

## Розділ 5. Основи ООП в С#

### 5.1. Класи в С#. Поля та методи

**Клас** – це шаблон який визначає форму, зміст та поведінку об'єкту. Об'єкт – це екземпляри класу. За допомогою класу реалізується перший з основних принципів об'єктно-орієнтованого програмування – *інкапсуляція*. Інкапсуляція – це об'єднання в одному цілому даних та алгоритмів обробки цих даних. Цей термін також включає в себе приховування даних, тобто приховування від зовнішнього користувача деталей реалізації об'єкта. Дані зберігаються у вигляді *полів*, а алгоритми обробки у вигляді *методів*. Поля, ще іноді називають змінними класу. Оголошення класу в мові С# наступне:

```
[модифікатор_доступу] class <ім'я_класу> {  
    // оголошення полів  
    [мод._доступу] <тип1> <ім'я_поля1>  
        [=початкове_значення1];  
    [мод._доступу] <тип2> <ім'я_поля2>;  
        [=початкове_значення2]  
    // ...  
    // оголошення методів  
    [мод._доступу] <тип_значення1> <ім'я_методу1>  
        ([параметри1]) {  
        //тіло методу  
    }  
    [мод._доступу] <тип_значення2> <ім'я_методу2>  
        ([параметри2]) {  
        //тіло методу  
    }  
    //...  
}
```

Де [модифікатор\_доступу] - дозволяє вказувати рівень доступу до класу або елементів класу, приймає значення з таб. 5.1.

Таблиця 5.1. Модифікатори рівня доступу

Модифікатор	Опис
<b>private</b>	компонент доступний тільки в середині класу, приймається за замовчуванням
<b>protected</b>	компонент доступний тільки в середині класу або в класі нащадку

<b>internal</b>	компонент доступний тільки в середині класу або в цій же програмі (або зборці)
<b>protected internal</b>	компонент доступний тільки в середині класу, в класі нащадку або в цій же програмі (або зборці)
<b>public</b>	компонент доступний без обмежень

Оголошення змінних класів або полів ми вже розглядали на попередніх заняттях, зазначимо тільки, що при створенні об'єкту, якщо не задані початкові значення, поля отримають наступні початкові значення:

Таблиця 5.2. Значення за замовчуванням

Тип поля	Значення за замовчуванням
<b>Всі числові типи</b>	0
<b>bool</b>	false
<b>char</b>	'\0'
<b>Об'єктні типи</b>	null – що означає пусте посилання

Приклад класу, який складається тільки з полів:

```
public class Transformer {    //клас трансформатор
    public int windingCount = 2; //кількість обмоток
    public int temperature;    //поточна температура
    public string power;       //потужність
    public string model;       //марка
    public int yearBuilt;      //рік випуску
}
```

Більш детально зупинимося на методах:

тип\_значення1, тип\_значення2 – тип змінної, що повертається методом. Якщо метод нічого не повертає – цей тип задається ключовим словом **void**, інакше тіло циклу обов'язково повинно містити оператор **return**, за допомогою якого метод повертає значення заданого типу. Якщо тип об'єктний, то повертається посилання (вказівник) на цей об'єкт.

параметри1, параметри2 – необов'язковий список параметрів вигляду:

```
[ref|out|params] <тип_параметру1> <ім'я_параметру1>
[, <тип_параметру2> <ім'я_параметру2>[, ...]]
```

наприклад:

```
public class Power {    //клас електрична потужність
    //... поля якщо потрібно
    // метод обчислює повну потужність
    public float Full(float U, float I) {
        return U * I;
    }
    // метод для обчислення активної потужності
    public float Active(
        float U, float I, float cosFi
    ) {
        return Full(U, I) * cosFi;
    }
}
```

## 5.2. Створення об'єкту. Конструктор класу

Для описаного класу `Transformer` об'єкт створюємо за допомогою ключового слова **new**.

```
Transformer myTrans1 = new Transformer();
```

Тепер ми можемо звертатися до полів та методів створеного об'єкту використовуючи оператор «крапка» (.) разом з посиланням на об'єкт:

```
myTrans1.windingCount = 2;
myTrans1.model = "АТДЦТН-400/330/110/35";
myTrans1.yearBuilt = 2003;
Console.WriteLine("Марка={0}", myTrans1.model);
```

В цьому прикладі `Transformer()` після ключового слова **new** – це **конструктор класу**. Більш детально з конструкторами познайомимося в наступному розділі. Зараз же зазначимо, що це спеціальний метод, який викликається при створенні об'єкту і ім'я його співпадає з іменем класу.

## 5.3. Типи передачі параметрів

Для простих типів передача параметрів може бути за значенням та за посиланням, для об'єктних тільки за посиланням. При **передачі за значенням**, в методі створюється копія змінної-параметра в яку записується значення і при зміні значення в середині методу значення в базовій змінній

не змінюється. Для того, щоб змінювати значення в середині методу використовується *передача за посиланням*.

За замочуванням передача простих параметрів відбувається за значенням. Для передачі простих параметрів за посиланням використовується ключове слово **ref**. Наприклад:

```
using System;
using System.Collections.Generic;
using System.Text;

class Test_ref {
    void Mov(int a, int b) {
        int x = a;
        a = b;
        b = x;
    }
    void Xchg(ref int a, ref int b) {
        int x = a;
        a = b;
        b = x;
    }
    static void Main(string[] args) {
        int a = 3, b = 4;
        Test_ref p = new Test_ref();
        p.Mov(a, b);
        //значення a, b не змінюються
        Console.WriteLine("a={0}, b={1}", a, b);
        p.Xchg(ref a, ref b);
        //a та b обмінялися значеннями
        Console.WriteLine("a={0}, b={1}", a, b);
        Console.ReadKey();
    }
}
```

Виведе на екран наступне:

```
a=3, b=4
a=4, b=3
```

Для передачі простих параметрів за значенням або за допомогою **ref** їм обов'язково задавати початкові значення.

Для більшості випадків від методів достатньо, щоб вони повертали

одне значення, щоб отримати з методу додаткові значення – використовуються ***вихідні параметри***. Вихідні параметри оголошуються за допомогою ключового слова **out**, і вимагають обов'язкової ініціалізації в середині методу. Приклад:

```
using System;
using System.Collections.Generic;
using System.Text;

class Test_out {
    void Power(int c, out int a, out int b) {
        a = c * c;
        b = c * c * c;
    }
    static void Main(string[] args) {
        int a, b, c = 5;
        Test_out p = new Test_out();
        p.Power(c, out a, out b);
        Console.WriteLine("a={0}, b={1}", a, b);
        Console.ReadKey();
    }
}
```

Виведе на екран значення квадрату та кубу числа 5:

a=25, b=125

Щоб мати можливість передати в метод, змінну кількість параметрів можна використати ключове слово **params**. Воно може використовуватись тільки один раз і для останнього параметру, вказуючи, що метод може мати будь-яку кількість параметрів цього типу.

```
using System;
using System.Collections.Generic;
using System.Text;

class Test_params {
    int Sum(params int[] a) {
        int s = 0;
        foreach(int i in a) {
            s += i;
        }
    }
}
```

```

    }
    return s;
}

static void Main(string[] args) {
    Test_params p = new Test_params();
    Console.WriteLine(
        "Sum={0}", p.Sum(1, 2, 3, 4)
    );
    Console.WriteLine(
        "Sum={0}", p.Sum(1, 2, 3, 4, 5)
    );
    Console.ReadKey();
}
}

```

Виведе на екран:

Sum=10

Sum=15

#### 5.4. Перевантаження методів

**Перевантаження методів** - дозволяє використовувати одні й ті самі імена методів змінюючи набір аргументів. Це корисно коли схожі дії треба виконати для різних типів даних, або для різних наборів даних. Наприклад:

```

using System;
using System.Collections.Generic;
using System.Text;

class Test_Overload {
    //для додавання цілих
    int Add(int a, int b) {
        return a + b;
    }
    //для додавання дійсних
    double Add(double a, double b) {
        return a + b;
    }
    //для збільшення на 1
    int Add(int a) {
        return ++a;
    }
    static void Main(string[] args) {

```

```

        Test_Overload t = new Test_Overload ();
        Console.WriteLine("Sum={0}", t.Add(1, 2));
        Console.WriteLine("Sum={0}", t.Add(3.4, 4.4));
        Console.WriteLine("Sum={0}", t.Add(7));
        Console.ReadKey();
    }
}

```

Виведе:

```

Sum=2
Sum=7, 8
Sum=8

```

### Контрольні запитання та завдання

1. Навести власний приклад класу, що пов'язаний з майбутньою спеціальністю. Клас повинен містити поля різних типів та методи з різними наборами аргументів.
2. Які бувають модифікатори рівня доступу?
3. Який синтаксис створення об'єкту?
4. Навести приклади для різних типів передачі параметрів.
5. Що таке перевантаження методів? Створити приклади перевантаження методів.