

Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



Звіт

З лабораторної роботи №4

Варіант – 10

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Виключення»

Виконав: ст. гр. КІ-306

Миценко О. С.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2023

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

Завдання (варіант № 10)

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.

Вихідний код програми

ExpressionCalculator.java

```
package KI306.Mytsenko.Lab4;
```

```
import java.io.FileWriter;  
import java.io.IOException;  
import java.io.PrintWriter;
```

```
/**
```

```
 * Клас ExpressionCalculator виконує обчислення виразу та запис результатів у файл.
```

```
 * Вираз:  $y = \text{tg}(x) / \text{ctg}(x)$ 
```

```
 */
```

```
public class ExpressionCalculator {  
    private double x;
```

```
/**
```

```
 * Конструктор для створення об'єкта ExpressionCalculator зі значенням x.
```

```
 *
```

```
 * @param x Значення x, для якого буде обчислюватися вираз.
```

```
 */
```

```
public ExpressionCalculator(double x) {  
    this.x = x;  
}
```

```
/**
```

```
 * Обчислює вираз  $y = \text{tg}(x) / \text{ctg}(x)$ .
```

```
 *
```

```

* @return Результат обчислення виразу.
* @throws ArithmeticException Виникає, якщо виникає помилка при обчисленні виразу.
*/
public double calculateExpression() throws ArithmeticException, NewException {
    double tanX = Math.tan(x);
    double cotanX = 1 / Math.tan(x);
    if(x>20){
        throw new NewException("Значення більше за 20");
    }

    if (Double.isInfinite(tanX) || Double.isNaN(tanX) || Double.isInfinite(cotanX) || Double.isNaN(cotanX)) {
        throw new ArithmeticException("Вираз не визначений (tg(x) або ctg(x) мають недопустиме значення).");
    }

    if (Math.abs(cotanX) < 1e-6) {
        throw new ArithmeticException("Ділення на нуль у виразі (ctg(x) дуже мале).");
    }

    return tanX / cotanX;
}

/**
 * Записує результат обчислення виразу у файл "result.txt".
 *
 * @param result Результат обчислення виразу.
 * @throws IOException Виникає, якщо виникає помилка при записі у файл.
 */
public void saveResultToFile(double result) throws IOException {
    try (PrintWriter writer = new PrintWriter(new FileWriter("result.txt"))) {
        writer.println("Результат обчислення виразу: " + result);
    }
}
}

```

Main.java

```

package KI306.Mytsenko.Lab4;
import KI306.Mytsenko.Lab4.ExpressionCalculator;

import java.io.IOException;
import java.util.InputMismatchException;
import java.util.Scanner;

/**
 * Головний клас програми для обчислення виразу та збереження результату у файл.
 *
 * @author Oleksandr Mytsenko
 * @since 1.0
 */

```

```

public class Main {
    /**
     * Точка входу в програму.
     *
     * @param args Масив аргументів командного рядка.
     */
    public static void main(String[] args) throws IOException {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Введіть значення x: ");

        try {
            double x = scanner.nextDouble();

            ExpressionCalculator calculator = new ExpressionCalculator(x);
            double result = calculator.calculateExpression();
            calculator.saveResultToFile(result);
            System.out.println("Результат обчислення: " + result);
        } catch (ArithmeticException | InputMismatchException e) {
            System.err.println("Помилка обчислення: " + e.getMessage());
        }
    }
}

```

Результат виконання програми

```

"C:\Users\Oleksandr Mytsenko\.jdk\openjdk-20.0.2\bin\java.exe" "-javaagent:C:\Program File
Введіть значення x: 0
Помилка обчислення: Вираз не визначений (tg(x) або ctg(x) мають недопустиме значення).

```

Рис.1 – Скріншот виконання програми

Фрагмент згенерованої документації

The screenshot shows a web browser window displaying the Java documentation for the `ExpressionCalculator` class. The page is titled "Package K1306.Mytsenko.Lab4" and "Class ExpressionCalculator". It shows the class hierarchy, including `java.lang.Object` and `K1306.Mytsenko.Lab4.ExpressionCalculator`. The class is defined as `public class ExpressionCalculator extends Object`. A description of the class states: "Клас ExpressionCalculator виконує обчислення виразу та запис результатів у файл. Вираз: $y = \text{tg}(x) / \text{ctg}(x)$ ". The page also includes a "Constructor Summary" section with a constructor `ExpressionCalculator(double x)` and a "Method Summary" section with tabs for "All Methods", "Instance Methods", and "Concrete Methods". A warning message at the bottom of the browser window states: "Internet Explorer restricted this webpage from running scripts or ActiveX controls." The taskbar at the bottom shows the date and time as 12:06 AM on 9/23/2023.

Відповіді на контрольні запитання

1. Визначення терміну «виключення»: Виключення (або Exception) - це об'єкт, який виникає в Java в результаті виникнення помилкової або непередбачуваної ситуації під час виконання програми. Вони можуть виникати через помилки програміста, недійсні дані введені користувачем або інші непередбачувані обставини.
2. Ситуації використання виключень: Виключення виправдано використовувати, коли виникає потреба обробити помилки або непередбачені ситуації в програмі. Вони допомагають програмістам зберегти контроль над програмою і реагувати на помилки в елегантний спосіб, замість того, щоб допустити аварійне завершення програми.
3. Ієрархія виключень в Java: У Java існує ієрархія класів виключень, коренем якої є клас `java.lang.Throwable`. Він розділяється на дві основні гілки: `java.lang.Error` (помилки, які не рекомендується обробляти) і `java.lang.Exception` (виключення, які можна обробляти). Класи виключень повинні наслідуватися від `Exception` або його підкласів.
4. Створення власного класу виключень: Для створення власного класу виключень потрібно створити новий клас, який наслідується від класу `Exception` або одного з його підкласів. Зазвичай цей клас містить конструктори та може додавати додаткові поля та методи, які допомагають ідентифікувати або обробляти помилки.
5. Синтаксис оголошення методів, що можуть генерувати виключення: Методи, які можуть генерувати виключення, повинні бути оголошені з ключовим словом `throws`, і після нього слідує список класів виключень, які можуть бути викинуті. Наприклад:

```
public void doSomething() throws SomeException, AnotherException {  
    // код методу }  
}
```

6. Вказання виключень у заголовках методів і коли: Вказання виключень у заголовках методів (за допомогою `throws`) необхідне, коли метод може генерувати виключення, але не обробляє їх в самому методі. Це допомагає програмістам, які використовують цей метод, знати, які виключення можуть бути викинуті і як їх обробити.
7. Генерація контрольованого виключення: Для генерації контрольованого виключення використовуйте ключове слово `throw`, а потім створіть новий

об'єкт виключення та викидайте його. Наприклад: `throw new
MyException("Помилка в програмі");`

8. Призначення та особливості роботи блоку `try`: Блок `try` використовується для визначення коду, в якому можуть виникнути виключення. Якщо в блоку `try` виникає виключення, виконання програми переходить до блоку `catch` або `finally`. Основна мета блоку `try` - забезпечити обробку виключень.
9. Призначення та особливості роботи блоку `catch`: Блок `catch` використовується для обробки виключень, які виникли в блоку `try`. Він містить код для обробки виключення та зазвичай приймає один або кілька параметрів, які вказують на типи виключень, які може обробляти.
10. Призначення та особливості роботи блоку `finally`: Блок `finally` використовується для виконання коду, який завжди має бути виконаним, незалежно від того, чи виникло виключення в блоку `try`. Це корисно, наприклад, для вивільнення ресурсів (наприклад, закриття файлу), які повинні бути вивільнені навіть у випадку виникнення виключення.

Висновок Ознайомився з виключеннями, функціями та виводом у файл у мові Java. Розробив програму яка обчислює вираз та записує результат у файл.