**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**Інститут комп’ютерних технологій, автоматики та метрології**

**кафедра “Електронних обчислювальних машин”**

****

Звіт

з лабораторної роботи №4

дисципліни «Кросплатформні засоби програмування»

на тему: **«**Виключення**»**

Варіант 15

**Виконала:**

студентка групи КІ-303

Сухарник А.А.

**Прийняв:**

Іванов Ю.С.

Львів – 2024

ЛАБОРАТОРНА РОБОТА №4

**ВИКЛЮЧЕННЯ**

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

**Теоретичний матеріал**

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

Всі виключення в мові Java поділяються на контрольовані і неконтрольовані та спадкуються від суперкласу Throwable. Безпосередньо від цього суперкласу спадкуються 2 класи Error і Exception (див. рис. 1). Ієрархія класів, що спадкує клас Error, описує внутрішні помилки і ситуації, що пов’язані з браком ресурсів у системі підтримки виконання програм. Жоден об’єкт цього типу самостійно згенерувати неможна. При виникненні внутрішньої помилки можна лише відобразити повідомлення користувачу та спробувати коректно завершити виконання програми. Такі помилки є нечастими. Ієрархія класів, що спадкує клас Exception поділяється на клас RuntimeException та інші. Виключення типу RuntimeException виникають внаслідок помилок програмування. Всі інші помилки є наслідком непередбачених подій, що виникають під час виконання коректної програми, наприклад, помилок вводу/виводу. Класи, що спадкуються від Error та RuntimeException, відносяться до неконтрольованих виключень. Всі інші класи відносяться до контрольованих виключень. Лише контрольовані виключення можуть бути згенеровані програмістом у коді програми явно за допомогою ключового слова throw. Для всіх контрольованих виключень компілятор перевіряє наявність відповідних обробників.

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.

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. Приклад оголошення методу, що може генерувати виключення: public int loadData(String fName) throws EOFException, MalformedURLException { … } Слід зауважити, що оголошення всіх можливих виключень, які може генерувати метод, є поганим стилем програмування. Оголошувати слід лише всі контрольовані виключення. Якщо цього не зробити, то компілятор видасть повідомлення про помилку. Якщо метод оголошує, що він може генерувати виключення певного класу, то він може також генерувати виключення і його підкласів. Наприклад, оголошення методу int loadData(String fName) можна переписати наступним чином, не конкретизуючи, яке саме виключення похідне від класу IOException ми генеруватимемо

**Завдання (Варіант №15)**

1. Створити клас, що реалізує метод обчислення виразу (**y=cos(2x)/ctg(3x-1)**) 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.

**Виконання завдання**

**EquationsApp.java**

Package 303.Sukharnyk.lab4;  
  
import java.io.IOException;  
import java.util.Scanner;  
  
*/\*\*  
 \* Драйвер для запуску програми обчислення виразу.  
 \* <p>  
 \* Клас Main відповідає за ініціалізацію обчислювача виразів та взаємодію з користувачем через консоль.  
 \* Користувач вводить значення змінної x, після чого програма обчислює значення виразу та виводить результат.  
 \* Також результат зберігається у файл.  
 \* </p>  
 \*  
 \* @version 1.0  
 \*/*public class EquationsApp {  
  
 */\*\*  
 \* Точка входу до програми. Виконується отримання введеного користувачем значення,  
 \* обчислення результату виразу та запис результату у файл.  
 \*  
 \* @param args аргументи командного рядка  
 \*/* public static void main(String[] args) {  
 ExpressionCalculator calculator = new ExpressionCalculator();  
 Scanner scanner = new Scanner(System.*in*);  
  
 try {  
 // Отримання значення x від користувача  
 System.*out*.print("Введіть значення x: ");  
 double x = scanner.nextDouble();  
  
 // Обчислення результату  
 double result = calculator.calculate(x);  
 System.*out*.println("Результат: " + result);  
  
 // Запис результату у файл  
 String filePath = "result.txt";  
 calculator.writeResultToFile(result, filePath);  
 System.*out*.println("Результат записано у файл: " + filePath);  
  
 } catch (IllegalArgumentException e) {  
 System.*out*.println("Помилка: " + e.getMessage());  
 } catch (IOException e) {  
 System.*out*.println("Помилка запису у файл: " + e.getMessage());  
 } finally {  
 // Закриття сканера для уникнення витоку ресурсів  
 scanner.close();  
 }  
 }  
}

**ExpressionCalculator.java**

package 303.Sukharnyk.lab4;  
  
  
import java.io.FileWriter;  
import java.io.IOException;  
  
*/\*\*  
 \* Клас для обчислення виразу y = cos(2x)/ctg(3x-1).  
 \* Використовується для демонстрації механізму виключень і запису результатів у файл.  
 \*/*public class ExpressionCalculator {  
  
 */\*\*  
 \* Обчислює вираз y = cos(2x)/ctg(3x-1).  
 \*  
 \* @param x значення змінної x  
 \* @return результат обчислення виразу  
 \* @throws IllegalArgumentException якщо x = 0, оскільки вираз не визначений  
 \*/* public double calculate(double x) throws IllegalArgumentException {  
 if (x == 0) {  
 throw new IllegalArgumentException("Значення x не може бути 0, оскільки вираз не визначений.");  
 }  
 var cotangent = Math.cos(3 \* x - 1) / Math.sin(3 \* x - 1);  
  
 return Math.cos(2 \* x) / cotangent;  
  
 }  
  
 */\*\*  
 \* Записує результат обчислення у файл.  
 \*  
 \* @param result результат обчислення  
 \* @param filePath шлях до файлу  
 \* @throws IOException якщо виникає помилка при записі у файл  
 \*/* public void writeResultToFile(double result, String filePath) throws IOException {  
 try (FileWriter writer = new FileWriter(filePath)) {  
 writer.write("Результат обчислення: " + result);  
 }  
 }  
}

**Відповіді на контрольні питання**

1. Дайте визначення терміну «виключення». ***Виключення*** *— це подія, яка перериває нормальний хід виконання програми через виникнення помилки або іншої виняткової ситуації.*
2. У яких ситуаціях використання виключень є виправданим? *Використання виключень виправдане, коли необхідно обробити непередбачені помилки, наприклад, помилки вводу/виводу, некоректні дані, відсутність ресурсів.*
3. Яка ієрархія виключень використовується у мові Java?

*Throwable*

*Exception (контрольовані)*

*RuntimeException (неконтрольовані)*

*Error (критичні помилки)*

1. Як створити власний клас виключень?

*public class MyException extends Exception {*

*public MyException(String message) {*

*super(message);*

*}*

*}*

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

*public void myMethod() throws MyException {*

*// код*

*}*

1. Які виключення слід вказувати у заголовках методів і коли? *У заголовках методів слід вказувати контрольовані виключення (які наслідуються від Exception), якщо метод їх генерує. Неконтрольовані виключення (RuntimeException) можна не вказувати.*
2. Як згенерувати контрольоване виключення? *throw new MyException("Повідомлення про помилку");*
3. Розкрийте призначення та особливості роботи блоку try. ***Блок try*** *використовується для визначення ділянки коду, де можуть виникнути виключення.*
4. Розкрийте призначення та особливості роботи блоку catch. ***Блок catch*** *використовується для перехоплення та обробки виключення, що було згенероване в блоці try.*
5. Розкрийте призначення та особливості роботи блоку finally. ***Блок finally*** *виконується після завершення блоків try-catch, незалежно від того, чи виникло виключення, і використовується для звільнення ресурсів.*

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