

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#.

Варіант 15

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, уведення 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 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=O1×100/P
2					
...					
	Разом	Σ	Σ		

using System;

class Record

{

private int number;

private string factory;

public int plannedConsumption;

public int actualConsumption;

public Record(int number, string factory, int plannedConsumption, int
actualConsumption)

{

this.number = number;

this.factory = factory;

this.plannedConsumption = plannedConsumption;

this.actualConsumption = actualConsumption;

}

public int CalculateDeviation()

{

return actualConsumption - plannedConsumption;

```
}
```

```
public double CalculatePercentageDeviation()
{
    return CalculateDeviation() * 100.0 / plannedConsumption;
}
```

```
public override string ToString()
{
    return $"| {number,-5} | {factory,-10} | {plannedConsumption,-20} |
{actualConsumption,-25} | {CalculateDeviation(),-10} |
{CalculatePercentageDeviation(),-10:F2}% |";
}
}
```

```
class Program
{
    static void Main(string[] args)
    {
        Record[] records = new Record[]
        {
            new Record(1, "Завод1", 1000, 950),
            new Record(2, "Завод2", 1500, 1400),
        };
    }
}
```

```
Console.WriteLine("-----
-----");
```

```
Console.WriteLine("| № з/п | Завод | Планове споживання | Фактичне  
споживання | Відхилення | Відсоток відхилення |");
```

```
Console.WriteLine("-----  
-----");
```

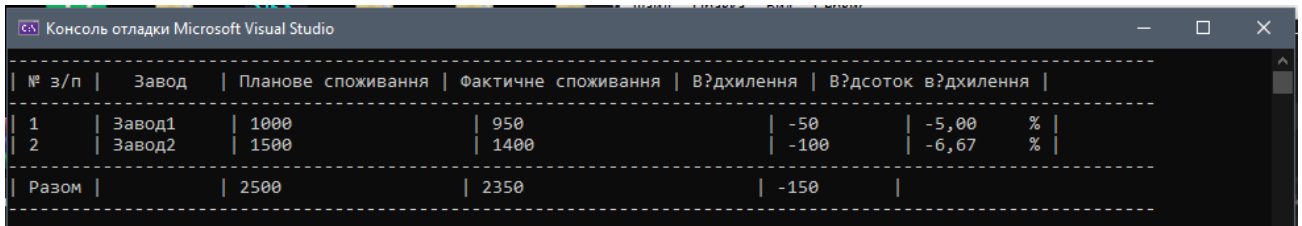
```
foreach (var record in records)  
{  
    Console.WriteLine(record);  
}
```

```
Console.WriteLine("-----  
-----");
```

```
int totalPlanned = 0;  
int totalActual = 0;  
foreach (var record in records)  
{  
    totalPlanned += record.plannedConsumption;  
    totalActual += record.actualConsumption;  
}  
Console.WriteLine($"Разом | | {totalPlanned,-20} | {totalActual,-25} |  
{totalActual - totalPlanned,-10} |");
```

```
Console.WriteLine("-----  
-----");  
}
```

}



Консоль отладки Microsoft Visual Studio

№ з/п	Завод	Планове споживання	Фактичне споживання	В?дхилення	В?дсоток в?дхилення
1	Завод1	1000	950	-50	-5,00 %
2	Завод2	1500	1400	-100	-6,67 %
Разом		2500	2350	-150	

Рис. 1 – Результат виконання програми для обробки відомості споживання електроенергії на заводах міста

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.

```
using System;
```

```
class City
```

```
{
```

```
    private string cityName;
```

```
    private string countryName;
```

```
    private int population;
```

```
    private string postalCode;
```

```
    private string[] districts;
```

```
    private static int cityCount = 0;
```

```
    private static int totalPopulation = 0;
```

```
public City(string cityName, string countryName, int population, string
postalCode, string[] districts)
{
    this.cityName = cityName;
    this.countryName = countryName;
    this.population = population;
    this.postalCode = postalCode;
    this.districts = districts;

    cityCount++;
    totalPopulation += population;
}
```

```
public void InputData()
{
    Console.WriteLine("Введіть назву міста:");
    cityName = Console.ReadLine();

    Console.WriteLine("Введіть назву країни:");
    countryName = Console.ReadLine();

    Console.WriteLine("Введіть кількість жителів у місті:");
    population = Convert.ToInt32(Console.ReadLine());

    Console.WriteLine("Введіть поштовий індекс міста:");
    postalCode = Console.ReadLine();
}
```

```
Console.WriteLine("Введіть назви районів міста (через кому):");  
string districtsInput = Console.ReadLine();  
districts = districtsInput.Split(',');  
  
cityCount++;  
totalPopulation += population;  
}
```

```
public void DisplayData()  
{  
    Console.WriteLine($"Назва міста: {cityName}");  
    Console.WriteLine($"Назва країни: {countryName}");  
    Console.WriteLine($"Кількість жителів: {population}");  
    Console.WriteLine($"Поштовий індекс: {postalCode}");  
    Console.WriteLine("Назви районів міста:");  
    foreach (var district in districts)  
    {  
        Console.WriteLine($"- {district}");  
    }  
}
```

```
public static int GetCityCount()  
{  
    return cityCount;  
}
```

```
public static int GetTotalPopulation()  
{
```

```

        return totalPopulation;
    }
}

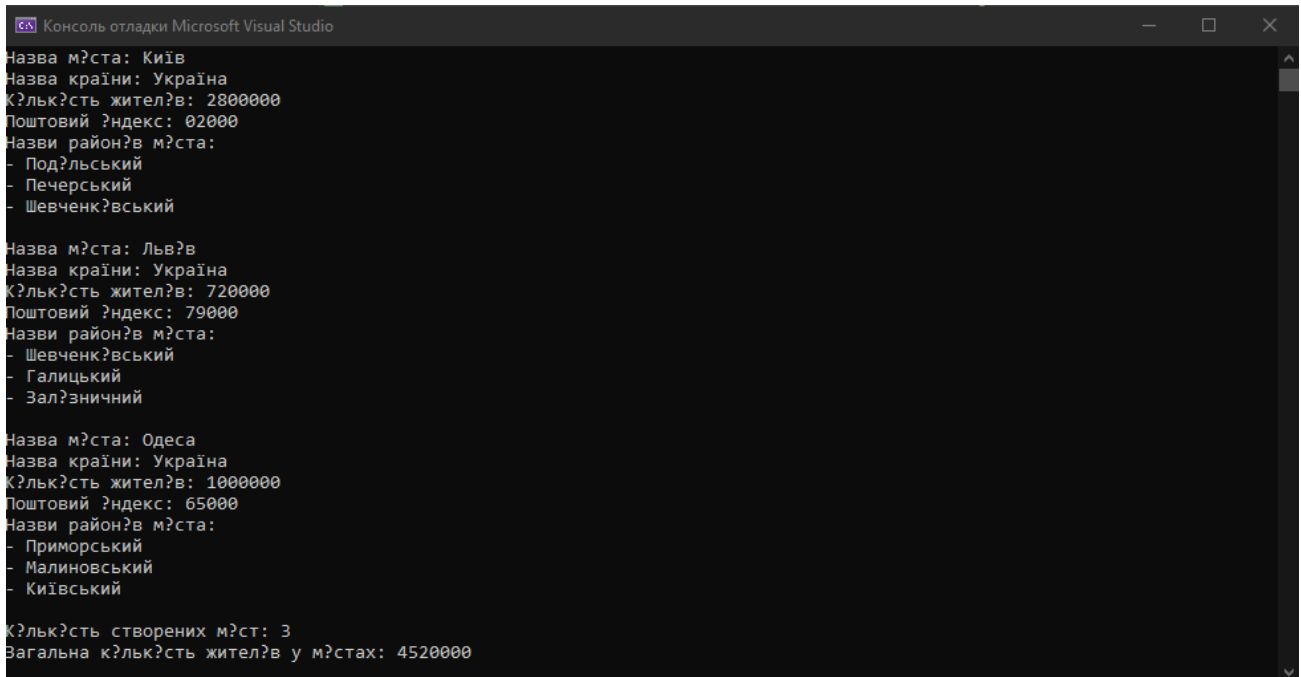
class Program
{
    static void Main(string[] args)
    {
        City[] cities = new City[]
        {
            new City("Київ", "Україна", 2800000, "02000", new string[]
{"Подільський", "Печерський", "Шевченківський"}),
            new City("Львів", "Україна", 720000, "79000", new string[]
{"Шевченківський", "Галицький", "Залізничний"}),
            new City("Одеса", "Україна", 1000000, "65000", new string[]
{"Приморський", "Малиновський", "Київський"}),
        };

        foreach (var city in cities)
        {
            city.DisplayData();
            Console.WriteLine();
        }

        Console.WriteLine($"Кількість створених міст: {City.GetCityCount()}");
        Console.WriteLine($"Загальна кількість жителів у містах:
{City.GetTotalPopulation()}");
    }
}

```


}



```
Консоль отладки Microsoft Visual Studio
Назва м?ста: Київ
Назва країни: Україна
К?льк?сть жител?в: 2800000
Поштовий ?ндекс: 02000
Назви район?в м?ста:
- Под?льський
- Печерський
- Шевченк?вський

Назва м?ста: Льв?в
Назва країни: Україна
К?льк?сть жител?в: 720000
Поштовий ?ндекс: 79000
Назви район?в м?ста:
- Шевченк?вський
- Галицький
- Зал?зничний

Назва м?ста: Одеса
Назва країни: Україна
К?льк?сть жител?в: 1000000
Поштовий ?ндекс: 65000
Назви район?в м?ста:
- Приморський
- Малиновський
- Київський

К?льк?сть створених м?ст: 3
Загальна к?льк?сть жител?в у м?стах: 4520000
```

Рис. 2 – Результат роботи класу «Місто»

Контрольні питання

1. Принципи об'єктно-орієнтованого програмування:
 - 1) Спадкування: Можливість створення нового класу на основі існуючого.
 - 2) Інкапсуляція: Приховання деталей реалізації і надання доступу лише до необхідного.
 - 3) Поліморфізм: Можливість об'єднання різних типів даних або функцій у одному інтерфейсі.
2. Поняття класу й об'єкта, співвідношення між ними:
 - 1) Клас: Це шаблон для створення об'єктів. Визначає поля (дані) та методи (функції), які можуть бути використані об'єктом.
 - 2) Об'єкт: Екземпляр класу. Має власні значення для полів класу та може викликати методи, визначені у класі.

3. Життєвий цикл об'єкта в програмі:

1) Створення (інстанціювання): Коли об'єкт створюється за допомогою оператора `new`.

2) Використання: Об'єкт використовується для виконання певних дій, виклику методів або доступу до полів.

3) Знищення (збирання мусору): Коли об'єкт більше не потрібен програмі, він може бути видалений з пам'яті.

4. Призначення й варіанти використання посилання `this`.
`this` вказує на поточний об'єкт класу. Використовується для відмінення між локальними змінними і членами класу, коли вони мають однакові імена.

5. Що означає перевантаження методу?

Перевантаження методу означає визначення кількох методів з однаковим ім'ям у класі, але з різними параметрами. Це дозволяє викликати один метод за різних умов.

6. Призначення конструктора.

Конструктор – це спеціальний метод класу, який автоматично викликається при створенні об'єкта цього класу. Використовується для ініціалізації об'єкта.

7. Основні особливості конструктора:

1) Ім'я конструктора має збігатися з ім'ям класу.

2) Конструктор може мати параметри або ж не мати їх.

3) Конструктор може бути перевизначений (`override`), якщо в класі є спадкованість.