# КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА



## ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №4**

# з курсу

**«Об’єктно-орієнтоване програмування»**

*Студента 2 курсу*

*групи ПП-21 спеціальності 122 «Комп'ютерні науки» ОП «Прикладне програмування»*

%username%

*Викладач:*

к.ф.-м.н., доц. Шолохов О.В.

## Київ – 202

**1.Назва роботи**

Обробка виключних ситуацій

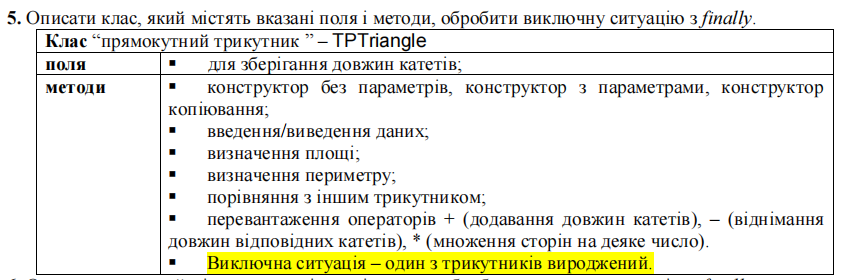
1. **Тема роботи**

Обробка виключних ситуацій

1. **Мета роботи**

Набуття навичок передбачення виключних ситуацій та їх обробки.

1. **Умова завдання**



1. **Рішення**

Об’являємо класс Трикутник. Оголошуємо змінні двох катетів. Описуємо метод, що дозволяє ввести довжини катетів та гіпотенузу, виводячи також запрошення. Описуємо метод, що перевіряє трикутник на виродженість. Описуємо метод, що обчислює площу трикутника за формулою двох катетів ((a \* b/2), де a і b - катети). Описуємо метод, що обчислює периметр. Описуємо метод, що порівнює два трикутники. Описуємо метод, що виводить усі дані про трикутник. Перевантажуємо унарні оператори “+” та “-”, бінарний оператор “\*” згідно з вищевказаною умовою завдання, після чого описуємо всі конструктори (стандартний за замовчуванням, конструктор з параметрами, конструктор копіювання (той, що копіює дані з іншого класу). У функції main створюємо екземпляр классу, скориставшись конструктором без параметрів, після чого викликаємо методи классу.

Код:

using System;

namespace Ra\_226

{

class Program

{

public class Triangle

{

double legOne;

double legTwo;

double hypotenuse;

public void basicInputSystem()

{

try

{

Console.WriteLine("Please enter the length of the first leg below:");

legOne = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Please enter the length of the second leg below:");

legTwo = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Please enter the length of the hypotenuse below:");

hypotenuse = Convert.ToDouble(Console.ReadLine());

if (legOne + legTwo < hypotenuse || hypotenuse + legOne < legTwo || hypotenuse + legTwo < legOne)

{

throw new Exception();

}

}

catch (Exception e)

{

Console.WriteLine("That triangle cannot exist in this reality plane and therefore will be replaced by an Egyptian one.");

}

finally

{

legOne = 3;

legTwo = 4;

hypotenuse = 5;

}

}

public double areaCalculator()

{

double area = (legOne \* legTwo) / 2;

return area;

}

public double perimeterCalculator()

{

double perimeter = Math.Sqrt((Math.Pow(legOne, 2) + Math.Pow(legTwo, 2))) + legOne + legTwo;

return perimeter;

}

public void comparator(Triangle heavensGate)

{

if (legOne == heavensGate.legOne && legTwo == heavensGate.legTwo)

{

Console.WriteLine("Triangles are equal.");

} else

{

Console.WriteLine("Triangles are not equal.");

}

}

public void basicOutputSystem()

{

Console.WriteLine("First leg is " + legOne + " miles long; second leg is " + legTwo + " miles long.");

Console.WriteLine("Area is " + areaCalculator() + " square miles; perimeter is " + perimeterCalculator() + " miles.");

}

public static double operator + (Triangle heavensGate)

{

return (heavensGate.legOne + heavensGate.legTwo);

}

public static double operator -(Triangle heavensGate)

{

if (heavensGate.legOne > heavensGate.legTwo)

{

return heavensGate.legOne - heavensGate.legTwo;

} else

{

return heavensGate.legTwo - heavensGate.legOne;

}

}

public static Triangle operator \*(Triangle heavensGate, double multiplier)

{

heavensGate.legOne \*= multiplier;

heavensGate.legTwo \*= multiplier;

return heavensGate;

}

public Triangle()

{

}

public Triangle(double legOneInput, double legTwoInput)

{

legOne = legOneInput;

legTwo = legTwoInput;

}

public Triangle(Triangle heavensGate)

{

legOne = heavensGate.legOne;

legTwo = heavensGate.legTwo;

}

}

static void Main(string[] args)

{

string logo = @" x

x x

x x

x x

x x x

x xxx x

x x x

x x

xxxxxxxxxxxxxxxxx";

Console.WriteLine(logo);

Triangle joyDivision = new Triangle();

joyDivision.basicInputSystem();

joyDivision.basicOutputSystem();

Triangle newOrder = new Triangle(joyDivision);

Console.WriteLine("Enter the multiplier below:");

double fiatMultipla = Convert.ToDouble(Console.ReadLine());

newOrder \*= fiatMultipla;

newOrder.basicOutputSystem();

joyDivision.comparator(newOrder);

Console.WriteLine("+ overload results: " + +newOrder);

Console.WriteLine("- overload results: " + -newOrder);

}

}

}

Результат роботи програми:

[screenshot]

[screenshot]

1. **Контрольні запитання**
2. Пояснити механізм генерування та опрацювання виключень.

У блоці try вібдувається спроба виконати деякі дії; якщо під час цієї спроби виникає виключення, то воно обробляється в наступному блоці catch, після чого відпрацьовує останній блок finally.

1. Що відбувається у випадку, коли не обробити виключну ситуацію?

Программа аварійно завершиться або поверне виключення за замовчуванням (в залежності від типу виключення)

1. Найуживаніші властивості класу *exception*.

message (повідомлення про помилку), StackTrace (місце виникнення помилки), InnerException (якщо поточний виняток згенерований, в свою чергу, іншим винятком, то ця властивість поверне посилання на перший з них), HelpLink (посилання на інформацію з причинами винятку), Source (збірка, в якій виник виняток).

1. **Висновки**

В результаті виконання даної лабораторної роботи я оволодів практичними навичками створення алгоритмів для виконання типових завдань на створення класу з заданими полями та методами, пошуку площі та периметра прямокутного трикутника та набув навичок створення класів з використанням перевантаження операторів та обробкою виключних ситуацій. Вважаю дану лабораторну роботу виконаною в повному обсязі.