### 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 №2

**Виконав**:  
студент 1-го курсу  
групи KNms1-B23  
Веретко С. І.

**Прийняла**:  
старший викладач,  
Мястковська М.О.

Кам’янець-Подільський – 2023

1. Теоретичні відомості

Виділяють три поширених терміни, призначеними для опису аномалій:

1. **Програмні помилки.** Так, зазвичай, називають помилки, що допускають програмісти.

List<int> lst1 = new List<int>();

lst1 = null;

lst1.Add(5);

// System.NullReferenceException: 'Object reference not set to an instance of an object.'

//lst1 was null.

2. **Користувацькі помилки**. Наприклад, введення замість числа тексту може призвести до помилки, якщо в коді не передбачена обробка таких ситуацій.

3. **Винятки**. Винятками зазвичай називаються аномалії, що виникають під час виконання, які важко, а часом і неможливо, врахувати під час програмування додатку. Прикладами винятків можуть бути спроба підключення до бази даних, яка більше не існує, відкриття пошкодженого файлу або спроба встановлення зв'язку з машиною, яка в поточний момент знаходиться в автономному режимі. У кожному з цих випадків програміст (або кінцевий користувач) мало що може зробити з подібними "винятковими" обставинами.

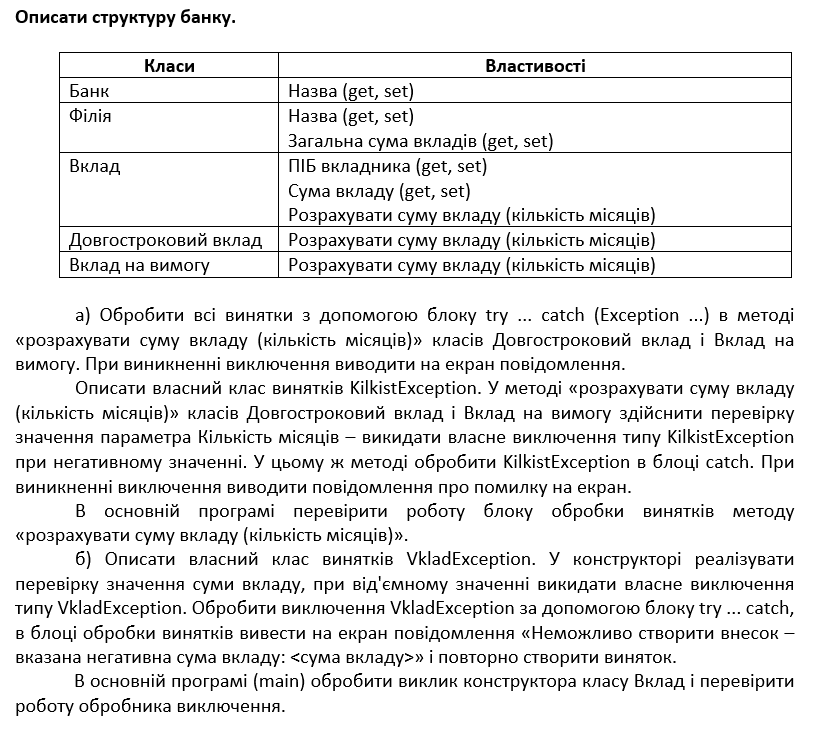
*Структурована обробка винятків в .NET* – це прийом, призначений для роботи з винятковими ситуаціями, які виникають під час виконання. У термінології .NET під "винятком" маються на увазі програмні помилки, некоректне введення користувачем і помилки часу виконання.

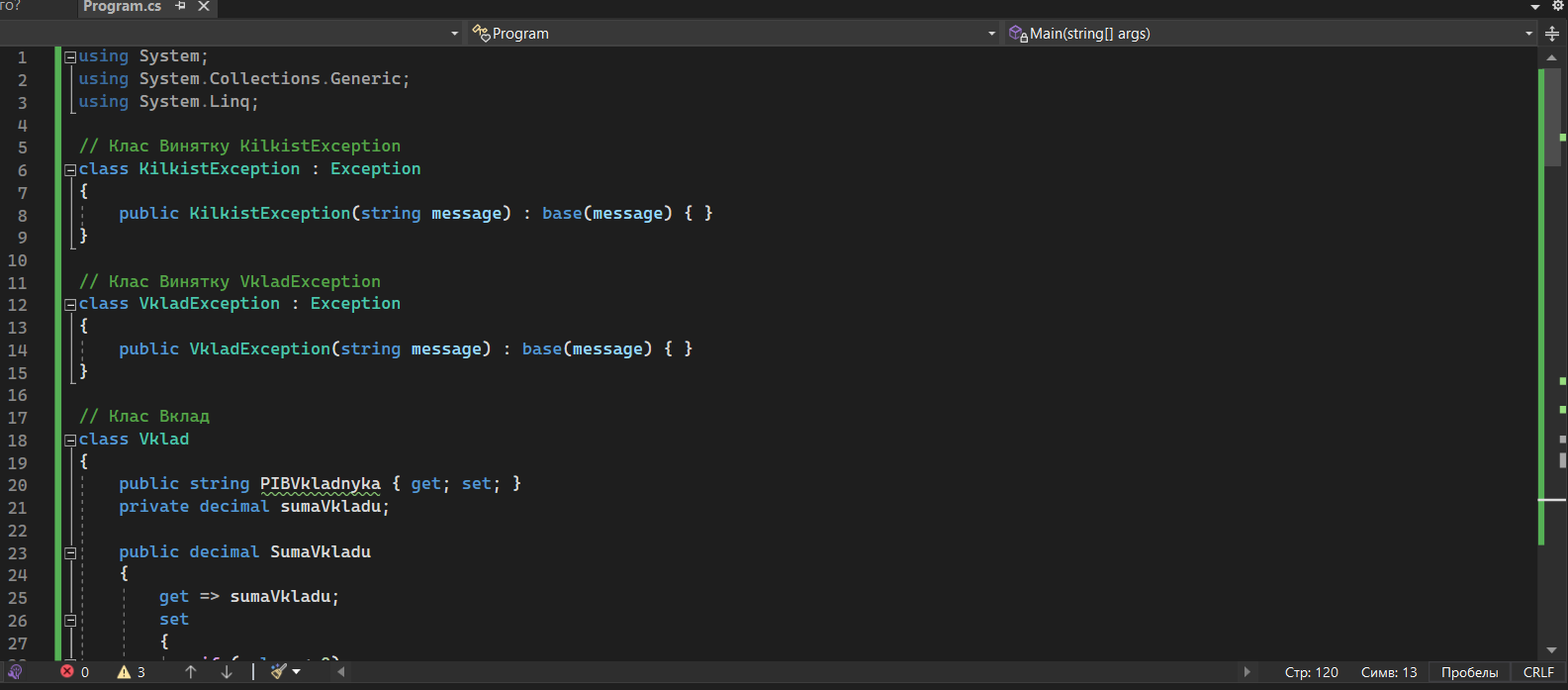
Оператор **try** вказує блок коду, що призначений для обробки помилкових ситуацій або очистки. За блоком **try** має слідувати блок **catch**, блок **finally** або обидва блоки одразу.

Блок catch виконується тоді, коли виникає помилка в блоці try. Блок finally виконується після блока try та після блока catch (якщо він присутній), забезпечуючи очистку не залежно від того, виникла помилка чи ні.

Блок catch має доступ до об’єкта Exception, який містить інформацію про помилку.

# 2. Завдання для виконання

  
3. Послідовність виконання завданнь



4. Лістинг програми

using System;

using System.Collections.Generic;

using System.Linq;

// Клас Винятку KilkistException

class KilkistException : Exception

{

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

}

// Клас Винятку VkladException

class VkladException : Exception

{

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

}

// Клас Вклад

class Vklad

{

public string PIBVkladnyka { get; set; }

private decimal sumaVkladu;

public decimal SumaVkladu

{

get => sumaVkladu;

set

{

if (value < 0)

{

throw new VkladException($"Неможливо створити внесок – вказана негативна сума вкладу: {value}");

}

sumaVkladu = value;

}

}

public virtual decimal RozrahuvatySumuVkladu(int kilkistMisjatsiv)

{

return SumaVkladu \* kilkistMisjatsiv;

}

}

// Довгостроковий вклад

class DovgostrokovyiVklad : Vklad

{

public override decimal RozrahuvatySumuVkladu(int kilkistMisjatsiv)

{

try

{

if (kilkistMisjatsiv < 0)

{

throw new KilkistException("Кількість місяців не може бути від'ємною.");

}

return base.RozrahuvatySumuVkladu(kilkistMisjatsiv);

}

catch (KilkistException ex)

{

Console.WriteLine($"Помилка при розрахунку суми довгострокового вкладу: {ex.Message}");

return 0;

}

}

}

// Вклад на вимогу

class VkladNaVymogu : Vklad

{

public override decimal RozrahuvatySumuVkladu(int kilkistMisjatsiv)

{

try

{

if (kilkistMisjatsiv < 0)

{

throw new KilkistException("Кількість місяців не може бути від'ємною.");

}

return base.RozrahuvatySumuVkladu(kilkistMisjatsiv);

}

catch (KilkistException ex)

{

Console.WriteLine($"Помилка при розрахунку суми вкладу на вимогу: {ex.Message}");

return 0;

}

}

}

// Клас Банк

class Bank

{

public string Name { get; set; }

public List<Branch> Branches { get; set; } = new List<Branch>();

}

// Клас Філія

class Branch

{

public string Name { get; set; }

public decimal TotalDeposits { get; set; }

public List<Vklad> Deposits { get; set; } = new List<Vklad>();

}

class Program

{

static void Main(string[] args)

{

try

{

// Приклад створення банку та додавання філій та вкладів

Bank myBank = new Bank { Name = "MyBank" };

Branch branch1 = new Branch { Name = "Branch 1" };

Branch branch2 = new Branch { Name = "Branch 2" };

myBank.Branches.Add(branch1);

myBank.Branches.Add(branch2);

Vklad deposit1 = new DovgostrokovyiVklad { PIBVkladnyka = "Client A", SumaVkladu = 1000 };

Vklad deposit2 = new VkladNaVymogu { PIBVkladnyka = "Client B", SumaVkladu = 1500 };

branch1.Deposits.Add(deposit1);

branch2.Deposits.Add(deposit2);

// Приклад розрахунку суми вкладу для конкретного вкладу

decimal calculatedAmount = deposit1.RozrahuvatySumuVkladu(12);

Console.WriteLine($"Розрахована сума для вкладу клієнта A: {calculatedAmount}");

}

catch (Exception ex)

{

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

}

}

}

5. Результат роботи програми



6. [xanax4rl/lab (github.com)](https://github.com/xanax4rl/lab)