

Міністерство освіти і науки України  
Національний університет “Львівська політехніка”

Кафедра ЕОМ



### **Звіт**

з лабораторної роботи №5

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Виключення»

Виконав: ст.гр. КІ-34

Степанов А. О.

Прийняв:

викл. каф. ЕОМ

Іванов Ю. С.

**Львів 2022**

**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.Lab5 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленої програми.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

**Варіант 21**

$$21. y = \sin(3x - 5) / \operatorname{ctg}(2x)$$

**Лістинг програми:**

**Файл EquationsApp.java**

```
package KI34.Stepanov.Lab5;
import java.util.*;
import java.io.*;

/**
 * Class <code>EquationsApp</code> Implements driver for Equations class
 * @author Andriy Stepanov
 * @version 1.0
 */
public class EquationsApp {
    /**
     * @param args function's parameters
     */
    public static void main(String[] args)
    {
        try
        {
            Scanner in = new Scanner(System.in);
            PrintWriter fout = new PrintWriter(new File("Result"));
            try
            {
                try
                {
                    Equations eq = new Equations();
                    System.out.print("Enter X: ");
                    fout.print(eq.calculate(in.nextInt()));
                }
                finally
                {
                    // Цей блок виконається за будь-яких обставин
                    fout.flush();
                    fout.close();
                }
            }
        }
    }
}
```

```

    }
    catch (CalcException ex)
    {
        // Блок перехоплює помилки обчислень виразу
        System.out.print(ex.getMessage());
    }
}
catch (FileNotFoundException ex)
{
    // Блок перехоплює помилки роботи з файлом навіть якщо вони
    // виникли у блоці finally
    System.out.print("Exception reason: Perhaps wrong file path");
}
}
}

```

## Файл Equations.java

```

package KI34.Stepanov.Lab5;

/**
 * Class <code>Equations</code> implements method for  $\sin(3x-5)/\operatorname{ctg}(2x)$ 
 * expression
 * calculation
 *
 * @author Andriy Stepanov
 * @version 1.0
 */
public class Equations {
    /**
     * Method calculates the  $\sin(3x-5)/\operatorname{ctg}(2x)$  expression
     *
     * @param x Angle in degrees
     * @throws CalcException
     */
    public double calculate(int x) throws CalcException {
        double y, radSin, radCtg;
        radSin = (3 * x - 5) * Math.PI / 180.0;
        radCtg = 2 * x * Math.PI / 180.0;
        try {
            y = Math.sin(radSin) / (1.0 / Math.tan(radCtg));

            // Якщо результат не є числом, то генеруємо виключення
            if (y == Double.NaN || y == Double.NEGATIVE_INFINITY ||
                y == Double.POSITIVE_INFINITY || x % 90 == 0 || x == 45 ||
                (2 * x) % 270.0 == 0)
                throw new ArithmeticException();
        } catch (ArithmeticException ex) {
            // створимо виключення вищого рівня з поясненням причини
            // виникнення помилки
            if (radCtg % Math.PI / 2.0 == 0)
                throw new CalcException("Exception reason: Illegal value of X
for cotangent calculation");
            else if (x == 45 || (2 * x) % 270.0 == 0)
                throw new CalcException("Exception reason: X = 0");
            else
                throw new CalcException("Unknown reason of the exception during
exception calculation");
        }

        return y;
    }
}

```

## Файл CalcException.java

```
package KI34.Stepanov.Lab5;

/**
 * Class <code>CalcException</code> more precises ArithmeticException
 *
 * @author Andriy Stepanov
 * @version 1.0
 */
public class CalcException extends ArithmeticException {
    public CalcException() {
    }

    public CalcException(String cause) {
        super(cause);
    }
}
```

### Результат виконання програми:

```
Enter X: 90
Exception reason: Illegal value of X for cotangent calculation
Process finished with exit code 0
|
```

*Виключення для неприпустимого значення для котангенса*

```
Enter X: 45
Exception reason: X = 0
Process finished with exit code 0
```

*Виключення при діленні на нуль*

```
Enter X: 20

Process finished with exit code 0
|
```

*Успішне виконання програми*



Result - Notepad

File Edit Format View Help

0.687350178241008

*Успішний запис результату у файл*

# Package KI34.Stepanov.Lab5

package KI34.Stepanov.Lab5

| All Classes and Interfaces | Classes  | Exception Classes |
|----------------------------|--|-------------------|
| Class                      | Description  |                   |
| CalcException              | Class CalcException more precises ArithmeticException                                    |                   |
| Equations                  | Class Equations implements method for $\sin(3x-5)/\text{ctg}(2x)$ expression calculation |                   |
| EquationsApp               | Class EquationsApp Implements driver for Equations class                                 |                   |

## Згенерована документація

Package KI34.Stepanov.Lab5

### Class CalcException

java.lang.Object<sup>Ⓜ</sup>  
  java.lang.Throwable<sup>Ⓜ</sup>  
    java.lang.Exception<sup>Ⓜ</sup>  
      java.lang.RuntimeException<sup>Ⓜ</sup>  
        java.lang.ArithmeticException<sup>Ⓜ</sup>  
          KI34.Stepanov.Lab5.CalcException

All Implemented Interfaces:

Serializable<sup>Ⓜ</sup>

```
public class CalcException
extends ArithmeticExceptionⓂ
```

Class CalcException more precises ArithmeticException

Version:

1.0

Author:

Andriy Stepanov

See Also:

Serialized Form

#### Constructor Summary

| Constructors                             | Description |
|--|-------------|
| CalcException()                          |             |
| CalcException(String <sup>Ⓜ</sup> cause) |             |

#### Method Summary

| Methods inherited from class java.lang.Throwable <sup>Ⓜ</sup>   |
|---|
| addSuppressed <sup>Ⓜ</sup> , fillInStackTrace <sup>Ⓜ</sup> , getCause <sup>Ⓜ</sup> , getLocalizedMessage <sup>Ⓜ</sup> , getMessage <sup>Ⓜ</sup> , getStackTrace <sup>Ⓜ</sup> , getSuppressed <sup>Ⓜ</sup> , initCause <sup>Ⓜ</sup> , printStackTrace <sup>Ⓜ</sup> , printStackTrace <sup>Ⓜ</sup> , printStackTrace <sup>Ⓜ</sup> , printStackTrace <sup>Ⓜ</sup> , setStackTrace <sup>Ⓜ</sup> , toString <sup>Ⓜ</sup> |

## Інформація про клас CalcException

Class Equations

java.lang.Object<sup>ⓘ</sup>  
KI34.Stepanov.Lab5.Equations

```
public class Equations
extends Objectⓘ
```

Class Equations implements method for  $\sin(3x-5)/\operatorname{ctg}(2x)$  expression calculation

Version:  
1.0

Author:  
Andriy Stepanov

Constructor Summary

| Constructors |             |
|--------------|-------------|
| Constructor  | Description |
| Equations()  |             |

Method Summary

| All Methods       |                  |  |
|-------------------|------------------|--|
| Instance Methods  |                  |  |
| Concrete Methods  |                  |  |
| Modifier and Type | Method           | Description  |
| double            | calculate(int x) | Method calculates the $\sin(3x-5)/\operatorname{ctg}(2x)$ expression |

Methods inherited from class java.lang.Object<sup>ⓘ</sup>

equals<sup>ⓘ</sup>, getClass<sup>ⓘ</sup>, hashCode<sup>ⓘ</sup>, notify<sup>ⓘ</sup>, notifyAll<sup>ⓘ</sup>, toString<sup>ⓘ</sup>, wait<sup>ⓘ</sup>, wait<sup>ⓘ</sup>, wait<sup>ⓘ</sup>

Інформація про клас Equations

Constructor Details

| Equations          |
|--------------------|
| public Equations() |

Method Details

| calculate  |
|--|
| public double calculate(int x)<br>throws CalcException               |
| Method calculates the $\sin(3x-5)/\operatorname{ctg}(2x)$ expression |
| Parameters:  |
| x - Angle in degrees   |
| Throws:  |
| CalcException  |

Інформація про клас Equations

Package KI34.Stepanov.Lab5

## Class EquationsApp

java.lang.Object<sup>Ⓢ</sup>  
KI34.Stepanov.Lab5.EquationsApp

```
public class EquationsApp  
extends ObjectⓈ
```

Class EquationsApp Implements driver for Equations class

Version:

1.0

Author:

Andriy Stepanov

### Constructor Summary

#### Constructors

| Constructor    | Description |
|----------------|-------------|
| EquationsApp() |             |

### Method Summary

#### All Methods Static Methods Concrete Methods

| Modifier and Type | Method                            | Description |
|-------------------|-----------------------------------|-------------|
| static void       | main(String <sup>Ⓢ</sup> [] args) |             |

#### Methods inherited from class java.lang.Object<sup>Ⓢ</sup>

equals<sup>Ⓢ</sup>, getClass<sup>Ⓢ</sup>, hashCode<sup>Ⓢ</sup>, notify<sup>Ⓢ</sup>, notifyAll<sup>Ⓢ</sup>, toString<sup>Ⓢ</sup>, wait<sup>Ⓢ</sup>, wait<sup>Ⓢ</sup>, wait<sup>Ⓢ</sup>

## Інформація про клас EquationsApp

### Constructor Details

#### EquationsApp

```
public EquationsApp()
```

### Method Details

#### main

```
public static void main(StringⓈ[] args)
```

Parameters:

args - function's parameters

## Інформація про клас EquationsApp

## Відповіді на 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.

## **2. 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

## **3. Яка ієрархія виключень використовується у мові Java?**

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`.

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.

## **4. Як створити власний клас виключень?**

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.



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

Приклад оголошення методу, що може генерувати виключення:

```
public int loadData(String fName) throws EOFException, MalformedURLException  
{  
    ...  
}
```

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

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.

## **7. Як згенерувати контрольоване виключення?**

Генерація контрольованих виключень відбувається за допомогою ключового слова `throw` після якого необхідно вказати об'єкт класу виключення який і є власне виключенням, що генерує метод. Це можна зробити двома шляхами, використовуючи іменовані або анонімні об'єкти:

```
throw new IOException();
```

```
IOException ex = new IOException();
```

```
throw ex;
```

## **8. Розкрийте призначення та особливості роботи блоку `try`.**

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  
}
```

## 9. Розкрийте призначення та особливості роботи блоку **catch**.

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 перехопленні вказаного в параметрах об'єкту винятку */
```

```
    програмний_код
```

```
}
```

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

Інструкція **finally** застосовується після інструкцій **try-catch** в тому випадку, коли незалежно від того, чи перехоплюється виняток, чи не перехоплюється, чи програма припиняє своє виконання, потрібно щоб виконався певний програмний код

```
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 перехопленні вказаного в параметрах об'єкту винятку */
```

```
    програмний_код
```

```
}
```

```
finally { /* код в інструкції finally виконається незалежно від того, чи було перехоплено помилку, чи ні */
```

```
    програмний_код
```

```
}
```

**Висновок:**

На цій лабораторній роботі я оволодів навиками використання механізму виключень при написанні програм мовою Java.