Кам’янець-Подільський національний університет

 імені Івана Огієнка

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

Навчальна дисципліна «Об’єктно-орієнтоване програмування»

ЛАБОРАТОРНА РОБОТА #0107

Тема: Обробка винятків

Варіант №3

Виконав:

студент 1-го курсу

групи KNms1-B21

Волошин В.Ю.

Прийняла:

старший викладач,

Мястковська М.О.

Кам’янець-Подільський – 2023

1. **Хід роботи.**

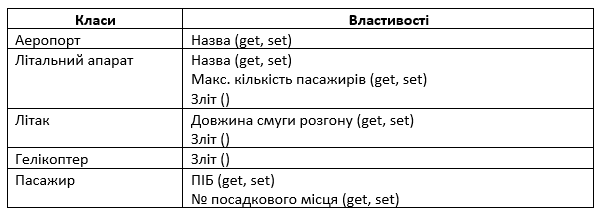
Обробка винятків в C# є важливою частиною програмування, що дозволяє ефективно управляти та вирішувати непередбачені ситуації в процесі виконання програми. Використання блоків try, catch, та finally спрощує обробку винятків, а оператор throw дозволяє програмістам вручну викидати винятки для вказаної ситуації.

Специфікація типів винятків у блоках catch надає можливість розділяти обробку різних видів помилок. Використання блоку finally дозволяє розмістити код, який завжди виконується, незалежно від того, чи сталася помилка чи ні.

Під час виникнення винятку створюється об'єкт винятку, який містить інформацію про помилку. Класи винятків у C# наслідуються від базового класу Exception, спрощуючи їх обробку.

Можливість вкладення блоків try дозволяє більш гнучко управляти винятками у різних частинах коду. Також розробники можуть створювати власні класи винятків, що відповідають специфіці їх програм та полегшують обробку власних виняткових ситуацій.

1. **Завдання до виконання.**



а) Обробити всі винятки з допомогою блоку try ... catch (Exception ...) в методі «зліт» класів Літак та Гелікоптер. При виникненні виключення виводити на екран повідомлення про помилку.

Описати власний клас винятків KilkistException. У методі «додати пасажира» класу Літальний апарат викидати власне виключення типу KilkistException при перевищенні максимальної кількості пасажирів (порівнювати поточну кількість пасажирів і значення властивості «макс. Кількість пасажирів»). У цьому ж методі обробити KilkistException в блоці catch. При виникненні виключення виводити повідомлення про помилку на екран.

В основній програмі (main) перевірити роботу блоку обробки винятків методів «зліт» і «додати пасажира».

б) Описати власний клас винятків SmugaRozgonuException. У конструкторі реалізувати перевірку значення довжини смуги розгону, при від'ємному значенні або нульовому значенні викидати власне виключення типу SmugaRozgonuException. Обробити виключення SmugaRozgonuExceptionс допомогою блоку try ... catch, в блоці обробки винятків вивести на екран повідомлення «Неможливо створити літак - вказана неправильна довжина смуги розгону: <довжина смуги розгону>» і повторно створити виняток.

В основній програмі (main) обробити виклик конструктора класу Літак і перевірити роботу обробника виключення.

1. **Лістинг програми.**

using System;

// Власний клас винятку KilkistException

public class KilkistException : Exception

{

    public KilkistException(string message) : base(message) { }

}

// Власний клас винятку SmugaRozgonuException

public class SmugaRozgonuException : Exception

{

    public SmugaRozgonuException(string message) : base(message) { }

}

// Клас Аеропорт

public class Airport

{

    public string Name { get; set; }

}

// Клас Літальний апарат

public class Aircraft

{

    public string Name { get; set; }

    public int MaxPassengers { get; set; }

    public void Zlit()

    {

        try

        {

            Console.WriteLine($"{Name} злітає...");

        }

        catch (Exception ex)

        {

            Console.WriteLine($"Помилка при зльоті: {ex.Message}");

        }

    }

}

// Клас Літак

public class Airplane : Aircraft

{

    private int \_smugaRozgonu;

    public int SmugaRozgonu

    {

        get { return \_smugaRozgonu; }

        set

        {

            try

            {

                if (value <= 0)

                    throw new SmugaRozgonuException($"Неможливо створити літак - вказана неправильна довжина смуги розгону: {value}");

                \_smugaRozgonu = value;

            }

            catch (SmugaRozgonuException ex)

            {

                Console.WriteLine(ex.Message);

                throw;

            }

        }

    }

    public void Zlit()

    {

        try

        {

            Console.WriteLine($"{Name} літає...");

        }

        catch (Exception ex)

        {

            Console.WriteLine($"Помилка при зльоті: {ex.Message}");

        }

    }

}

// Клас Гелікоптер

public class Helicopter : Aircraft

{

    public void Zlit()

    {

        try

        {

            Console.WriteLine($"{Name} піднімається в повітря...");

        }

        catch (Exception ex)

        {

            Console.WriteLine($"Помилка при зльоті: {ex.Message}");

        }

    }

}

// Клас Пасажир

public class Passenger

{

    public string PIB { get; set; }

    public int SeatNumber { get; set; }

}

// Клас Літальний апарат

public class FlyingMachine

{

    private int \_currentPassengers = 0;

    public int MaxPassengers { get; set; }

    public void AddPassenger()

    {

        try

        {

            if (\_currentPassengers >= MaxPassengers)

                throw new KilkistException($"Перевищено максимальну кількість пасажирів: {\_currentPassengers}");

            Console.WriteLine("Пасажир доданий до літального апарату.");

            \_currentPassengers++;

        }

        catch (KilkistException ex)

        {

            Console.WriteLine($"Помилка при додаванні пасажира: {ex.Message}");

        }

    }

}

class Program

{

    static void Main()

    {

        try

        {

            // Приклад виклику методу зліту для літака

            Airplane airplane = new Airplane();

            airplane.Name = "Boeing 737";

            airplane.SmugaRozgonu = -500; // Генерує виняток SmugaRozgonuException

            airplane.Zlit();

            // Приклад виклику методу додавання пасажира до літального апарату

            FlyingMachine flyingMachine = new FlyingMachine();

            flyingMachine.MaxPassengers = 2;

            flyingMachine.AddPassenger();

            flyingMachine.AddPassenger(); // Додаємо другого пасажира

        }

        catch (Exception ex)

        {

            Console.WriteLine($"Помилка в основній програмі: {ex.Message}");

        }

    }

}

1. **Результат роботи.**

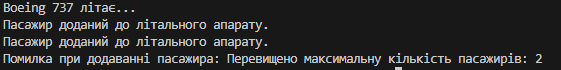
Без помилки



Неправильно вказана довжина смуги розгону



Перевищено максимальну кількість пасажирів



1. **Посилання на проект.**