Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

ВИКЛЮЧЕННЯ

Методичні вказівки
до лабораторної роботи № 4 з курсу “Кросплатформні засоби програмування”
для студентів базового напряму 6.123 - “Комп’ютерна інженерія”

Затверджено
на засіданні кафедри
”Електронні обчислювальні машини”
Протокол № 1 від 28.08.2023 р.

Львів – 2023

Виключення: Методичні вказівки до лабораторної роботи № 4 з курсу “Кросплатформні засоби програмування” для студентів базового напряму 6.123 - “Комп’ютерна інженерія” / Укладач: Олексів М.В. – Львів: Національний університет “Львівська політехніка”, 2023, 14 с.

Укладач Олексів М. В., к.т.н.

Рецензент

Відповідальний за випуск: Мельник А. О., професор, завідувач кафедри

ВИКЛЮЧЕННЯ

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.

Ієрархія класів виключень

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.

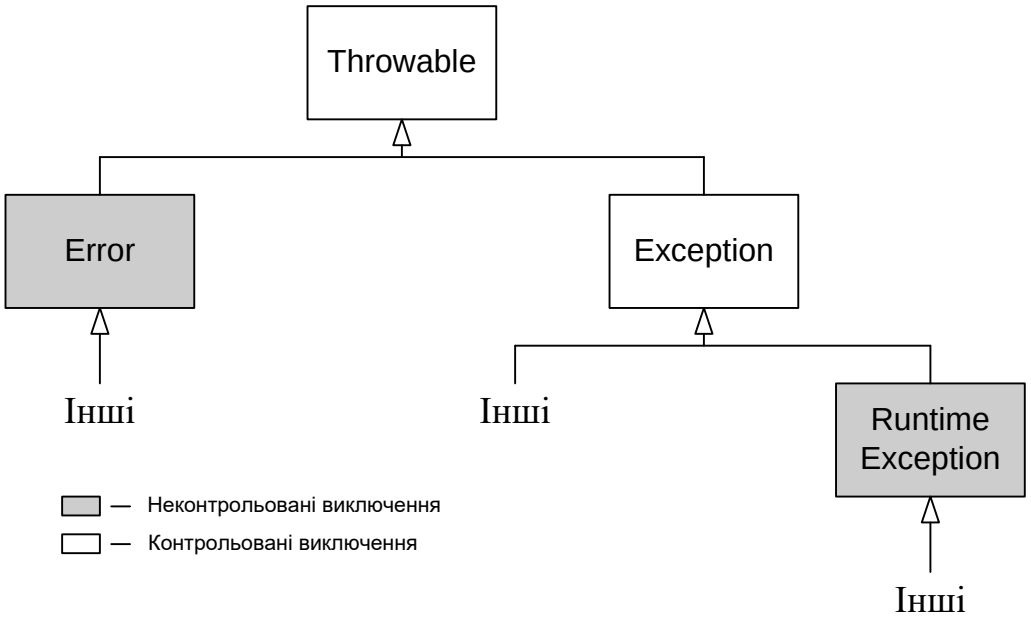

Рис. 1. Ієрархія класів виключень.

Створення власних класів виключень

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.

Приклад власного класу виключень:

```
class FileFormatException extends IOException
{
      public FileFormatException()
      {}

      public FileFormatException(String message)
      {
            /*
            Даний виклик конструктора суперкласу дозволяє
            використовувати метод toString() класу Throwable
            */
            super(message);
      }
}
```

Оголошення контрольованих виключень

Виключення можуть генеруватися лише методами. Якщо метод може генерувати виключення певного класу, то назву цього класу слід вказати в заголовку методу після

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:

```
public int loadData(String fName) throws IOException, MalformedURLException
{
      …
}
```

Проте зауважимо, що метод підкласу не може генерувати більш загальне контрольоване виключення, ніж метод суперкласу, що перевизначається. Виключення можуть лише конкретизуватися або не виникати взагалі. Зокрема, якщо метод суперкласу не генерує контрольованих виключень взагалі, то і метод підкласу не може їх генерувати. Проте в цьому випадку виключення можуть бути згенеровані і перехоплені в середині такого методу.

Генерація контрольованих виключень

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:

```
1. throw new IOException();

2. IOException ex = new IOException();
   throw ex;
```

Деякі класи контрольованих виключень мають конструктори, що приймають рядок з описом причини виникнення виключення. Такі конструктори корисно застосовувати для полегшення пошуку місця виникнення помилки під час виконання програми.

Перехоплення виключень

Перехоплення виключень здійснюється з метою коректного опрацювання критичних помилок повідомлення про які були згенеровані у процесі виконання програми. Якщо виключення виникає і ніде не перехоплюється, то програма припиняє свою роботу і виводить на консоль повідомлення про тип виключення та вміст стеку. Графічні програми

(аплети і прикладні програми) виводять те ж саме повідомлення, повертаючись після цього у цикл обробки користувацького інтерфейсу.

Перехоплення виключень відбувається за допомогою блоків `try/catch/finally`, що мають такий синтаксис:

```
try
{
      // етап 1
      Код, що може згенерувати виключення
      // етап 2
}
catch (тип 1 виключення)
{
      // етап 3
      Опрацювання виключення типу 1
      // етап 4
}
catch (тип 2 виключення)
{
      Опрацювання виключення типу 2
}
....
catch (тип N виключення)
{
      Опрацювання виключення типу N
}
finally
{
      // етап 5
      Код, що виконується за будь-яких обставин
}
// етап 6
```

Блоки `catch` і `finally` у цій конструкції є необов'язковими.

Розглянемо можливі варіанти роботи цієї конструкції у різних ситуаціях.

1. Якщо код у блоці `try` не генерує ніяких виключень, то програма спочатку повністю виконає блок `try`, а потім блок `finally`. Тобто виконаються етапи 1, 2, 5 і 6.

2. Якщо код у блоці `try` згенерував виключення, то подальше виконання коду в цьому блоці припиняється і відбувається пошук блоку `catch` тип у заголовку якого співпадає з типом виключення після чого виконується блок `finally`. Таким чином виконуються етапи 1, 3, 4, 5 і 6.

Якщо виключення генерується у блоці `catch`, то після виконання блоку `finally` керування передається викликаючому методу і виконуються лише етапи 1, 3 та 5.

3. Якщо виключення не опрацьовується у жодному з блоків `catch`, то виконується блок `finally` і керування передається викликаючому методу. Таким чином виконуються етапи 1 і 6.

4. Якщо ж блоки `finally` і `catch` відсутні, то керування передається викликаючому методу.

Мова Java дає розробнику вибір або самому перехопити і опрацювати виключення у методі, або делегувати це право іншому методу. Як правило, слід перехоплювати лише ті виключення, які ви самі можете опрацювати, або, які були створені вами. Решту виключень слід передавати далі по ланцюгу викликаних методів.

Для того щоб передати виключення далі по ланцюгу викликаних методів у заголовку методу, що розробляється, за допомогою ключового слова `throws` слід вказати типи виключень, що генерує цей метод але не опрацьовує, передаючи це право викликаючому методу. Якщо ж метод містить код опрацювання виключень, то вказувати типи виключень, що перехоплюються у методі, в заголовку методу не треба.

Якщо ви перевизначаєте метод суперкласу, що не генерує виключень, то ви зобов'язані перехоплювати всі контрольовані виключення, що генеруються в цьому методі.

Виключення можна генерувати у блоці `catch` створюючи цим самим ланцюг виключень. Зазвичай це робиться тоді, коли розробнику необхідно змінити тип виключення на тип виключення вищого рівня. Це може бути корисним, наприклад, тоді коли розробляється підсистема і розробнику, що використовуватиме її в подальшому слід знати, що помилка відбулася саме у цій підсистемі, а тип помилки значення не має. При цьому, у виключені вищого рівня за допомогою методу `setCause(Throwable)` можна розмістити первинне виключення як причину виникнення помилки. Одержати первинне виключення з виключення вищого рівня можна за допомогою методу `Throwable getCause()`. Для одержання повідомлення, що розміщено у виключені, слід використати метод `getMessage()`. Фактичний тип об'єкту виключення можна отримати за допомогою рефлексії: `назваОб'єкту.getClass().getName()`.

Блоки `try/catch` і `try/finally` рекомендовано розділяти розміщуючи у внутрішньому блоці `try/finally` код для звільнення ресурсів, а у зовнішньому `try/catch` код для опрацювання помилок. Це дозволить коректно опрацьовувати ситуацію, коли виключна ситуація генерується у блоці `finally` і не опрацьовується в подальшому ніяким з блоків `catch`.

Розміщення оператора `return` у блоці `finally` може призвести до неочікуваних результатів, тому цього слід уникати.

КОНТРОЛЬНІ ПИТАННЯ

1. Дайте визначення терміну «виключення».
2. У яких ситуаціях використання виключень є виправданим?
3. Яка ієрархія виключень використовується у мові Java?
4. Як створити власний клас виключень?
5. Який синтаксис оголошення методів, що можуть генерувати виключення?
6. Які виключення слід вказувати у заголовках методів і коли?
7. Як згенерувати контрольоване виключення?
8. Розкрийте призначення та особливості роботи блоку try.
9. Розкрийте призначення та особливості роботи блоку catch.
10. Розкрийте призначення та особливості роботи блоку finally.

ЛІТЕРАТУРА

1. Schildt H. Java: The Complete Reference, 12th Edition. / Herbert Schildt. – McGraw Hill, 2021. – 1280 p.
2. Java SE Documentation at a Glance [електронний ресурс]. – Режим доступу до документації: http://www.oracle.com/technetwork/java/javase/documentation/index.html

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.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.

ВАРІАНТИ ЗАВДАННЬ

1. y=tg(x)
2. y=ctg(x)
3. y=sin(x)/cos(x)
4. y=cos(x)/sin(x)
5. y=2x/sin(x)
6. y=tg(x)/sin(2x)
7. y=ctg(x)/sin(7x-1)
8. y=sin(x)/sin(2x-4)
9. y=tg(x)/3x
10. y=tg(x)/ctg(x)
11. y=ctg(x)/tg(x)
12. y=sin(x)/tg(4x)
13. y=sin(x)/ctg(8x)
14. y=cos(x)/tg(2x)
15. y=cos(2x)/ctg(3x-1)
16. y=7x/tg(2x-4)
17. y=(x-4)/sin(3x-1)
18. y=tg(x)/(sin(4x) – 2cos(x))
19. y=ctg(x)/(sin(2x) + 4cos(x))
20. y=tg(x)ctg(2x)
21. y=sin(3x-5)/ctg(2x)
22. y=tg(4x)/x
23. y=ctg(8x)/x
24. y=sin(x-9)/(x-cos(2x))
25. y=1/sin(x)
26. y=1/cos(4x)
27. y=1/tg(2x)
28. y=1/ctg(2x)
29. y=sin(x)/(x+tg(x))
30. y=cos(x)/(x+2ctg(x))

ПОРЯДОК ВИКОНАННЯ

1. Запустити на виконання середовище Eclipse IDE.

2. Дослідити тестову програму, що наведена в методичних вказівках.

3. Почати створення власного проекту. Для цього слід викликати підпункут меню File->New->Project… У вікні, що відкриється слід вибрати Java Project (рис. 2) та натиснути кнопку ”Next>”.

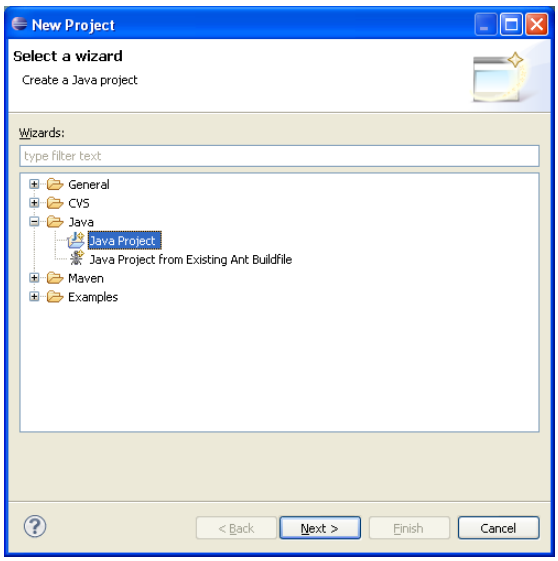

Рис. 2. Діалогове вікно вибору типу проекту в Eclipse IDE.

4. У вікні, що відкрилося, необхідно вказати назву нового проекту, місце розташування проекту, версію JRE на яке орієнтована програма, топологію проекту (Project layout) та натиснути кнопку ”Next>” (рис. 3).

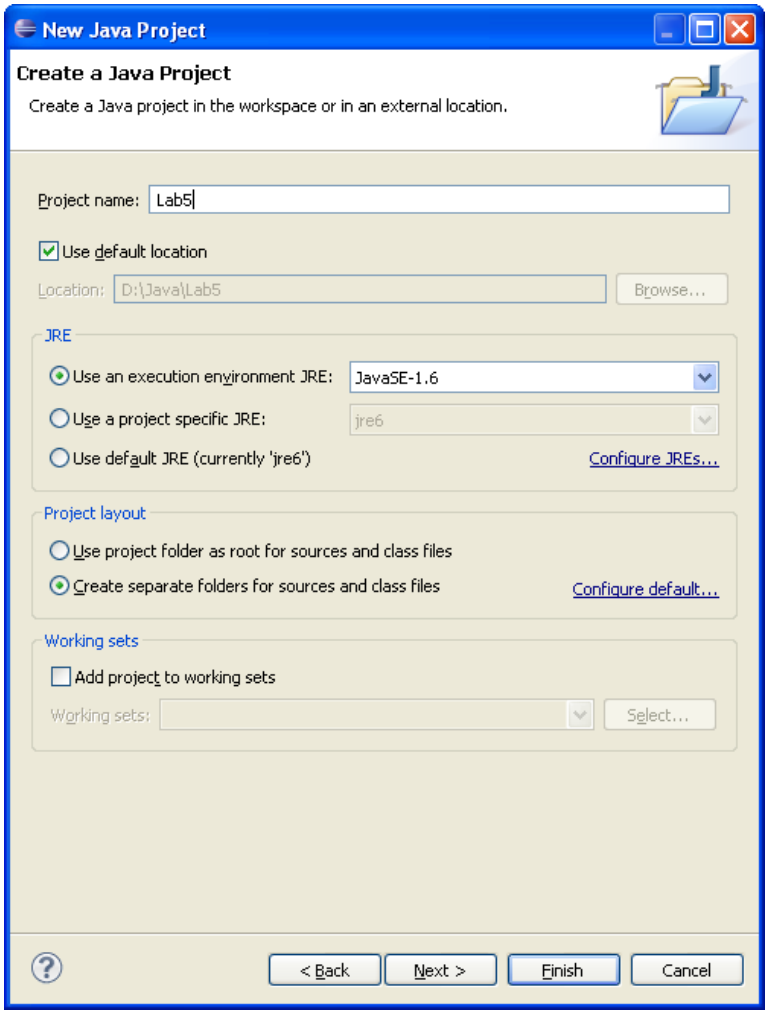

Рис. 3. Діалогове вікно створення нового проекту в Eclipse IDE.

5. У вікні, що відкрилося, натиснути кнопку ”Finish” (рис. 4).

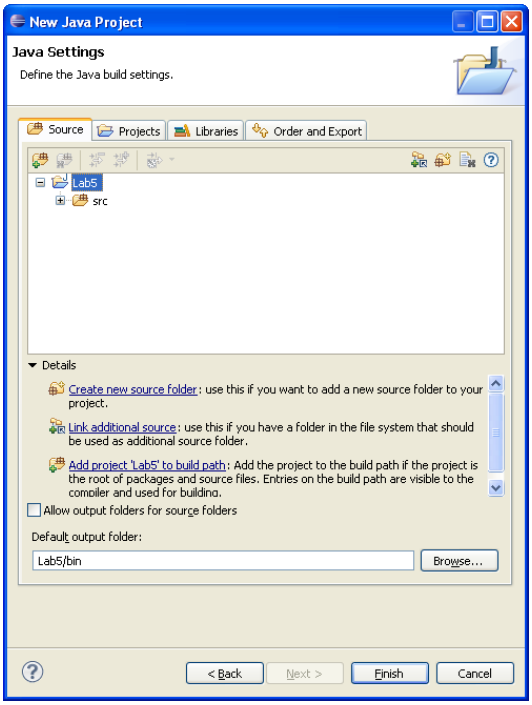

Рис. 4. Діалогове вікно налаштування властивостей побудови нового проекту в Eclipse IDE.

6. У вікні, що з’явилося (рис.5), викликати підпункт меню File->New->Class. У вікні, що відкрилося, слід задати кореневий каталог проекту, назву пакету, назву класу драйвера, що створюється, модифікатор доступу класу, базовий клас, методи, які слід згенерувати автоматично, вказати чи генерувати коментарі автоматично та натиснути кнопку ”Finish” (рис. 6). Аналогічним чином створити клас, що реалізує завдання.

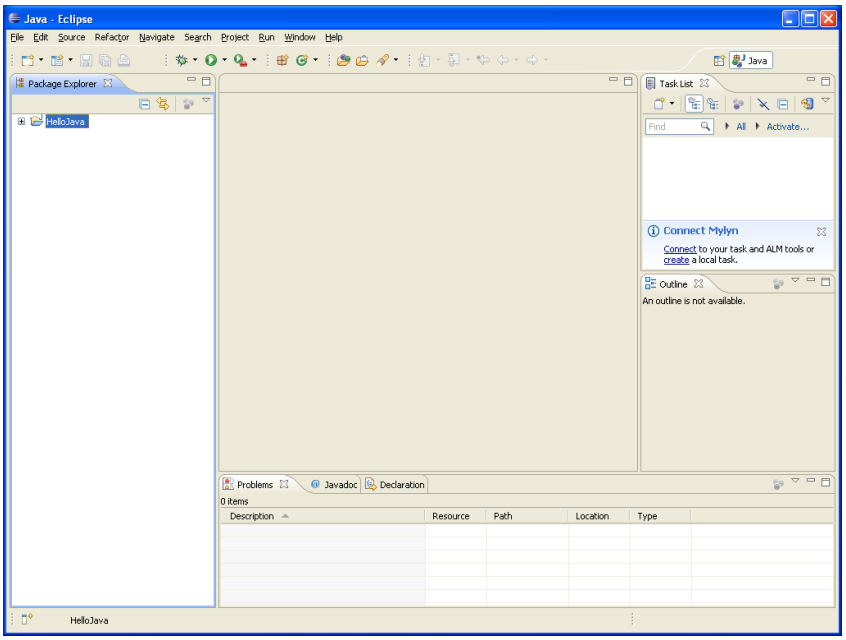

Рис. 5. Середовище Eclipse IDE з відкритим проектом Java.

6. У створеному файлі написати програму згідно завдання.

7. Запустити програму на виконання. Для цього слід вибрати підпункт меню Run-> Run.

8. Встановити точки переривання, запустити налагоджувач (Run->Debug) та покроково дослідити процес виконання програми (рис. 7).

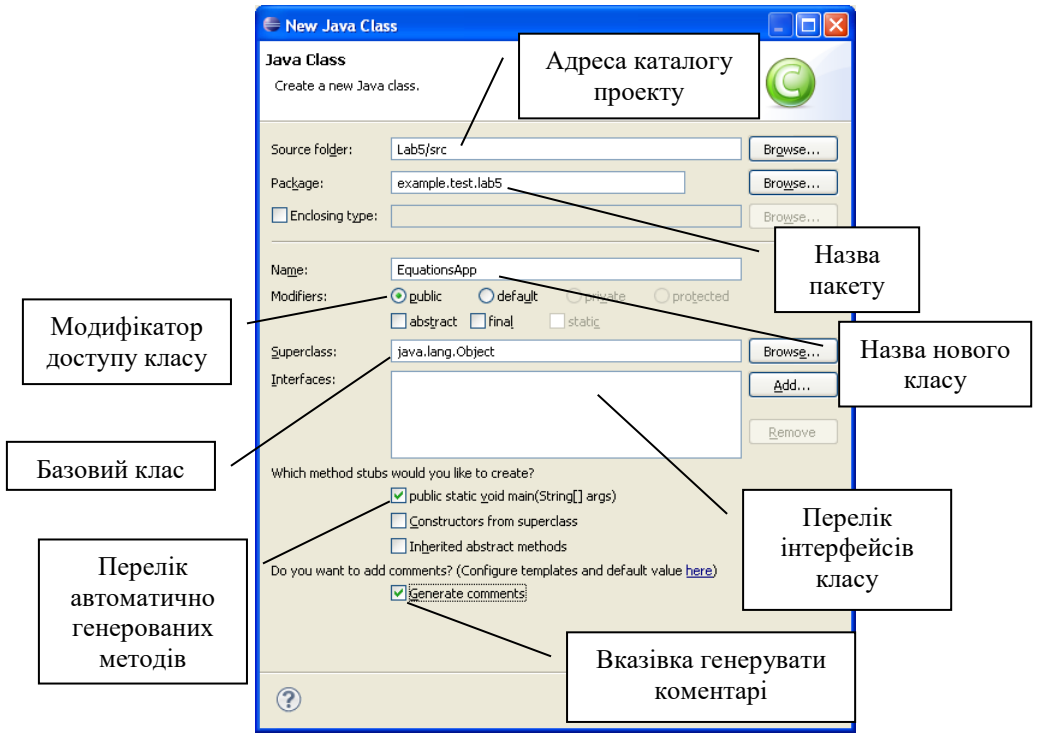

Рис. 6. Створення нового класу з використанням візуального конструктора класів.

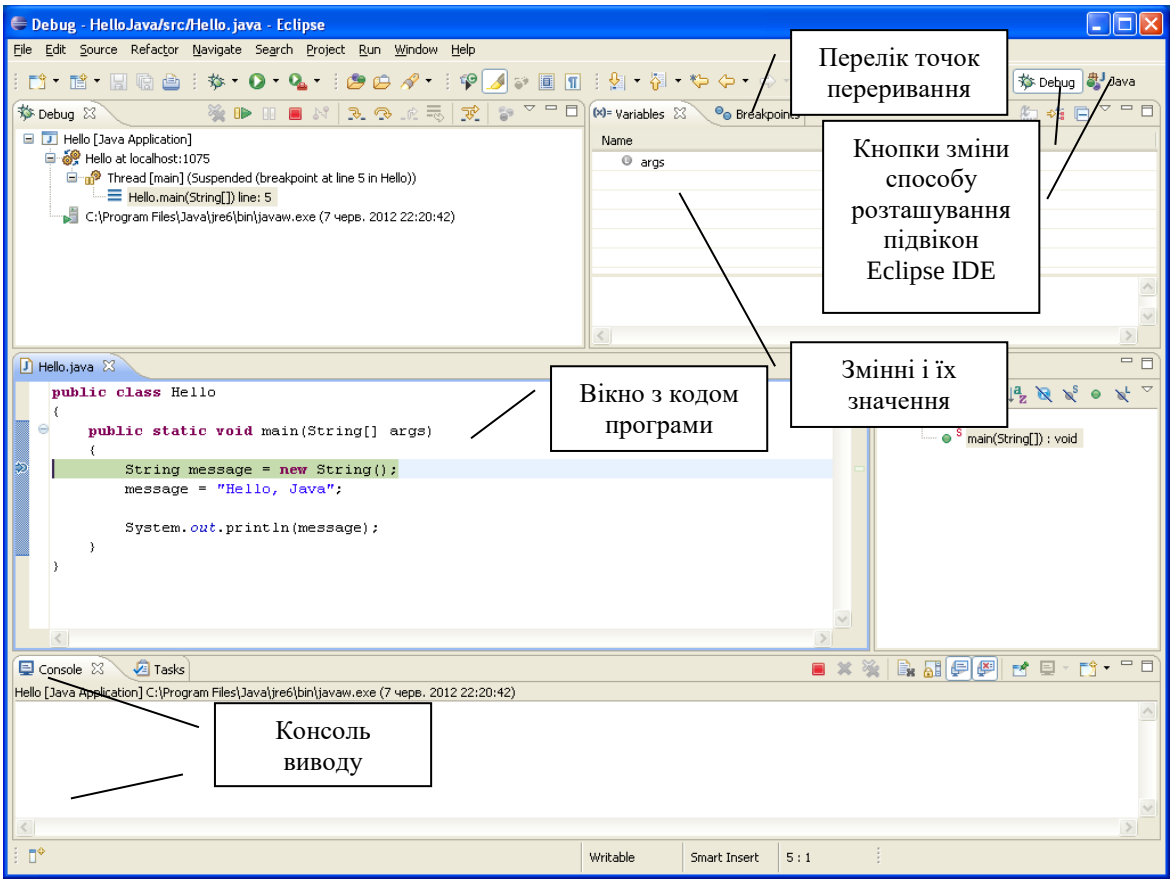

Рис. 7. Приклад вікна налагодження програми.

9. Перебуваючи у підкаталозі \src автоматично згенерувати документацію для всіх загальнодоступних класів у каталог \doc вашого проекту за допомогою команди:

```
javadoc -d ..\doc Група.Прізвище.Lab4
```

Проаналізувати автоматично згенеровану документацію.

ПРИКЛАД ПРОГРАМИ

Файл EquationsApp.java

```
/*******************************************************************************
 * Copyright (c) 2013-2023 Lviv Polytechnic National University. All Rights Reserved.
 *
 * This program and the accompanying materials are made available under the terms
 * of the Academic Free License v. 3.0 which accompanies this distribution, and is
 * available at https://opensource.org/license/afl-3-0-php/
 *
 * SPDX-License-Identifier: AFL-3.0
 *******************************************************************************/

package example.test.lab4;

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 {

	/**
	 * @param args
	 */
	public static void main(String[] args)
	{
		try
		{
			out.print("Enter file name: ");
			Scanner in = new Scanner(System.in);
			String fName = in.nextLine();
			PrintWriter fout = new PrintWriter(new File(fName));
			try
			{
				try
				{
					Equations eq = new Equations();
					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
				out.print(ex.getMessage());
			}
		}
		catch (FileNotFoundException ex)
		{
			// Блок перехоплює помилки роботи з файлом навіть якщо вони
			// виникли у блоці finally
			out.print("Exception reason: Perhaps wrong file path");
		}
	}
}
```

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

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


/**
 * Class <code>Equations</code> implements method for ((2 / tg(x)) / x) expression
 * calculation
 * @author EOM Stuff
 * @version 1.0
 */
class Equations
{
	/**
	 * Method calculates the ((2 / tg(x)) / x) expression
	 * @param <code>x</code> Angle in degrees
	 * @throws CalcException
	 */
	public double calculate(int x) throws CalcException
	{
		double y, rad;
		rad = x * Math.PI / 180.0;

		try
		{
			y = (2.0 / Math.tan(rad)) / x;

			// Якщо результат не є числом, то генеруємо виключення
			if (y==Double.NaN || y==Double.NEGATIVE_INFINITY ||
y==Double.POSITIVE_INFINITY || x==90 || x== -90)
				throw new ArithmeticException();
		}
		catch (ArithmeticException ex)
		{
			// створимо виключення вищого рівня з поясненням причини
			// виникнення помилки
			if (rad==Math.PI/2.0 || rad==-Math.PI/2.0)
				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;
	}
}
```

НАВЧАЛЬНЕ ВИДАННЯ

ВИКЛЮЧЕННЯ

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторної роботи № 4 з дисципліни “ Кросплатформні засоби програмування ”
для студентів базового напряму 6.123 - “Комп’ютерна інженерія”

Укладач Олексів М. В., к.т.н.

Редактор

Комп’ютерне верстання

Здано у видавництво . Підписано до друку
Формат 70x100/16. Папір офсетний. Друк на різографі
Умовн. друк. арк. Обл..-вид. арк..
Тираж прим. Зам..

Видавництво Національного університету “Львівська політехніка”
Реєстраційне свідоцтво ДК №751 від 27.12.2001 р.

Поліграфічний центр Видавництва
Національного університету “Львівська політехніка”

Вул. Ф. Колесси, 2. Львів, 79000