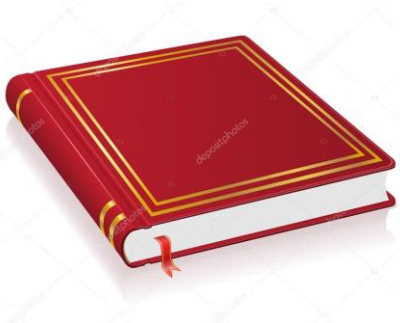
Знайомство з ООП. Поглиблене вивчення мови C#

Тема 1. Основи об’єктно-орієнтованого підходу

Об'єктно-орієнтоване програмування

- Об'єктно-орієнтоване програмування – сучасний підхід до розробки програмного забезпечення, який ґрунтується на уявленні про програму як про сукупність об'єктів, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію наслідування.
- В основі концепції об'єктно-орієнтованого програмування лежить поняття об'єкта — певної сутності, яка об'єднує в собі поля (дані) і методи (виконувані об'єктом дії).

Об’єкти навколо нас



Клас

- Клас – це множина об’єктів, які мають спільну структуру та поведінку.

- Клас – це абстрактний тип даних. З допомогою класу описується деяка сутність (її характеристики і можливі дії). Наприклад, клас може описувати автомобілі, комп’ютери, тварин, людей і т. д. Описавши клас, ми можемо створити його екземпляр – об’єкт. Об’єкт – це вже конкретний представник класу.

Клас Автомобіль

Клас Комп’ютер

Клас Кошеня

Клас Людина

Приклад класу Людина

- Кожна людина має Ім’я та Прізвище.
- Кожна людина прожила певну кількість років і має свій Вік.

```
class Person
{
    public string firstname;     // Ім'я
    public string lastname;      // Прізвище
    public int age;              // Вік
}
```

Приклад створення об’єкта класу Людина

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Person p1 = new Person();
        p1.firstname = "Тарас";
        p1.lastname = "Шевченко";
        p1.age = 15;

        Console.WriteLine($"Ім'я:{p1.firstname}  Прізвище:{p1.lastname}  Вік:{p1.age}");
    }
}
```

Модифікатори доступу до членів класу

Всі члени класу, як і сам клас, мають свій рівень доступу:

- public – доступ до члена можливий з будь-якого місця програми, наприклад, з іншого класу

- private – доступ до члена можливий тільки усередині класу

- Поля класу зазвичай мають модифікатор доступу private, тобто доступ до поля можливий лише усередині класу. Для того, щоб можна було мати доступ до даних поля, в класі оголошують методи, які мають доступ public.

Методи

- Клас може містити методи.
- Метод – це невелика підпрограма, яка виконує, в ідеалі, тільки одну функцію.
- Метод зазвичай призначений для доступу до даних об'єкта та виконання певних дій над ними.
- Метод зазвичай має модификатор доступу public.

Приклади методів

```
private string firstname;     // Ім'я
private string lastname;      // Прізвище
private int age;              // Вік

// Метод для встановлення значення поля "firstname"
2 references
public void SetFirstName(string firstname)
{
    this.firstname = firstname;
}

// Метод для отримання значення поля "firstname"
2 references
public string GetFirstName()
{
    return firstname;
}
```

Приклад використання методів

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Person p1 = new Person();

        p1.SetFirstName("Тарас");
        p1.SetLastName("Шевченко");
        p1.SetAge(15);

        Console.WriteLine($"Ім'я:{p1.GetFirstName()}  Прізвище:{p1.GetLastName()}  Вік:{p1.GetAge()}");

        p1.SetAge(16);

        Console.WriteLine($"Ім'я:{p1.GetFirstName()}  Прізвище:{p1.GetLastName()}  Вік:{p1.GetAge()}");
    }
}
```

Конструктори

- Конструктор – це метод класу, призначений для ініціалізації об'єкта при його створенні.
- Ім’я конструктора завжди збігається з ім'ям класу.
- При оголошенні конструктора, не потрібно вказувати тип, що повертається.
- Конструктор слід оголошувати як public.

Приклад конструктора без параметрів

```
// Конструктор
3 references
public Person()
{
    firstname = "Михайло";
    lastname = "Туренко" ;
    age = 15;
}
```

Приклад конструктора з параметрами

```
// Конструктор
2 references
public Person(string firstname, string lastname, int age)
{
    this.firstname = firstname;
    this.lastname = lastname;
    this.age = age;
}
```

Приклади використання різних конструкторів

```
class Program
{
    0 references
    static void Main(string[] args)
    {
        Person p1 = new Person("Тарас","Шевченко",15);
        p1.AboutMe();

        Console.WriteLine("Введіть і'мя");
        string name = Console.ReadLine();
        Console.WriteLine("Введіть прізвище");
        string lastname = Console.ReadLine();
        Console.WriteLine("Введіть вік");
        int age = int.Parse(Console.ReadLine());
        Person p2 = new Person(name, lastname, age);
        p2.AboutMe();

        Person p3 = new Person("Дарина", "Костенко");
        p3.AboutMe();

        Person p4 = new Person();
        p4.AboutMe();
    }
}
```

Властивості

Властивість – це член класу, який надає механізм доступу до поля класу. При використанні властивості компілятор перетворює це звернення на виклик відповідного неявного методу. Такий метод називається аксессор (accessor). Існує два таких методи: get (для отримання даних) і set (для запису).

Оголошення простої властивості має наступну структуру:

```
[модифікатор доступу] [тип] [ім'я_властивості] {
  get  {  // тіло аксессора для читання з поля }
  set  {  // тіло аксессора для запису у полі }
}
```

Приклади властивостей

```
public string LastName // Властивість
{
    get
    {
        return lastname; // Повертає значення поля
    }

    set
    {
       lastname = value; // Встановлює значенне поля
    }
}
```

Домашнє завдання

Напишіть клас Car (Автомобіль) з полями «Назва» та «Максимальна швидкість» автомобіля. Додайте до класу необхідні методи, конструктори та властивості. Створіть у класі Program два об’єкти класу Car, порівняйте їх швидкість та виведіть назву автомобіля, який має більшу швидкість.