**1. Базовий клас, породжений клас, ієрархія класів**

* **Базовий клас (superclass)** — це клас, який містить спільні поля/методи, і від якого можуть наслідуватись інші класи.
* **Породжений клас (subclass)** — це клас, який наслідує властивості та методи базового класу і може додавати або перевизначати функціональність.
* **Ієрархія класів** — це структура класів, пов’язаних через наслідування, що створює дерево типів. Вона дозволяє повторне використання коду, узагальнення та поліморфізм.

**Приклад:**

**class** **Animal** { public void Eat() {} }

**class** **Dog** : Animal { public void Bark() {} }

**2. Конструктори базового та породжених класів**

Під час створення об'єкта **спочатку викликається конструктор базового класу**, а потім — породженого.

* За замовчуванням викликається безпараметровий конструктор базового класу.
* Якщо базовий клас не має конструктора без параметрів, у породженому класі треба явно вказати виклик потрібного конструктора:

**class** **Base**

{

public Base(string name) { /\* ... \*/ }

}

**class** **Derived** : Base

{

public Derived() : base("Noname") { }

}

**3. Статичне та динамічне зв’язування**

* **Статичне зв’язування** (compile-time binding): виклик методу визначається під час компіляції. Зазвичай використовується для **неvirtual-методів**.
* **Динамічне зв’язування** (run-time binding): метод визначається під час виконання, залежно від фактичного типу об’єкта. Використовується при поліморфізмі з **virtual/override**.

**class** **A** { public virtual void Print() => Console.WriteLine("A"); }

**class** **B** : A { public override void Print() => Console.WriteLine("B"); }

A obj = new B();

obj.Print(); // Виведе "B" – динамічне зв’язування

**4. Роль віртуальних (virtual) методів**

* Ключове слово virtual дозволяє **перевизначити метод** у похідних класах.
* У похідному класі метод перевизначається через override.
* Віртуальні методи — основа **поліморфізму**, коли один інтерфейс може мати різні реалізації.

**Переваги:**

* Гнучке розширення функціоналу.
* Підтримка динамічного зв’язування.

**5. Абстрактний клас**

* **Абстрактний клас** — це клас, який **не можна створити як об’єкт напряму** (має хоча б один абстрактний метод або позначений як abstract).
* Може містити як реалізовані, так і абстрактні методи.
* Абстрактні методи не мають тіла, і **обов’язково перевизначаються у похідному класі**.

abstract **class** **Shape**

{

public abstract double Area(); // абстрактний метод

}

**class** **Circle** : Shape

{

public override double Area() => Math.PI \* r \* r;

}

**Особливості:**

* Використовується для опису **загального контракту** або сутності.
* Є проміжним шаблоном для конкретних класів.