

Міністерство освіти і науки України  
Національний університет «Львівська політехніка»  
Кафедра «Електронних обчислювальних машин»



Звіт  
з лабораторної роботи № 4  
з дисципліни: «Кросплатформенні засоби програмування»  
на тему: «“ВИКЛЮЧЕННЯ.”»

**Виконав:**

студент групи *KI-306*

*Олесько Б. А.*

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

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.

Варіант 11  $y = \text{ctg}(x)/\text{tg}(x)$

## Код програми

```
/**
 * EquationsApp is a simple Java application for calculating the result of the equation
 *  $y = \text{ctg}(x)/\text{tg}(x)$  and writing the result to a file specified by the user.
 *
 * This program prompts the user for a file name, an integer value for 'x', and then
 * calculates the result of the equation based on the value of 'x'. It handles exceptions
 * related to file operations and equation calculation.
 *
 * @author Olesko Name
 * @version 1.0
 */
package KI306lab4Olesko;

import java.util.Scanner;
import java.io.*;
import static java.lang.System.out;

class EquationsApp {

    public static void main(String[] args) {
        out.print("Enter file name: ");
        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("Enter X: ");
            int x = in.nextInt();
```

```

        double result = eq.calculate(x);
        System.out.println(result);
        fout.print(result);
        fout.flush();
        fout.close();
    } catch (FileNotFoundException ex) {
        out.print("Exception reason: File not found");
    } catch (CalcException ex) {
        out.print(ex.getMessage());
    } finally {
        if (fout != null) {
            fout.close();
        }
    }
}
}
}

```

```

package KI306lab4Olesko;
/**
 * The Equations class provides a method to calculate the result of the equation
 *  $y = \cot^2(x)$  based on the input angle 'x' (in degrees).
 *
 * This class handles exceptions related to invalid input values and provides
 * a method for calculating the square of the cotangent (ctg) of the given angle 'x'.
 * It also includes logic to normalize 'x' if it exceeds 360 degrees.
 *
 * @author Olesko Bohdan
 * @version 1.0
 */
public class CalcException extends ArithmeticException {
    public CalcException() {}
    public CalcException(String cause) {
        super(cause);
    }
}

class Equations {
    /**
     * Calculate the square of the cotangent (ctg2) of the given angle 'x' (in degrees).
     *
     * @param x The input angle in degrees.
     * @return The square of the cotangent of 'x'.
     * @throws CalcException If 'x' is an invalid value for cotangent calculation.
     */
    public double calculate(int x) throws CalcException {
        double y, rad;
        int i=1;
        if(x>360) {
            for(i=1 ;x<i*360;i++) {}
            x=x-(i*360);
        }
        rad = Math.toRadians(x);
        try {
            double tanValue = Math.tan(rad);
            if (rad==Math.PI ||x==0 ||x==360||x==90||x==270) {
                throw new ArithmeticException();
            }
            y = 1.0 / (tanValue * tanValue);
        } catch (ArithmeticException ex) {
            throw new CalcException("Exception reason: Illegal value of X " +x+ " for cotangent calculation");
        }
        return y;
    }
}

```

## Результат програми

```
Enter file name: 1
Enter X: 165
13.928203230275482
|
```

## Документація

Package `KI306lab4Olesko`

### Class `EquationsApp`

`java.lang.Object`  
`KI306lab4Olesko.EquationsApp`

class `EquationsApp`  
extends `Object`

#### Constructor Summary

##### Constructors

Constructor	Description
-------------	-------------

<code>EquationsApp()</code>	
-----------------------------	--

#### Method Summary

##### All Methods

##### Static Methods

##### Concrete Methods

Modifier and Type	Method	Description
static void	<code>main(String[] args)</code>	

##### Methods inherited from class `java.lang.Object`

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `toString`, `wait`, `wait`, `wait`

## Відповіді на контроль завдання

1. Виключення (або exception) - це подія, яка виникає під час виконання програми і вказує на помилку або непередбачену ситуацію.
2. Виключення виправдані, коли програма не може нормально продовжувати своє виконання через помилку або непередбачену умову, і коли це потрібно повідомити процесу обробки помилок.
3. У Java існує ієрархія класів виключень, з вершини якої є клас `Throwable`. Два основних підкласи цього класу - `Error` і `Exception`. `Exception` ділиться на контрольовані (checked) та неконтрольовані (unchecked) виключення.
4. Для створення власного класу виключень потрібно створити клас, який успадковує або реалізує класи або інтерфейси з ієрархії `Throwable`.

5. Вказуємо вибрані виключення у списку throws в заголовку методу, наприклад: `public void myMethod() throws MyException { ... }`.

6. Вказувати виключення в заголовках методів слід тільки тоді, коли це контрольовані виключення (checked exceptions) і метод не обробляє їх самостійно.

7. Для генерації контрольованого виключення використовуємо ключове слово `throw` разом із відповідним об'єктом виключення, наприклад: `throw new MyException("Повідомлення про помилку");`.

8. Блок `try` використовується для обгортання коду, який може викинути виключення. Він визначає область, в якій можуть виникнути помилки.

9. Блок `catch` використовується для обробки виключень, які були викинуті в блоку `try`. У блоках `catch` вказуються типи виключень, які можна обробляти, і код для їх обробки.

10. Блок `finally` використовується для виконання коду, незалежно від того, чи виникло виключення в блоку `try`. Він використовується для ресурсів і фіналізації, і виконується завжди, навіть якщо було викинуто виключення.

## Висновок

На даній лабораторній роботі я навчився працювати і обробляти різні види помилок.