові ситуації та виконувати відповідний обробник (блок catch) для обробки цих виняткових ситуацій.

Робота блоку try:

Код, який може генерувати винятки, розміщується в блоку try.

Якщо виключення виникає в межах блоку try, виконання коду в try зупиняється, і контроль передається блокам catch, що відповідають типам винятків.

Якщо будь-який виняток відповідає одному з блоків catch, відповідний блок catch виконується.

Якщо виняток не відповідає жодному з блоків catch, воно переходить на рівень вище у викликуючий код або може завершити програму.

1. Розкрийте призначення та особливості роботи блоку catch.

**Відповідь:**

Блок catch використовується в Java разом з блоком try для обробки винятків (виключень), які можуть виникнути у блоку try. Основне призначення блоку catch - це перехоплення та обробка виняткових ситуацій.

Обробка винятків:

Кожен блок catch визначає, який тип винятку (або його підклас) можна обробити.

Якщо виняток відповідає типу, зазначеному у блоку catch, то відповідний блок catch виконується.

Після виконання обробника catch, виконання коду продовжується після всіх блоків catch (якщо вони є) або після блоку try, якщо виняток не був перехоплений.

Змінна винятку:

Змінна винятку (e1, e2, тощо) є об'єктом винятку, який містить інформацію про саме виняток, яке було викликано. Цю змінну можна використовувати для отримання додаткової інформації про виняток, наприклад, його повідомлення чи стек викликів.

Ця змінна доступна лише всередині відповідного блоку catch.

Порядок блоків catch:

Важливо зазначити блоки catch від найбільш конкретного типу винятку до більш загального. Це означає, що блоки catch для підкласів винятків повинні йти перед блоками для батьківських класів винятків.

Неправильний порядок може призвести до того, що деякі винятки не будуть перехоплені або оброблені неправильно.

1. Розкрийте призначення та особливості роботи блоку finally.

**Відповідь:**

Блок finally використовується в Java разом з блоком try-catch для створення коду, який завжди виконується, незалежно від того, чи виникала помилка, чи ні. Основне призначення блоку finally - це визначення коду, який потрібно виконати незалежно від того, чи виникла помилка у блоку try. Виконання незалежно від винятку:

Блок finally завжди виконується після блоку try-catch, незалежно від того, чи виникла помилка чи було виконано блок catch. Навіть якщо виключення було перехоплено і оброблено, код блоку finally виконається після цього.

Якщо в блоку try не виникає виключення і не потрібно викликати блок catch, код блоку finally також виконується.

Якщо у вас є блоки try та catch, але немає finally, код в блоках try та catch виконується, але не гарантується виконання коду після блоку try-catch.

Використання блоку finally:

Блок finally використовується для звільнення ресурсів, таких як закриття файлів, відкриття з'єднань з базою даних тощо, незалежно від того, чи виникали помилки в блоку try.

Це може бути використано для виконання певних завершальних операцій, які мають бути виконані навіть після виникнення виключення.

**Висновок:** на цій лабораторній роботі я оволодів навиками використання механізму виключень при написанні програм мовою Java. Та використав цей механізм на практиці при обчисленні формули.