📘 Դաս 7. Ժառանգում (Inheritance) և Պոլիմորֆիզմ (Polymorphism)

🎯 Նպատակ

Ուսանողը պետք է կարողանա.

- բացատրել **inheritance** գաղափարը,
- տարբերել **base class / derived class**,
- հասկանալ **protected** modifier-ը,
- կիրառել **virtual, override, sealed**,
- բացատրել և կիրառել **polymorphism** (compile-time vs runtime),
- իմանալ `abstract` և `interface`-ի տարբերությունները:

📖 Տեսական մաս

1. Ի՞նչ է ժառանգումը (Inheritance)

- Class A → կարող է տրամադրել հատկություններ ու մեթոդներ Class B-ին:
- **Code reuse** → նորից չգրես նույնը:
- Base class (Parent) → ընդհանուր հատկություններ:
- Derived class (Child) → հատուկ հատկություններ:

```
public class Person
{
    public string Name { get; set; }
    public void Introduce() => Console.WriteLine($"Hi, I'm {Name}");
}

public class Student : Person
{
```

```
    public string StudentId { get; set; }
}
```

Օգտագործում.

```
var s = new Student { Name = "Anna", StudentId = "S001" };
s.Introduce(); // ժառանգվել է Person-ից
```

---

2. Protected modifier

- `private` → միայն class-ի ներսում:
- `protected` → class + ժառանգների ներսում:

```
public class Animal
{
    protected string Type = "Unknown";
}

public class Dog : Animal
{
    public void ShowType() => Console.WriteLine(Type); // OK
}
```

---

3. Virtual, Override, Sealed

Virtual & Override

```
public class Animal
{
    public virtual void Speak()
    {
        Console.WriteLine("Some sound...");
    }
}

public class Dog : Animal
{
    public override void Speak()
    {
        Console.WriteLine("Woof!");
    }
}
```

```
var a = new Animal();
a.Speak(); // "Some sound..."

var d = new Dog();
d.Speak(); // "Woof!"
```

Sealed (արգելում է override)

```
public class Cat : Animal
{
    public sealed override void Speak()
    {
        Console.WriteLine("Meow!");
    }
}
```

---

4. Polymorphism (բազմակերպվածություն)

- **Compile-time (method overloading)** → նույն անուն, տարբեր պարամետրեր:
- **Runtime (virtual/override)** → տարբեր class-եր տարբեր վարք են տալիս նույն մեթոդին:

Օրինակ.

```
List<Animal> animals = new()
{
    new Dog(),
    new Cat(),
    new Animal()
};

foreach (var animal in animals)
{
    animal.Speak(); // տարբեր արդյունքներ (runtime polymorphism)
}
```

---

5. Abstract Classes

- Չի կարելի ստեղծել instance:

- Պարունակում է abstract methods, որոնք պարտադիր է override անել:

```
public abstract class Shape
{
    public abstract double Area();
}

public class Circle : Shape
{
    public double Radius { get; set; }
    public Circle(double r) => Radius = r;
    public override double Area() => Math.PI * Radius * Radius;
}
```

---

6. Interfaces

- Պարզ պայմանագիր (contract) → մեթոդների ստորագրություններ:
- Class-ը պարտավոր է իրագործել:

```
public interface IMovable
{
    void Move();
}

public class Car : IMovable
{
    public void Move() => Console.WriteLine("Car is moving...");
}
```

---

7. Abstract vs Interface

Համեմատություն	Abstract Class	Interface
Instance	Չի կարելի	Չի կարելի
Fields	Ունի	Չի կարող ունենալ (մինչև C# 8.0)
Constructor	Ունի	Չունի

Multiple inheritance	Միայն մեկ	Կարող ես մի քանիսը implement անել
Օգտագործում	Ընդհանուր տրամաբանություն + պարտադիր methods	Պայմանագիր (contract)

💻 Լաբորատոր աշխատանք

Լաբ. 1. Animals Speak

- Base class → Animal (`virtual Speak()`)
- Derived → Dog, Cat, Cow
- Console demo → List, foreach → տարբեր ձայներ:

Լաբ. 2. Shapes Area

- Abstract class Shape (`abstract double Area()`)
- Derived → Circle, Rectangle, Triangle
- Console demo → shapes list → Area-ների գումար:

Լաբ. 3. Interfaces in Action

- Interface IMovable (`Move()`)
- Classes → Car, Human, Robot
- Console demo → List, foreach Move:

🏠 Տնային առաջադրանք

1. Գրիր class hierarchy “Employees”:
    - Base class → Employee (Name, Salary, virtual Work())
    - Derived → Manager, Developer, Designer (override Work)
    - Console demo → List, foreach Work():
2. Գրիր abstract class Appliance (TurnOn, TurnOff abstract methods):
    - Derived → TV, WashingMachine:
    - Console demo → appliances list:
3. Գրիր interface IPayable (PaySalary method):
    - Implement → Employee, Freelancer:
    - Console demo → List, foreach PaySalary:

🧩 Քննարկման հարցեր

- Ինչու է polymorphism-ը կարևոր OOP-ում:
- Ե՞րբ ընտրել abstract class, ե՞րբ interface:
- Կարո՞ղ է class-ը միաժամանակ ժառանգել մեկ class և մի քանի interface:

📎 Ամփոփում

Այս դասից հետո ուսանողը.

- Տիրապետում է **inheritance-ի հիմնական սկզբունքներին**,
- Կարող է կիրառել **virtual, override, sealed**,
- Տարբերում է **abstract class** և **interface**,
- Իմանում է polymorphism-ի գաղափարը իրական օրինակում: