

### Лабораторна робота 3.

## РОЗРОБКА ДОДАТКІВ ІЗ ВИКОРИСТАННЯМ БАЗОВИХ ЕЛЕМЕНТІВ ОБ'ЄКТНО – ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

**Мета роботи:** Придбання практичних навичок використання базових елементів об'єктно-орієнтованого програмування (ООП) під час розробки консольних додатків мовою C#.

### Короткі теоретичні відомості ООП

основане на трьох основних принципах:

1. Інкапсуляція – об'єднання в єдине ціле (клас) даних і алгоритмів обробки цих даних. В ООП дані називаються полями, а алгоритми – методами.
2. Спадкування – можливість створення ієрархії класів, коли класинащадки отримують від класа-попередника поля й методи.
3. Поліморфізм (від гр. *poly* - багато і *morphos* – форма, означає багато форм) – це властивість класів однієї ієрархії вирішувати схожі за змістом завдання за допомогою різних алгоритмів.

#### *Класи*

Класи – це основний спосіб організації даних у C#. Будь-яка програма, що написана цією мовою, повинна складатися не менш чим із одного класу (отже C# є "справжньою" об'єктно-орієнтованою мовою, на відміну, скажімо, від C++, у якому використання класів можливе, але необов'язково). *Функції в мові C#*

Будь-яка функція повинна бути членом класу (методом). Методи можуть приймати або не приймати параметри, повертати або не повертати значення. Параметри можуть передаватися в методи за значенням або за посиланням. Методи можуть бути статичними або методами екземплярів класу (об'єктів).

## *Модифікатори*

Одним з важливих принципів об'єктно-орієнтованого підходу до програмування є інкапсуляція даних: вважається, що внутрішній устрій класу й конкретна реалізація його методів повинні бути невідомі зовнішнім споживачам. Для виразу різних рівнів доступності елементів програми в C# існує обширний набір модифікаторів. Наприклад, поля та методи можуть мати такі основні модифікатори доступу:

`public` (даний елемент класу доступний усім зовнішнім споживачам);  
`private` (елемент недоступний за межами опису класу; цей модифікатор ставиться за умовчанням); `internal` (елемент доступний тільки для класів, визначених у тій же збірці, що і даний клас).

Крім того, існують модифікатори, що змінюють поведінку елементів класу, наприклад:

`const` (свідчить, що дана змінна не може бути змінена);  
`readonly` (описує змінні, які можуть набути значення тільки при самому описі або в конструкторі класу); `static` (вказує, що даний елемент належить типу об'єкта, а не конкретному екземпляру).

Приклад програми, у якій є клас `Hero`, наведений нижче:

```
class Hero
{
    private int Age;    private
    string Name;    private
    readonly bool Alive;

    // Конструктор
    public Hero(int Age, string Name)
    {
        this.Age = Age; this.Name = Name;
        Alive = true;
    }
}
```

```

    public int GetAge() { return Age; }
    public string GetName() { return Name; }
    public bool IsAlive() { return Alive; }
    public static void Hobby() { /*...*/ }    public
    void Talk() { /*...*/ }
        /*...*/
    }

    // Створення об'єкта класу Hero
    Hero LittleHero = new Hero(19, "Student");
    Console.WriteLine(LittleHero.GetName() + ", " +
        LittleHero.GetAge() + ", " + LittleHero.IsAlive());

```

*Ключове слово this* є посиланням на поточний екземпляр об'єкта; посилання *this* є схованим покажчиком, що передається в кожний нестатичний метод класу; будь-який метод може використовувати *this* для доступу до інших нестатичних методів і змінних цього об'єкта.

### *Конструктори класів*

Конструктор – метод, який ініціалізує стан об'єкта після його створення. Конструктор має те ж ім'я, що й клас, у якому він використовується; конструктор не повертає значення (навіть типу `void`), конструктор викликається автоматично.

Якщо конструктор не заданий у програмі, то він буде автоматично згенерований компілятором для побудови відповідних об'єктів, конструктор можна перевантажувати.

### *Статичні методи класу та конструктори*

Статичний конструктор, як і всі статичні методи класу, пов'язаний з класом, а не з об'єктом і необхідний для ініціалізації статичних даних. У статичного конструктора існує цілий ряд особливостей:

- у класі може бути лише один статичний конструктор;
- при оголошенні статичного конструктора не можна передавати параметри

та не можна вказувати модифікатори доступу; статичний конструктор виконується лише один раз, незалежно від кількості створених об'єктів даного класу; статичний конструктор буде викликаний до будь-якого першого звернення до цього класу (зазвичай перед першим викликом будь-якого члена класу); статичний конструктор ніколи не викликається ніяким іншим кодом, а лише середовищем, що виконує .NET під час завантаження класу; статичний конструктор має доступ лише до статичних членів класу.

Для прикладу зі статичним конструктором створимо клас, що описує банківські філії, статичне поле міститиме бонус у відсотках для оформлення депозитів, поточний баланс у кожної філії буде свій.

```
class Bank
{
    private double _currBalance;
    private static double _bonus;    public
    Bank(double balance)
    {
        _currBalance = balance;
    }    static
    Bank()
    {
        _bonus = 1.04;
    }
    public static void SetBonus(double newRate)
    {
        _bonus = newRate;
    }
    public static double GetBonus()
    {
        return
        _bonus;
    }
}
```

```

    public void SetBalance(double newBalance)
    {
        _currBalance = newBalance;
    }
    public double GetBalance()
    {
        return _currBalance;
    }
    public double GetPercents()
    {
        double percent = _currBalance * _bonus;
return percent;

        }
    }

```

```

Bank b1 = new Bank(10000);
WriteLine("Поточний баланс:" + b1.GetBalance());
WriteLine("Поточний бонусний відсоток:" + Bank.GetBonus());
WriteLine("Сума з урахуванням відсотків:" + b1.GetPercents());

```

### *Масиви об'єктів у C#*

Класи є користувацькими типами даних, але за своїми властивостями не відрізняються від стандартних класів C# і можуть утворювати масиви. Далі наведений фрагмент програми, в якому створюється масив для зберігання двох об'єктів класу MyClass та два об'єкта цього класу додаються до масиву:

```

MyClass[] ArrayOfObject = new MyClass[2]; for (int i = 0; i < ArrayOfObject.Length; i++)
    ArrayOfObject[i] = new MyClass();

```

```

class MyClass
{
    /*....*/
}

```

## Завдання до лабораторної роботи Завдання

### 1.

Загальні вимоги до розроблювальних програм: наявність найпростішого текстового інтерфейсу користувача; розробка програми для обробки відомості (див. варіанти завдань).

Для цього необхідно описати клас, поля якого відповідають вихідним полям відомості. Для установки значень полів повинен використовуватися конструктор. Обчислення значень розрахункових полів відомості, одержання значень вихідних полів повинне виконуватися за допомогою відповідних нестатичних методів класу.

Програма повинна забезпечувати створення масиву об'єктів цього класу, введення вихідних даних з консолі й вивід на консоль вихідних значень і полів, що розраховуються, кожної із записів відомості, а також підсумкової інформації з відомості.

Варіант 1 Відомість нарахування зарплати співробітникам підприємства:

№ з/п	Прізвище	Зарплата, грн	Утримано, грн	Видано, грн
1	F	Z	P	$S = Z - P$
2				
...				
	Разом	$\Sigma$	$\Sigma$	$\Sigma$

Варіант 2 Відомість витрат палива на автобазах міста:

№ з/п	Автобаза	Витрачено палива, кг	Кількість автомашин, шт.	Середня витрата палива, кг
1	A	T	K	$C = T/K$
2				
...				
n				
	Разом	$\Sigma$	$\Sigma$	$/n \Sigma$

Варіант 3. Відомість використання часу в науковому центрі:

№ з/п	Кафедра	Використання часу на експериментальній установці, годин		Відхилення від плану	
		за планом	фактично	у годинах	у %

1	K	P	F	$O_1 = P - F$	$O_2 = O_1 \times 100/P$
2					
...					
	Разом	$\Sigma$	$\Sigma$		

**Варіант 4. Відомість споживання електроенергії на заводах міста:**

№ з/п	Завод	Споживання електроенергії, кВт/год.		Відхилення від плану	
		за планом	фактично	у кВт/год.	у %
1	Z	P	F	$O_1 = P - F$	$O_2 = O_1 \times 100/P$
2					
...					
	Разом	$\Sigma$	$\Sigma$		

**Варіант 5 Відомість руху матеріалів на складі підприємства за звітний період:**

	Склад	Рух матеріалів за період, грн			
№ з/п		залишок на початок періоду	отримано	видано	Залишок на кінець періоду
1	C	Oc	P	V	$R = Oc + P - V$
2					
...					
	Разом	$\Sigma$	$\Sigma$	$\Sigma$	$\Sigma$

**Варіант 6. Відомість прибутку підприємства за звітний період за видами продукції:**

№ з/п	Продукція	Кількість, шт.	Оптова ціна, грн	Собівартість, грн	Прибуток, грн
1	Pr	K	Z	C	$P = K(Z - C)$
2					
...					
	Разом	$\Sigma$			$\Sigma$

**Варіант 7. Відомість відвідування занять студентами:**

№ з/п	Прізвище	Пропущено, год.		Пропуск	
		усього	виправдано	у годинах	у %
1	F	V	O	$P_1 = V - O$	$P_2 = P_1 \times 100/V$
2					
...					

	Разом:	$\Sigma$	$\Sigma$	$\Sigma$	
--	--------	----------	----------	----------	--

Варіант 8.

Відомість обсягу поставок продукції в натуральному та вартісному виразах:

№ з/п	Продукція	Шифр	Обсяг постачань, шт.	Оптова ціна, грн	Обсяг постачань, грн
1	P	H	V	Z	$O = V \times Z$
2					
...					
	Разом		$\Sigma$	$\Sigma$	$\Sigma$

Варіант 9. Відомість розрахунку середньої вартості перевезення авіапасажирів:

№ з/п	Тип літака	Рейс	Витрати на рейс, грн.	Кількість пасажирів	Середня вартість перевезення, грн.
1	T	R	Z	K	$S = Z/K$
2					
...					
n					
	Разом:		$\Sigma$	$\Sigma$	/n $\Sigma$

Варіант 10. Відомість обліку часу роботи верстатів підприємства:

№ з/п	Тип верстата	Час роботи, год.		Відхилення від плану	
		за планом	фактично	у годинах	у %
1	Z	P	F	$O1 = P - F$	$O2 = O1 \times 100/P$
2					
...					
	Разом	$\Sigma$	$\Sigma$		

Варіант 11. Відомість випуску деталей робітниками цеху:

№ з/п	Прізвище	Кількість деталей, шт.		Брак	
		виготовлено	прийнято	шт.	%
1	Z	P	F	$O1 = P - F$	$O2 = O1 \times 100/P$
2					
...					
	Разом	$\Sigma$	$\Sigma$	$\Sigma$	

Варіант 12. Відомість наявності та руху основних фондів підприємства:



№ з/п	Назва фонду	Наявність на початок року, шт.	Поступило, шт.	Вибито, шт.	Наявність на кінець року, шт.
1	F	N1	P	V	$N2 = N1 + P - V$
2					
...					
	Разом	$\Sigma$	$\Sigma$	$\Sigma$	$\Sigma$

**Варіант 13. Відомість витрат палива на автобазах міста:**

№ з/п	Автобаза	Витрачено палива, кг	Кількість автомашин, шт.	Середня витрата палива, кг
1	A	T	K	$C = T/K$
2				
...				
n				
	Разом	$\Sigma$	$\Sigma$	$\Sigma / n$

**Варіант 14. Відомість використання часу в науковому центрі:**

№ з/п	Кафедра	Використання часу на експериментальній установці, годин		Відхилення від плану	
		за планом	фактично	у годинах	у %
1	K	P	F	$O1 = P - F$	$O2 = O1 \times 100/P$
2					
...					
	Разом	$\Sigma$	$\Sigma$		

**Варіант 15. Відомість споживання електроенергії на заводах міста:**

№ з/п	Завод	Споживання електроенергії, кВт/год.		Відхилення від плану	
		за планом	фактично	у кВт/год.	у %
1	Z	P	F	$O1 = P - F$	$O2 = O1 \times 100/P$
2					
...					
	Разом	$\Sigma$	$\Sigma$		

**Варіант 16. Відомість прибутку підприємства за звітний період за видами продукції:**

№ з/п	Продукція	Кількість, шт.	Оптова ціна, грн	Собівартість, грн	Прибуток, грн
1	Pr	K	Z	C	$P = K(Z - C)$

2					
...					
	Разом	$\Sigma$			$\Sigma$

Варіант 17. Відомість відвідування занять студентами:

№ з/п	Прізвище	Пропущено, год.		Пропуск	
		усього	виправдано	у годинах	у %
1	F	V	O	$P1 = V - O$	$P2 = P1 \cdot 100/V$
2					
...					
	Разом:	$\Sigma$	$\Sigma$	$\Sigma$	

Варіант 18.

Відомість обсягу поставок продукції в натуральному та вартісному виразах:

№ з/п	Продукція	Шифр	Обсяг поставок, шт.	Оптова ціна, грн	Обсяг поставок, грн
1	P	H	V	Z	$O = V \times Z$
2					
...					
	Разом		$\Sigma$	$\Sigma$	$\Sigma$

Варіант 19. Відомість розрахунку середньої вартості перевезення авіапасажирів:

№ з/п	Тип літака	Рейс	Витрати на рейс, грн.	Кількість пасажирів	Середня вартість перевезення, грн.
1	T	R	Z	K	$S = Z/K$
2					
...					
n					
	Разом:		$\Sigma$	$\Sigma$	/n $\Sigma$

Варіант 20. Відомість обліку часу роботи верстатів підприємства:

№ з/п	Тип верстата	Час роботи, год.		Відхилення від плану	
		за планом	фактично	у годинах	у %
1	Z	P	F	$O1 = P - F$	$O2 = O1 \times 100/P$
2					
...					

	Разом	$\Sigma$	$\Sigma$		
--	-------	----------	----------	--	--

Код програми:

```
using System;

namespace lab3_1
{
    public class Manufacture
    {
        public List<object> machines = new List<object>();

        public void addMachine(Machine machine)
        {
            machines.Add(machine);
        }

        public void printMachines()
        {
            foreach (var machine in machines)
            {
                Machine m = (Machine)machine;
                Console.WriteLine($"Type: {m.type}, Plan Time:
{m.plan_time}, Actual Time: {m.actual_time}");
            }
        }

        public int sumPlanTime()
        {
            int sum = 0;
            foreach (var machine in machines)
            {
                Machine m = (Machine)machine;
                sum += m.plan_time;
            }
            return sum;
        }

        public int sumActualTime()
        {
            int sum = 0;
            foreach (var machine in machines)
            {
                Machine m = (Machine)machine;
                sum += m.actual_time;
            }
            return sum;
        }
    }
}
```

```

public class Machine
{
    public string type { get; set; }
    public int plan_time { get; set; }
    public int actual_time { get; set; }

    public double calculateDeviation()
    {
        return Math.Abs(plan_time - actual_time);
    }
}

class Program
{
    static void Main(string[] args)
    {
        Machine m1 = new Machine() { type = "Z", plan_time = 2,
actual_time = 1 };
        Console.WriteLine($"Deviation is {m1.CalculateDeviation()}
hour(s)");

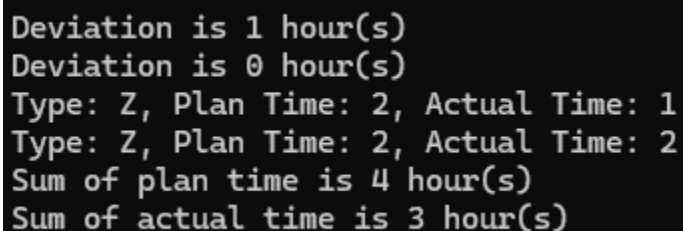
        Machine m2 = new Machine() { type = "Z", plan_time = 2,
actual_time = 2 };
        Console.WriteLine($"Deviation is {m2.CalculateDeviation()}
hour(s)");

        Manufacture man1 = new Manufacture();
        man1.addMachine(m1);
        man1.addMachine(m2);

        man1.printMachines();

        Console.WriteLine($"Sum of plan time is {man1.sumPlanTime()}
hour(s)");
        Console.WriteLine($"Sum of actual time is
{man1.sumActualTime()} hour(s)");
    }
}

```



```

Deviation is 1 hour(s)
Deviation is 0 hour(s)
Type: Z, Plan Time: 2, Actual Time: 1
Type: Z, Plan Time: 2, Actual Time: 2
Sum of plan time is 4 hour(s)
Sum of actual time is 3 hour(s)

```

Рисунок 3.1 – Результат роботи програми

## Завдання 2 (загальне).

Створіть клас «Місто». Необхідно зберігати у полях класу: назву міста, назву країни, кількість жителів у місті, поштовий індекс міста, назви районів міста. Окремі статичні поля повинні зберігати кількість об'єктів створених міст та загальну кількість жителів створених міст. Реалізуйте методи класу для введення даних, виведення даних. Реалізуйте доступ до окремих полів через методи класу.

Код програми:

```
using System;

namespace lab3_2
{
    public class City
    {
        private string _cityName;
        private string _countryName;
        private int _population;
        private int _postalCode;
        private string[] _districts;
        private int _cityCount;
        private int _totalPopulation;

        public City(string cityName, string countryName, int population,
int postalCode, string[] districts)
        {
            _cityName = cityName;
            _countryName = countryName;
            _population = population;
            _postalCode = postalCode;
            _districts = districts;

            _cityCount++;
            _totalPopulation += population;
        }

        public string GetCityName()
        {
            return _cityName;
        }
        public void SetCityName(string cityName)
        {
            _cityName=cityName;
        }
    }
}
```

```
public string GetCountryName()
{
    return _countryName;
}
public void SetCountryName(string countryName)
{
    _countryName=countryName;
}

public int GetPopulation()
{
    return (int) _population;
}
public void SetPopulation(int population)
{
    _population=population;
}

public int GetPostalCode()
{
    return (int) _postalCode;
}
public void SetPostalCode(int postalCode)
{
    _postalCode=postalCode;
}

public string[] GetDistricts()
{
    return _districts;
}
public void SetDistricts(string[] districts)
{
    _districts=districts;
}

public int GetCityCount()
{
    return _cityCount;
}
public int GetTotalPopulation()
{
    return _totalPopulation;
}

public void InputData()
{
    Console.Write("Введіть назву міста: ");
    _cityName = Console.ReadLine();

    Console.Write("Введіть назву країни: ");
    _countryName = Console.ReadLine();

    Console.Write("Введіть кількість жителів: ");
    _population = int.Parse(Console.ReadLine());
}
```

```

        Console.Write("Введіть поштовий індекс: ");
        _postalCode = int.Parse(Console.ReadLine());

        Console.Write("Введіть кількість районів: ");
        int districtCount = int.Parse(Console.ReadLine());

        _districts = new string[districtCount];

        for (int i = 0; i < districtCount; i++)
        {
            Console.Write($"Введіть назву району {i + 1}: ");
            _districts[i] = Console.ReadLine();
        }
    }

    public void OutputData()
    {
        Console.WriteLine($"Місто: {_cityName}");
        Console.WriteLine($"Країна: {_countryName}");
        Console.WriteLine($"Кількість жителів: {_population}");
        Console.WriteLine($"Поштовий індекс: {_postalCode}");
        Console.WriteLine("Райони:");

        foreach (string district in _districts)
        {
            Console.WriteLine($"- {district}");
        }
    }
}

class Program
{
    static void Main()
    {
        City Kremenchuk = new City("Kremenchuk", "Ukraine", 250000,
39600, ["Rakivka", "Molodizhniy", "Centr", "Rynok"]);
        Kremenchuk.OutputData();
    }
}

```

```

Microsoft Visual Studio Debug Console
C#
Місто: Kremenchuk
Країна: Ukraine
Кількість жителів: 250000
Поштовий індекс: 39600
Райони:
- Rakivka
- Molodizhniy
- Centr
- Rynok

```

Рисунок 3.2 – Результат роботи програми з класом “Місто”

### *Контрольні питання*

#### 1. Принципи об'єктно-орієнтованого програмування:

Інкапсуляція: Об'єднання даних та методів в об'єктах, приховування деталей реалізації.

Успадкування: Створення нових класів на основі існуючих, з успадкуванням властивостей та методів.

Поліморфізм: Здатність об'єктів різних класів реагувати на однакові повідомлення по-різному.

Абстракція: Виокремлення суттєвих характеристик об'єкта та ігнорування деталей.

#### 2. Поняття класу й об'єкта, співвідношення між ними:

Клас: Шаблон для створення об'єктів, що описує їхні властивості та поведінку.

Об'єкт: Екземпляр класу, що містить дані та методи, specific for this instance.

Співвідношення: Клас - це модель, а об'єкт - це екземпляр цієї моделі.

#### 3. Життєвий цикл об'єкта в програмі:

Створення: Виділення пам'яті та ініціалізація даних об'єкта.

Використання: Доступ до даних та методів об'єкта, виконання дій.

Видалення: Звільнення пам'яті, що використовується об'єктом.

#### 4. Призначення й варіанти використання посилання this:

Призначення: Доступ до екземпляра класу з його методів.

Варіанти використання:

Доступ до полів та методів класу.

Передача посилання на об'єкт як аргумент методу.

Використання в операторах порівняння.

#### 5. Що означає перевантаження методу:



Перевантаження: Наявність в класі декількох методів з однаковою назвою, але з різними наборами параметрів.

Призначення: Дозволяє використовувати один і той же метод для роботи з різними типами даних.

6. Призначення конструктора:

Призначення: Ініціалізація даних об'єкта при його створенні.

Особливості:

Має ту ж назву, що й клас.

Не має типу повернення.

Може мати різні параметри для ініціалізації даних.

7. Основні особливості конструктора:

Виклик: Автоматично викликається при створенні об'єкта.

Модифікатор доступу: Зазвичай public.

Кількість: Може бути кілька конструкторів з різними параметрами.

Дефолтний конструктор: Автоматично генерується, якщо не визначено жодного.