Polymorphism in C# - Deep Dive

Polymorphism lets objects of different concrete classes be treated through a shared interface. In C# this is achieved via interfaces and the virtual/override mechanism. The following pages use an animal hierarchy to illustrate:

- Interface-based polymorphism - Runtime dispatch with virtual methods - Extensibility benefits

IHayvan.cs

Interface defining the contract that every animal class must implement.

```
1  using System;
2
3  public interface IHayvan
4  {
5      // omurgalılar, memeliler v.s
6      public string sinif();
7      // hayvanin adi
8      public string tur();
9      // Yumurtlama dogurma v.s
10     public bool uremeYaparMi();
11     public bool soyuTukendiMi();
12     public string zehirliMi();
13 }
```

Yunus.cs

Base class (dolphin). Implements IHayvan. Marks key members virtual for overriding.

```cs
using System;

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

CinNehirYunusu.cs

Inherits Yunus and overrides its virtual methods to specialise behaviour.

```cs
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 "Çin Nehir Yunusu";
    }
}
```

KralKobra.cs

Implements IHayvan for snake type. Some methods are virtual for later override.

```
using System;

public class KralKobra : IHayvan
{
    // burada virtuala gerek yok
     public bool soyuTukendiMi()
    {
        return false;
    }
    public string sinif()
    {
        return "Sürüngenler";
    }
    public virtual string tur()
    {
        return "Yılan";
    }
    public bool uremeYaparMi()
    {
        return true;
    }
    public virtual string zehirliMi()
    {
        return "Zehirli";
    }
}
```

HintKralKobrasi.cs

Extends KralKobra. Overrides venom information & species name; adds classspecific method.

```cs
public class HintKralKobrasi : KralKobra
{
    public override string zehirliMi()
    {
        return "Çok Çok Zehirli";
    }
    public override string tur()
    {
        return "Hint Kral Kobrasi";
    }

    // interface'ye ek metod
    public int HintKralKobrasinaOzgunMetod()
    {
        return 30;
    }
}
```

Istemci.cs

Client that receives an IHayvan reference and prints information without knowing concrete type.

```cs
using System;

public class Istemci
{
    public Istemci(IHayvan hayvan)
    {
        Console.WriteLine("Hayvan Türü:" + hayvan.tur());
        Console.WriteLine();
        Console.WriteLine("Soyu tükendi mi?" + hayvan.soyuTukendiMi());
        Console.WriteLine();
        Console.WriteLine("Hayvan zehirli mi?" + hayvan.zehirliMi());
    }
}
```

Program.cs

Entry point. Randomly creates an animal and feeds it into Istemcidemonstrates runtime polymorphism.

```
using System;

namespace Polimorfizm
{
    class Program
    {
        static void Main(string[] args)
        {
            Random random = new Random();
            int rastgeleSayi = random.Next(1,5);
            IHayvan rastgeleHayvan = rastgeleHayvanUret(rastgeleSayi);

            Yunus yunus = new Yunus();
            CinNehirYunusu cinNehirYunusu = new CinNehirYunusu();

            KralKobra kralKobra = new KralKobra();
            HintKralKobrasi hintKralKobrasi = new HintKralKobrasi();

            // KralKobra kralKobra2 = new KralKobra();
            // KralKobra kralKobra3 = new KralKobra();

            Istemci istemci = new Istemci(hintKralKobrasi);

        }
        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?

- Single client code interacts with many animal types through IHayvan.
- Adding a new animal requires zero changes to Istemci or Program.
- The open/closed principle is satisfied system is open for extension, closed for modification.

Execution Flow

1. Program randomly selects an IHayvan implementation.
2. Reference is passed to Istemci.
3. Calls to tur(), soyuTukendiMi() etc. are dispatched at runtime to the overriding methods.