Polymorphism in C# - Complete Illustrated Guide

This guide demonstrates interface-based and virtual/override polymorphism using a concise animal hierarchy.

Each code block is shown as a colorhighlighted snippet for readability.

IHayvan.cs

IHayvan interface -- common contract every animal must follow.

```
using System;

public interface IHayvan
{
    // omurgalılar, memeliler v.s
    public string sinif();
    // hayvanin adi
    public string tur();
    // Yumurtlama dogurma v.s
    public bool uremeYaparMi();
    public bool soyuTukendiMi();
    public string zehirliMi();
}
```

Yunus.cs

Yunus base class implements IHayvan; virtual members allow overriding.

```
using System;

public class Yunus : IHayvan
{
    //Virtual metod, kalitimla gelen ve türetilmiş sınıfta override edilebilen metod.
    public virtual bool soyuTukendiMi()
    {
        // Yunusun turu tukenmediginden false dondurecek
        return false;
    }
    public virtual string sinif()
    {
        return "Memeliler";
    }
    public virtual string tur()
    {
        return "Yunus";
    }
    public bool uremeYaparMi()
    {
        return true;
    }
    public string zehirliMi()
    {
        return "Zehirsiz";
    }
}
```

CinNehirYunusu.cs

CinNehirYunusu derives from Yunus and overrides virtual members.

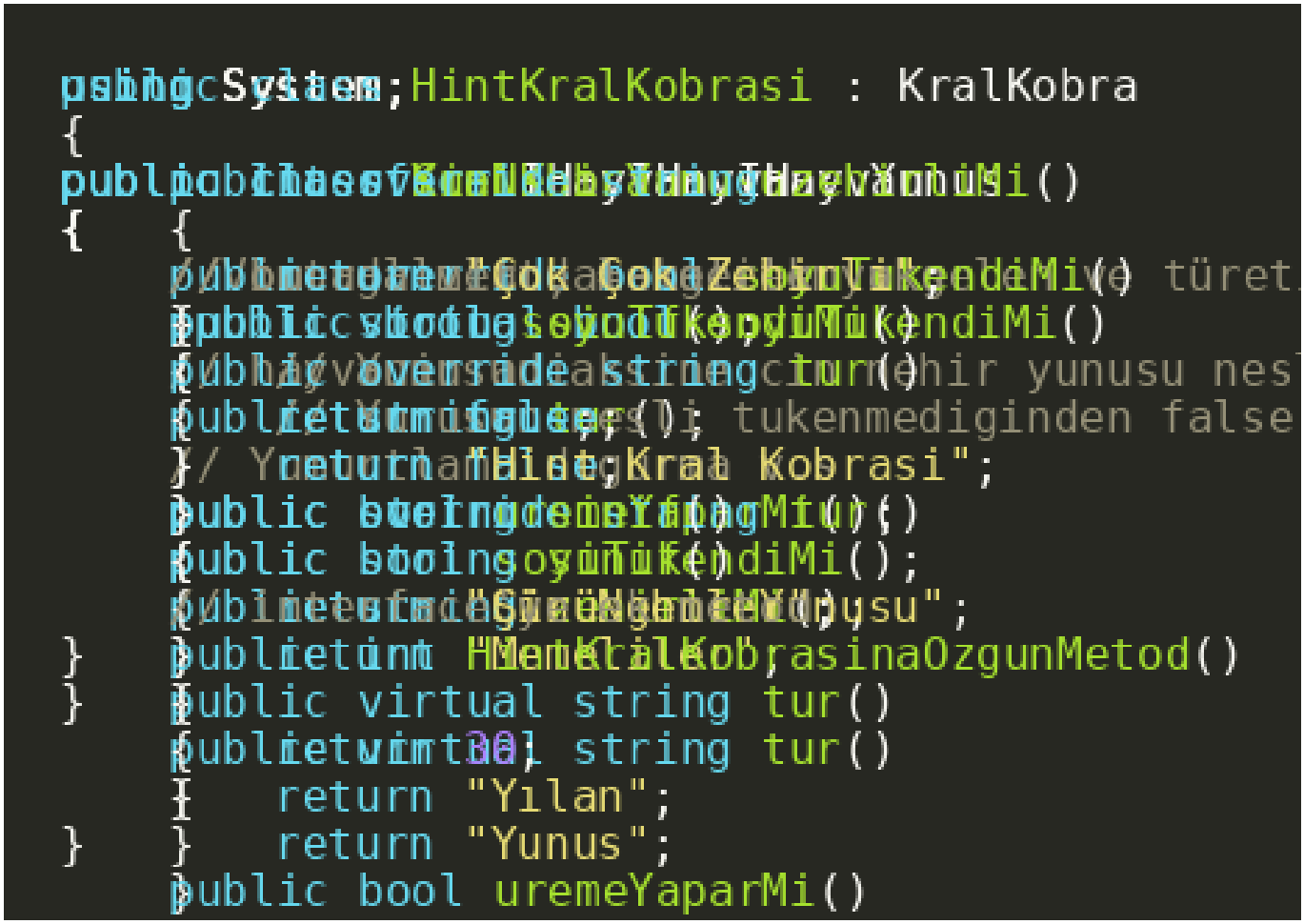
```
using System;

public class CinNehirYunusu: Yunus
{
    public override bool soyuTukendiMi()
    {
        // Yunusun aksine cin nehir yunusu nesli tukenmis bir hayvan
        return true;
    }
    public override string tur()
    {
        return "Cin Nehri Yunusu";
    }
}
```

KralKobra.cs

KralKobra implements IHayvan and marks some members virtual.

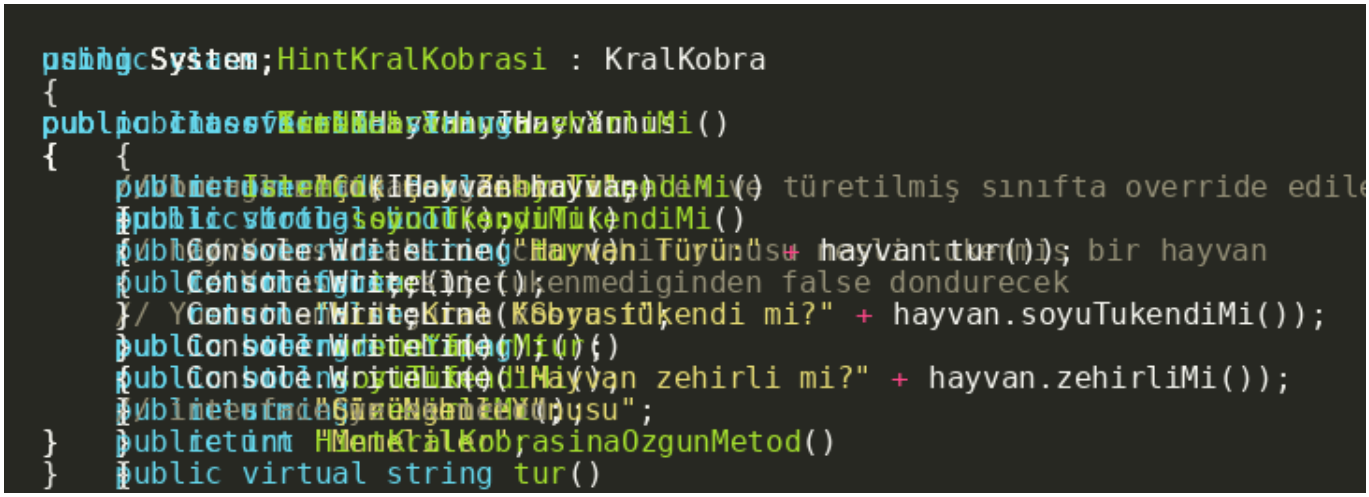
HintKralKobrasi.cs

HintKralKobrasi derives from KralKobra, overrides behaviour, adds extra method.



Istemci.cs

Istemci consumes IHayvan and prints its info, oblivious to concrete type.



Program.cs

Program creates random animal and demonstrates polymorphic behaviour.

```
using System;

public class HintKralKobrasi : KralKobra
{
    public override bool soyuTukendiMi()
    {
        return false; // Override edilmiş sınıfta override edi
    }

    public override string tur()
    {
        return "Hint Kral Kobrası";
    }

    public override bool uremeYaparMi()
    {
        return true;
    }

    public override string zehirliMi()
    {
        return "Zehirli";
    }
}
```

```
using System;

public class Program
{
    public static void Main(string[] args)
    {
        HintKralKobrasi hintKralKobrasi = new HintKralKobrasi();
        Console.WriteLine("Hint Kral Kobrası soyu tükendi mi? " + hintKralKobrasi.soyuTukendiMi());
        Console.WriteLine("Hint Kral Kobrası türü: " + hintKralKobrasi.tur());
        Console.WriteLine("Hint Kral Kobrası üreme yapar mı? " + hintKralKobrasi.uremeYaparMi());
        Console.WriteLine("Hint Kral Kobrası zehirli mi? " + hintKralKobrasi.zehirliMi());
    }
}
```

```
using System;

public class Program
{
    public static void Main(string[] args)
    {
        Random random = new Random();
        int zehirliMi = random.Next(1,5);
        Console.WriteLine("Zehirli mi? " + zehirliMi);
        IHayvan hayvan = rastgeleHayvanUret(zehirliMi);
        Console.WriteLine("Hayvan türü: " + hayvan.tur());
        Console.WriteLine("Soyu tükenmiş mi? " + hayvan.soyuTukendiMi());
        Console.WriteLine("Üreme yapar mı? " + hayvan.uremeYaparMi());
        Console.WriteLine("Zehirli mi? " + hayvan.zehirliMi());
    }

    public static IHayvan rastgeleHayvanUret(int rastgeleSayi)
    {
        if(rastgeleSayi == 1)
        {
            Yunus yunus = new Yunus();
            return yunus;
        }
        else if(rastgeleSayi == 2){
            CinNehirYunusu cinNehirYunusu = new CinNehirYunusu();
            return cinNehirYunusu;
        }
        else if(rastgeleSayi == 3){
            KralKobra kralKobra = new KralKobra();
            return kralKobra;
        }
        else
        {
            Yunus yunus = new Yunus();
            return yunus;
        }
    }
}
```

Why Polymorphism Matters

- Write once, work with any derived type.
- Extend the system with new classes without touching client code.
- Keep business logic decoupled from concrete implementations.

Execution Flow

Program selects an IHayvan implementation at runtime and passes it to Istemci.

Istemci calls methods declared on the interface; the CLR dispatches to the correct override based on actual type.