

## ЛАБОРАТОРНА РОБОТА №2

**Тема: КОНСТРУКТОРИ.МОДИФИКАТОРЫ ДОСТУПУ В C# И UML. ДИАГРАМИ КЛАСІВ. ВІДНОШЕННЯ МІЖ КЛАСАМИ**

**Мета:** Здійснити об'єктно-орієнтоване проектування з урахуванням модифікаторів доступу в C# и UML і додавання конструкторів у діаграму класів та реалізація їх у програмному коді. Вивчити особливості використання різних видів конструкторів. Доповнити програму конструкторами різних видів.

### Постановка задачі:

1. Доповнити інтерфейси та реалізації класів методами-конструкторами класів (використати конструктори по умовчання, ініціалізації (введення значень з клавіатури, завдання їх за виразами та через виклик інших методів-членів класу).
2. Протестувати програму, демонструючи послідовність викликів конструкторів виведенням на екран повідомлень про створення об'єктів
3. Продемонструвати створення об'єктів за допомогою різних видів конструкторів:
  - 3.1. через ініціалізацію значень атрибутів класів списком аргументів, використовуючи конструктор ініціалізації;
  - 3.2. через присвоєння одних об'єктів іншим того самого типу, використовуючи конструктор копіювання;
  - 3.3. створення об'єкта за допомогою конструктора за замовчуванням.

### ХІД ВИКОНАННЯ РОБОТИ

1. У звітному HTML-документі розмістити інформацію по наступним пунктам:
  - 1 **Тема, мета та постановка задачі лабораторної роботи №2.**
- 2 Продовжити проектування структури ПЗ - у UML-діаграму класів (**classes**) додати конструктори, з урахуванням типів зв'язків між класами.
- 3 . Розмістити діаграму класів у звітний HTML-документ.

Доповнити інтерфейси та реалізації класів методами-конструкторами класів (використати конструктори по умовчання, ініціалізації (введення значень з клавіатури, завдання їх за виразами та через виклик інших методів-членів класу).

- ☐ Протестувати програму, демонструючи послідовність викликів конструкторів виведенням на екран повідомлень про створення об'єктів
- ☐ Продемонструвати створення об'єктів за допомогою різних видів конструкторів:

- 3.1. через ініціалізацію значень атрибутів класів списком аргументів, використовуючи конструктор ініціалізації;
- 3.2. через присвоєння одних об'єктів іншим того самого типу, використовуючи конструктор копіювання;
- 3.3. через перетворення об'єктів або змінних різних типів в об'єкт даного класу, використовуючи конструктор перетворення;
- 3.4. створення пустого об'єкта за допомогою конструктора за замовчуванням.

#### **Примітка:**

При написанні програмного коду, пам'ятати, що при таких типах зв'язків між класами, як агрегація та композиція, конструктор, наприклад, класу-композиту (зв'язок - композиція) повинен визвати конструктор підлеглого класу. Теж саме буде відноситись і до класу-агреганту (зв'язок агрегація). Таким чином система розгортається самостійно.

## **II Проектування ПЗ**

### **1. Об'єктно-орієнтоване проектування (OOD) для усіх версій проекту**

2.1 Проектування структури ПЗ - UML-діаграма класів (**classes**), з обґрунтуванням типів зв'язків між класами, атрибутами та методами (*для усіх версій*).

2.2 Таблиця класів, яка буде містити назву класу, атрибути, методи та методи-конструктори .

Назва класу	Атрибути	Призначення атрибутів
	.....	
	Методи	Опис методів
	.....	

## **III Кодування (програмування)**

- 3.1 Файлова структура проекту (скрін шот Solution explorer).
- 3.2 Програмні коди асів з конструкторами
- 3.3 Програмний код класу Program. Виведення результатів виконання

## **IV Впровадження та супроводження ПЗ**

- 4.1. Посилання на папку з проектом, усіма файлами проекту, виконувальним файлом.exe, з можливістю відкрити проект зі звіту.
- 4.2 Запуск exe-файла та повідомлення про причину неможливості запустити exe-файл (через неспівпадіння версій MS Framework, Visual Studio тощо). Можливий запуск програми з генерованої документації на код

**ЗВІТИ З ЛАБОРАТОРНИХ РОБІТ**  
**З ДИСЦИПЛІНИ «Технології .NET розробки веб-додатків»**

Орієнтовний макет

Студент групи Іванов Іван Іванович

Лабораторна робота №1	Лабораторна робота №2	Лабораторна робота №3	Лабораторна робота №4	Лабораторна робота №5	Лабораторна робота	Лабораторна робота №7	Лабораторна робота №8	Лабораторна робота №9
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------	-----------------------	--------------------------	--------------------------	--------------------------

  

Тема, мета та постановка задачі ЛР№2

Проектування структури ПЗ

Таблиця класів (клас+атрибути+методи)

UML- діаграма класів

Кодування ПЗ (програмування)

Файлова структура проекту

Програмні коди класів з конструкторами

Програмний код класу Program  
Результат виконання програми

Впровадження та супроводження ПЗ

Папка з проектом

Запуск .exe файлу

В И С Н О В К И

Місце виведення інформації

**ПРИМІТКА:**

Звіт з лабораторних робіт слід підготувати у вигляді **гіпертекстового документа у форматі html**. Документ має містити меню, яке включає команди, що подані нижче. Слід реалізувати запуск програм на виконання з гіпертекстового документа. Звіт та проекти лабораторних робіт слід записати на Гугл-Диск. На диску має бути файл readme.txt, який містить відомості про автора звіту та проектів.

**Титульна html- сторінка.**

Назва роботи  
Автор (ПІБ, група, курс, № заліковки)  
Фото  
Рік навчання

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

- [Конструктори](#)
- [Виклик конструкторів](#)

## **Конструктори**

[\[up\]](#)

**Конструктор** – спеціальний метод, що автоматично викликається при створенні об'єкта.

- Призначення конструктора — створення екземплярів класу, тобто об'єктів.
- Клас може мати декілька конструкторів.
- Кількість конструкторів визначається кількістю способів створення об'єкта.

**Дії, що виконує конструктор:**

- виділення динамічної пам'яті для об'єкта;
- ініціалізація елементів даних класу;
- копіювання об'єктів, тобто елементів даних-членів класу;

**Правила визначення конструкторів:**

1. Конструктор має те саме ім'я, що і тип класу.
2. Конструктор ніколи не повертає значення, тому не можна вказувати тип значення, що повертається, навіть void.
3. Конструктор не має оператора return.
4. Конструктор не успадковується.
5. Не може бути оголошений як static, virtual, const.
6. Помилки з конструктора повертається через механізм обробки виключних ситуацій.
7. Визначення конструктора може знаходитися всередині класу або за його межами.
8. У випадку визначення конструктора за межами опису класу застосовується оператор :: розширення області видимості.
9. Конструктор може мати параметри і може бути без параметрів. Конструктор з параметрами ініціалізує об'єкт в точці його оголошення.
10. Конструктор без параметрів ініціалізує пустий об'єкт. Як правило використовується для виділення блока пам'яті і ініціалізації покажчика на можливі значення.

**Види конструкторів:**

1. [Конструктор за замовчуванням.](#)
2. [Конструктор ініціалізації.](#)
3. [Конструктор копії](#) для створення та ініціалізації нового об'єкта, використовуючи значення елементів даних існуючого об'єкта.

**Конструктор за замовчуванням:**

**Конструктор за замовчуванням** - спеціальний конструктор, який не має параметрів.

**Призначення конструктора за замовчуванням** – виділення пам'яті, проте присвоєння значень змінним класу не відбувається.

**Особливості використання конструктора:**

1. Якщо конструктор для класу не визначений, то компілятор генерує конструктор за замовчуванням.
2. Конструктори, генеровані компілятором, не присвоюють початкові значення змінним-членам класу. Тому потрібно визначати власний конструктор.
3. Якщо клас має конструктор без параметрів (явно визначений або генерований компілятором), можна визначити об'єкт класу без передачі аргументів конструктору.
4. Якщо конструктору не передаються аргументи, в означенні об'єкта не потрібно включати пусті круглі дужки.

5. Включати пусті круглі дужки потрібно під час оголошення функції, що повертає тип класу.
6. Якщо допустити помилку в оголошенні об'єкту, то компілятор не згенерує повідомлення про помилку доти, доки не буде спроби використати її як екземпляр класу.

### Конструктор ініціалізації:

**Конструктор ініціалізації** - спеціальний вид конструктора, який разом із створенням об'єкту присвоює його полям початкові значення, отримуючи їх як параметри.

### Особливості використання конструктора:

1. Ініціалізація змінних усередині інтерфейсу класу неможлива, тому що означення класу задає тільки тип кожної змінної-члена класу, але не резервує для них реальної області пам'яті.
2. Змінні-члени класу потрібно ініціалізувати кожний раз, як тільки створюється екземпляр класу.
3. Для ініціалізації змінних-членів в конструкторі класу найчастіше використовується операція присвоєння змінним-членам указаних значень.
4. Якщо певним типам даних, наприклад, константам і посиланням, не можна присвоїти значення, тоді застосовується список ініціалізації, який дозволяє ініціалізувати змінні без використання операцій присвоєння значень.

### Список ініціалізації:

- Список ініціалізації в означенні конструктора розміщується безпосередньо після списку параметрів.
- Ініціалізатор поля містить ім'я змінної з початковим значенням в круглих дужках, що записують поспіль.
- Якщо всі змінні-члени ініціалізуються через список ініціалізації, то тіло конструктора не містить операторів.
- Взагалі конструктор може мати як список ініціалізації, так і оператори в тілі конструктора.

[\[up\]](#)

### Конструктор копії:

**Призначення** - конструктор копіювання використовується для отримання копії змінних екземпляра класу під час оголошення об'єкта.

**Формат оголошення:** конструктор копії класу – це конструктор з *єдиним параметром*, тип котрого визначено як *посилання* на тип поточного класу.

### Випадки виклику конструктора копії:

1. Конструктор викликається під час створення тимчасових значень в результаті передачі об'єктів класу як параметрів-значень.
2. Компілятор автоматично викликає конструктор копіювання класу при поверненні функцією об'єкта класу.
3. Оскільки компілятор викликає конструктор копіювання кожний раз під час передачі об'єкта класу в функцію, об'єкт класу не можна передавати як перший параметр в сам конструктор копіювання. Потрібно передати посилання на нього.
4. Передача об'єкта у функцію є причиною *нескінченної рекурсії*.
5. Якщо тип значення, що повертається функцією, є об'єктом класу, то під час виклику функції компілятор генерує тимчасовий об'єкт цього класу і використовує значення, яке визначено в операторе return, для ініціалізації цього об'єкта.
6. Для виконання ініціалізації, компілятор також викликає конструктор копіювання.

### Особливості конструктора копії:

1. Якщо конструктор копіювання класу не визначено, то компілятор генерує його неявно.
2. Конструктор, що генерується компілятором, ініціалізує новий об'єкт, виконуючи операцію по елементного копіювання змінних існуючого об'єкта класу, який передається як аргумент.
3. Використовуючи об'єкт того самого типу, завжди можна ініціалізувати його, навіть якщо конструктор копіювання в класі не визначено.
4. Конструктори з єдиним параметром - посиланням на об'єкт дозволяють ініціалізувати об'єкт, використовуючи *знак рівності* в самому означенні об'єкта.
5. Використання знака рівності – альтернативний спосіб передачі єдиного значення конструктору. Це є операція ініціалізації.