Polymorphism in C# - Illustrated Guide

Polymorphism allows objects of different classes to be treated uniformly through a common interface.

In this guide we use a small animal hierarchy to demonstrate interface-based and virtual/override polymorphism in C#.

IHayvan.cs

IHayvan interface declares the contract every animal must follow.

```
interface IHayvan
{
    string tur();
    bool soyuTukendiMi();
}
```

Yunus_vs_CinNehirYunusu.cs

Yunus is a base class implementing the interface. CinNehirYunusu overrides the virtual members to change behaviour.

```
public class Yunus : IHayvan
{
    public virtual string tur() => "Yunus";
    public virtual bool soyuTukendiMi() => false;
}

public class CinNehirYunusu : Yunus
{
    public override string tur() => "Çin Nehir Yunusu";
    public override bool soyuTukendiMi() => true;
}
```

KralKobra_vs_HintKralKobrasi.cs

KralKobra implements the interface. HintKralKobrasi inherits and overrides behaviour.

```
public class KralKobra : IHayvan
{
    public string tur() => "Kral Kobra";
    public bool soyuTukendiMi() => false;
    public virtual string zehirliMi() => "Zehirli";
}

public class HintKralKobrasi : KralKobra
{
    public override string tur() => "Hint Kral Kobrası";
    public override string zehirliMi() => "Çok Çok Zehirli";
}
```

Program_Istemci.cs

Program creates a random animal and passes it to Istemci which prints properties without knowing exact type.

```
class Program
{
    static void Main()
    {
        IHayvan h = RastgeleHayvanUret();
        new Istemci(h);
    }

    static IHayvan RastgeleHayvanUret()
    {
        Random r = new Random();
        return r.Next(2) == 0 ? new CinNehirYunusu() : new HintKralKobrasi();
    }
}

public class Istemci
{
    public Istemci(IHayvan h)
    {
        Console.WriteLine($"Hayvan Türü: {h.tur()}");
        Console.WriteLine($"Soyu tükendi mi? {h.soyuTukendiMi()}");
    }
}
```

Why use polymorphism?

- Single client code can work with many concrete types.
- Adding new animal types requires no change to existing client code.
- Code stays clean, extensible, and testable.

Execution Flow

Program selects an IHayvan implementation at runtime -> Istemci prints its properties.

The correct overridden methods execute thanks to polymorphism.