Кам’янець-Подільський національний університет  
імені Івана Огієнка

КАФЕДРА КОМП’ЮТЕРНИХ НАУК

ЛАБОРАТОРНА РОБОТА #0107

з дисципліни «Об’єктно-орієнтоване програмування»

**Обробка винятків**

**Виконала**:  
студентка 2-го курса  
гр. KN1-B23  
Серебрянський Артем

**Прийняв**:  
старший викладач,  
Слободянюк О.В.

Дата здачі 09.05.2025р.

Кам’янець-Подільський – 2025

**Варіант: 9**Описати структуру зоопарка:

|  |  |
| --- | --- |
| **Класи** | **Властивості** |
| Зоопарк | Назва (get, set) |
| Вольєр / клітка | Номер (get, set)  Розмір (get, set)  Макс. кількість тварин (get, set)  Поточна кількість тварин (get, set) |
| Тварина | название (get, set)  boolean хищник (get, set)  Сформувати опис () |
| Риба | Глибоководна (get, set)  Сформувати опис () |
| Птах | Швидкість польоту (get, set)  Сформувати опис () |
| Звір | Середовище проживання (get, set)  Сформувати опис () |

а) Обробити всі винятки з допомогою блоку try ... catch (Exception ...) в методі «сформувати опис» класів Риба, Птах і Звір. При виникненні виключення виводити на екран повідомлення про помилку.

Описати власний клас винятків HyszhakException. У методі «додати тварина» в класі Клітка при спробі додати хижака в клітку, де вже містяться не хижі тварини викидати власне виключення типу HyszhakException. У цьому ж методі обробити HyszhakExceptionв блоці catch. При виникненні виключення виводити повідомлення про помилку на екран.

В основній програмі (main) перевірити роботу блоку обробки винятків методів «сформувати опис» класів Риба, Птах і Звір і методу «додати тварина» класу Клітка.

б) Описати власний клас винятків NazvaException. У конструкторі реалізувати перевірку значення властивості Назва, при значенні null або порожній рядку викидати власне виключення типу NazvaException. Обробити виняток NazvaException за допомогою блоку try ... catch, в блоці обробки винятків вивести на екран повідомлення «Неможливо створити тварина - не вказано назву» і повторно створити виняток.

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

**Покликання на Git Hub:**

**Код програми коли все заддано:**using System;

using System.Collections;

using System.Collections.Generic;

using System.Text; // Додано для використання Encoding

// Власні класи винятків

public class HyszhakException : Exception

{

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

}

public class NazvaException : Exception

{

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

}

// Основні класи зоопарку

public class Zoo

{

public string Name { get; set; }

public Zoo(string name)

{

Name = name;

}

}

public class Aviary

{

public int Number { get; set; }

public string Size { get; set; }

public int MaxAnimals { get; set; }

public int CurrentAnimals { get; set; }

private List<Animal> animals = new List<Animal>();

private bool containsPredator = false;

public void AddAnimal(Animal animal)

{

try

{

// Перевірка на хижака

if (animal.IsPredator && CurrentAnimals > 0 && !containsPredator)

{

throw new HyszhakException("Не можна додати хижака до вольєра з нехижими тваринами!");

}

if (!animal.IsPredator && containsPredator)

{

throw new HyszhakException("Не можна додати нехижу тварину до вольєра з хижаками!");

}

if (CurrentAnimals >= MaxAnimals)

{

throw new Exception("Вольєр переповнений!");

}

animals.Add(animal);

CurrentAnimals++;

if (animal.IsPredator) containsPredator = true;

Console.WriteLine($"Тварину {animal.Name} додано до вольєра №{Number}");

}

catch (HyszhakException ex)

{

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

}

catch (Exception ex)

{

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

}

}

}

public class Animal

{

private string name;

public string Name

{

get => name;

set

{

if (string.IsNullOrEmpty(value))

{

throw new NazvaException("Назва тварини не може бути порожньою");

}

name = value;

}

}

public bool IsPredator { get; set; }

public Animal(string name, bool isPredator)

{

try

{

Name = name;

IsPredator = isPredator;

}

catch (NazvaException ex)

{

Console.WriteLine("Неможливо створити тварину - не вказано назву");

throw; // Повторно генеруємо виняток

}

}

public virtual void FormDescription()

{

try

{

Console.WriteLine($"Тварина: {Name}, Хижак: {(IsPredator ? "так" : "ні")}");

}

catch (Exception ex)

{

Console.WriteLine($"Помилка при формуванні опису тварини: {ex.Message}");

}

}

}

public class Fish : Animal

{

public bool IsDeepWater { get; set; }

public Fish(string name, bool isPredator, bool isDeepWater) : base(name, isPredator)

{

IsDeepWater = isDeepWater;

}

public override void FormDescription()

{

try

{

base.FormDescription();

Console.WriteLine($"Тип: Риба, Глибоководна: {(IsDeepWater ? "так" : "ні")}");

}

catch (Exception ex)

{

Console.WriteLine($"Помилка при формуванні опису риби: {ex.Message}");

}

}

}

public class Bird : Animal

{

public int FlightSpeed { get; set; }

public Bird(string name, bool isPredator, int flightSpeed) : base(name, isPredator)

{

FlightSpeed = flightSpeed;

}

public override void FormDescription()

{

try

{

base.FormDescription();

Console.WriteLine($"Тип: Птах, Швидкість польоту: {FlightSpeed} км/год");

}

catch (Exception ex)

{

Console.WriteLine($"Помилка при формуванні опису птаха: {ex.Message}");

}

}

}

public class Beast : Animal

{

public string Habitat { get; set; }

public Beast(string name, bool isPredator, string habitat) : base(name, isPredator)

{

Habitat = habitat;

}

public override void FormDescription()

{

try

{

base.FormDescription();

Console.WriteLine($"Тип: Звір, Середовище проживання: {Habitat}");

}

catch (Exception ex)

{

Console.WriteLine($"Помилка при формуванні опису звіра: {ex.Message}");

}

}

}

class Program

{

static void Main(string[] args)

{

// Встановлення кодування для української мови

Console.OutputEncoding = Encoding.UTF8;

Console.InputEncoding = Encoding.UTF8;

Console.WriteLine("Тестування обробки винятків:");

// 1. Тестування методу FormDescription()

Console.WriteLine("\n1. Тестування методів FormDescription():");

try

{

Animal animal = new Animal("Лев", true);

animal.FormDescription();

Fish fish = new Fish("Щука", true, false);

fish.FormDescription();

Bird bird = new Bird("Орел", true, 80);

bird.FormDescription();

Beast beast = new Beast("Вовк", true, "Ліс");

beast.FormDescription();

}

catch (Exception ex)

{

Console.WriteLine($"Головний блок catch: {ex.Message}");

}

// 2. Тестування додавання тварин до вольєра

Console.WriteLine("\n2. Тестування додавання тварин до вольєра:");

try

{

Aviary aviary = new Aviary { Number = 1, Size = "Великий", MaxAnimals = 3 };

Animal deer = new Animal("Олень", false);

Animal tiger = new Animal("Тигр", true);

Animal rabbit = new Animal("Кролик", false);

aviary.AddAnimal(deer); // Додаємо нехижу тварину

aviary.AddAnimal(tiger); // Спроба додати хижака - має викликати HyszhakException

aviary.AddAnimal(rabbit); // Спроба додати нехижу тварину до хижака

}

catch (Exception ex)

{

Console.WriteLine($"Головний блок catch: {ex.Message}");

}

// 3. Тестування NazvaException

Console.WriteLine("\n3. Тестування NazvaException:");

try

{

Animal unnamedAnimal = new Animal("", false);

}

catch (NazvaException)

{

Console.WriteLine("Головний блок catch: NazvaException був оброблений");

}

catch (Exception ex)

{

Console.WriteLine($"Головний блок catch: {ex.Message}");

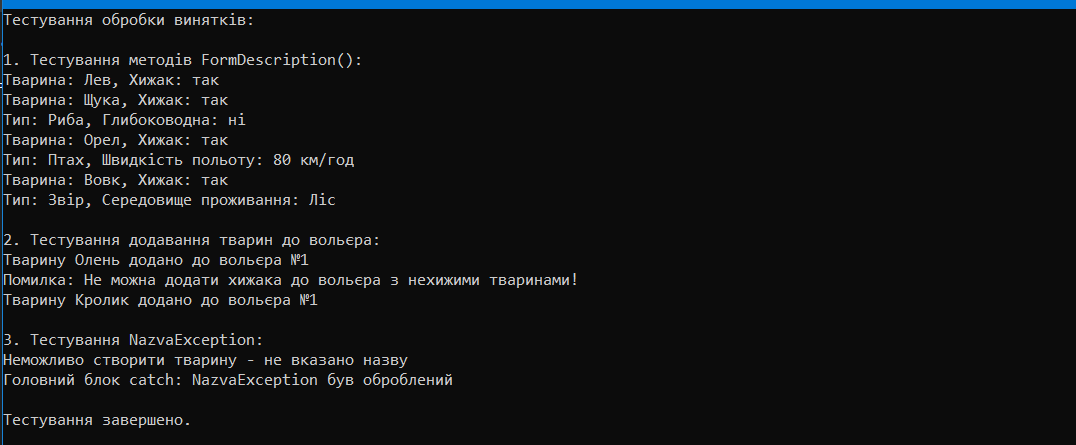
}

Console.WriteLine("\nТестування завершено.");

}

}

**Результат:**

****