

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

Скалій Т.В.

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 20**

$$y = \text{tg}(x) \text{ctg}(2x)$$

**Лістинг програми:**

**EquationsApp.java**

```
package KI34.Skalii.Lab5;
import java.util.Scanner;
import java.io.*;
import static java.lang.System.out;

/**
 * Class <code>EquationsApp</code> Implements driver for Equations class
 * @author EOM Stuff
 * @version 1.0
 */
public class EquationsApp {
    public static void main(String[] args) {
        System.out.println("Success");
        try
        {
            out.print("Enter file name: ");
            Scanner in = new Scanner(System.in);
            String fName = in.nextLine();
            PrintWriter fout = new PrintWriter(new File(fName));
            try
            {
                Equations eq = new Equations();
                out.print("Enter X: ");
                int x = in.nextInt();
                fout.print(eq.calculate(x));
                out.print(eq.calculate(x));
            }
            finally
            {
                fout.flush();
                fout.close();
            }
        }
    }
}
```

```

        }
    }
    catch (CalcException ex)
    {
        out.print(ex.getMessage());
    }
}
catch (FileNotFoundException ex)
{
    out.print("Exception reason: Perhaps wrong file path");
}
}
}

```

## Equations.java

```

package KI34.Skalii.Lab5;
/**
 * Class <code>Equations</code> implements method for  $y=\text{tg}(x)\text{ctg}(2x)$  expression
 calculation
 * @author Tetiana Skalii
 * @version 1.0
 */
class Equations
{
    public double calculate(int x) throws CalcException
    {
        double y, rad;
        rad = x * Math.PI / 180.0;
        try
        {
            y = Math.tan(rad) * ( 1 / Math.tan(rad * 2));
            if (x==90 || x== -90 || x==0 || x== -180 || x==180 || 2*x == 90 || 2*x == -
90 || 2*x == 180 || 2*x == -180 )
                throw new ArithmeticException();
        }
        catch (ArithmeticException ex)
        {
            if (x==90 || x== -90 || x== -180 || x==180 || 2*x == 90 || 2*x == -90 || 2*x
== 180 || 2*x == -180 )
                throw new CalcException("Exception reason: Illegal value of X for
tangent calculation");

            else if (x==0)
                throw new CalcException("Exception reason: X = 0");
            else
                throw new CalcException("Unknown reason of the exception during
exception calculation");
        }
        return y;
    }
}

```

## CalcException.java

```
package KI34.Skalii.Lab5;
/**
 * Class <code>CalcException</code> more precises ArithmeticException
 * @author Tetiana Skalii
 * @version 1.0
 */

class CalcException extends ArithmeticException
{
    public CalcException(){}

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

### Результат виконання програми:

```
<terminated> EquationsApp [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\jav
Success
Enter file name: Tetiana_Skalii_KI34
Enter X: 90
Exception reason: Illegal value of X for tangent calculation
```

*Рис.1.Виключення для неприпустимого значення для котангеса*

```
<terminated> EquationsApp [Java Application] C:\Pr
Success
Enter file name: Tetiana_Skalii_KI34
Enter X: 0
Exception reason: X = 0
```

*Рис.2.Виключення при значенні нуля*

```
<terminated> EquationsApp [Java Application] C:\Progr
Success
Enter file name: Tetiana_Skalii_KI34
Enter X: 10
0.48445439793711836
```

*Рис.3. Результат виконання програми*



Tetiana\_Skalii\_KI34: Блокнот

Файл Редагування Формат Вигляд Довідка

0.4337628342841029

Рис.4.Результат виконання програми у файлі

## Згенерована документація

### Package KI34.Skalii.Lab5

package KI34.Skalii.Lab5

All Classes and Interfaces	Classes	Exception Classes
Class	Description	
<b>CalcException</b>	Class CalcException more precises ArithmeticException	
<b>Equations</b>	Class Equations implements method for $y = \tan(x) \cdot \tan(2x)$ expression calculation	
<b>EquationsApp</b>	Class EquationsApp Implements driver for Equations class	

Рис.5.Вміст вкладки Package

```
class CalcException
extends ArithmeticException{}

Class CalcException more precises ArithmeticException

Version:
1.0

Author:
Tetiana Skalii
```

#### Constructor Summary

##### Constructors

Constructor	Description
CalcException()	
CalcException(String cause)	

#### Method Summary

##### Methods inherited from class java.lang.Throwable

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>, setStackTrace<sup>®</sup>, toString<sup>®</sup>

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

clone<sup>®</sup>, equals<sup>®</sup>, finalize<sup>®</sup>, getClass<sup>®</sup>, hashCode<sup>®</sup>, notify<sup>®</sup>, notifyAll<sup>®</sup>, wait<sup>®</sup>, wait<sup>®</sup>, wait<sup>®</sup>

Рис.6.Вміст вкладки Class(CalcException)

```
class Equations
extends Object
```

Class Equations implements method for  $y = \text{tg}(x) \cdot \text{ctg}(2x)$  expression calculation

Version:

1.0

Author:

Tetiana Skalii

### Constructor Summary

#### Constructors

Constructor	Description
<code>Equations()</code>	

### Method Summary

#### All Methods

#### Instance Methods

#### Concrete Methods

Modifier and Type	Method	Description
double	<code>calculate(int x)</code>	

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

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

### Constructor Details

Рис.7.Вміст вкладки Class(Equations)

```
public class EquationsApp
extends Object
```

Class EquationsApp Implements driver for Equations class

Version:

1.0

Author:

EOM Stuff

### 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`

Рис.8.Вміст вкладки Class(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 перехоленні 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 оволоділа навиками використання механізму виключень при написанні програм мовою Java.