**Контрольні запитання**

**1. Синтаксична конструкція типу class**

Клас у C# описується за допомогою ключового слова class і визначає новий тип, що поєднує дані та поведінку (через поля, властивості та методи):

public **class** **Person**

{

private string name;

public string Name

{

get { **return** name; }

set { name = value; }

}

public void SayHello()

{

Console.WriteLine($"Привіт, я {Name}");

}

}

Клас може мати: поля, властивості, методи, події, індексатори, конструктори, деструктори, оператори, вкладені типи тощо.

**2. Модифікатори доступу (специфікатори)**

Визначають, з яких частин програми доступні члени класу:

|  |  |
| --- | --- |
| **Специфікатор** | **Доступність** |
| public | Доступно з будь-якого місця |
| private | Доступно лише всередині цього класу |
| protected | Доступно в класі й похідних класах |
| internal | Доступно в межах поточної збірки (assembly) |
| protected internal | Доступно в похідних класах або в межах тієї ж збірки |
| private protected | Доступно в похідних класах в межах тієї ж збірки |

**3. Навіщо властивості у класі та їх синтаксис**

Властивості (property) — це безпечний спосіб доступу до полів об’єкта, з можливістю контролювати як читання, так і запис:

private int age;

public int Age

{

get { **return** age; }

set

{

**if** (value >= **0**) age = value;

}

}

Скорочена форма (автовластивість):

public int Age { get; set; }

Властивості забезпечують інкапсуляцію: зовні ми ніби працюємо з полем, але всередині можна виконати перевірки, логіку або обмеження.

**4. Коли створювати автовластивості та як їх ініціалізувати**

Автовластивості — коли не потрібно додаткової логіки в get/set. Синтаксис:

public string Name { get; set; }

Ініціалізація початковим значенням:

public string Name { get; set; } = "Невідомо";

Використовується для простих класів DTO, моделей, де поля виступають лише як контейнери даних.

**5. Що таке індексатор і його призначення**

**Індексатор** дозволяє об'єкту поводитись як масив — звертатися до його елементів через [], але не по індексу, а по будь-якому параметру.

public **class** **Student**

{

private Dictionary<DayOfWeek, string> schedule = new();

public string this[DayOfWeek day]

{

get => schedule.ContainsKey(day) ? schedule[day] : "Вихідний";

set => schedule[day] = value;

}

}

Використання:

student[DayOfWeek.Monday] = "Математика";

Console.WriteLine(student[DayOfWeek.Monday]);

Індексатори використовують для:

* доступу до колекцій, які приховані усередині класу;
* реалізації власної логіки при читанні/записі за ключем;
* зручності, коли об’єкт працює з асоційованими даними.