

Çok Biçimlilik (Polimorfizm)

Nesne Tabanlı Programlama

Polimorfizm Türleri

- Statik Polimorfizm
- Dinamik Polimorfizm

Static Çok Biçimlilik

- Static, ileride göreceğimiz bir kavramdır.
- Şimdilik sadece Polimorfizm çerçevesinde Static Polimorfizm'i değerlendireceğiz.
- Static polimorfizm; derleme zamanında sergilenen polimorfizm'dir. Hangi fonksiyonun çağrılacağına derleme zamanında karar verilir.
- C#'da static polimorfizm deyince aklımıza **Metot Overloading** terimi gelmelidir.
- **Metot Overloading**; aynı isimde birbirinden farklı imzalara sahip olan metotların tanımlanmasıdır. Ya da başka deyişle bir isme birden fazla farklı türde metot yüklemektir. Haliyle burada bir metodun birden fazla formunun olması **polimorfizm**'ken, bunlardan kullanılacak olanın derleme zamanında bilinmesi **statik polimorfizm** olarak nitelendirilmektedir.

```
class Matematik
{
    0 references
    public long Topla(int s1, int s2)
        => s1 + s2;
    0 references
    public long Topla(int s1, int s2, int s3)
        => s1 + s2 + s3;
    0 references
    public long Topla(int s1, int s2, int s3, int s4)
        => s1 + s2 + s3 + s4;
}
```

Dinamik Çok Biçimlilik

- Dinamik polimorfizm; çalışma zamanında sergilenen polimorfizm'dir. Yani hangi fonksiyonun çalışacağına run time'da karar verilir.
- C#'da dinamik polimorfizm deyince akla **Metot Override** gelmektedir.
- **Metot Override**; base class'ta virtual olarak işaretlenmiş metotların derived class'ta override edilerek ezilmesi/yeniden yazılması işlemidir. Haliyle burada aynı isimde birden fazla forma sahip fonksiyonun olması **polimorfizm**'ken, bunlardan hangisinin kullanılacağına çalışma zamanında bilinmesi **dinamik polimorfizm** olarak nitelendirilmektedir.

```
class Arac
{
    1 reference
    public virtual void Start()
    {
        Console.WriteLine("Araç çalıştı.");
    }
}
0 references
class Taksi : Arac
{
    1 reference
    public override void Start()
    {
        Console.WriteLine("Taksi çalıştı.");
    }
}
```

Polimorfizm Durumlarında Tür Dönüşümleri

- Polimorfizm, OOP'nin temel prensiplerinden biridir. Bir nesne, bu tür dönüşümleri kullanarak...

Dikkat edersen eğer Polimorfizm durumlarında kalıtsal açıdan üst bir referans ile işaretlenebilmiş herhangi bir nesneyi kendi türünden işaretleyebilmek için Cast operatörünü kullanarak object türüne özel olan UnBoxing'e benzer bir hamlede bulunmuş oluyoruz...

Buradan anlıyoruz ki, object türünde gerçekleştirilen UnBoxing durumu esasında object türü ile gerçekleştirilebilen Polimorfizm'in bir sonucudur...

Bu durumun terside geçerlidir. Yani ilgili nesne kendi türünden kalıtım olarak ataları olan diğer türlere Cast edilebilir.

```
C c = new C( );
```

```
A a = (A)c;
```

Dikkat ederseniz bu işlem için Cast operatörü kullanılmaktadır.

Örneğin, burada gördüğümüz üzere A türünden olan a referansındaki özünde C türünden nesne kendi türünden bir referansla işaretlenmiştir.

Şimdi gelin olayı biraz daha pekiştirebilmek ve bi tık daha derinleştirebilmek için konuyu dijital tahtamızla ele alalım.

Polimorfizm Durumlarında Tür Dönüşümleri

Polimorfizm durumlarında tür dönüşümünü gerçekleştirebilmek için **Cast** ya da **as** operatörleri kullanılabilir.

Misal;

Cast

Üst türden alt türe kalıtsal ilişkide dönüşüm sağlar.

```
A a = new C();
```

```
C c = (C)a;
```

Eğer ki, kalıtsal ilişki olmayan herhangi bir türe dönüştürülmeye çalışılırsa derleyici hatası verecektir.

Yok eğer kalıtsal ilişkide olup fiziksel nesnenin hiyerarşik altında olan bir türe dönüştürülmeye çalışılırsa run time hatası verecektir.

```
D d = (D)a;
```

Misal; D türü A'dan kalıtım almıyorsa eğer hiyerarşide yer edinmeyeceğinden dolayı bu durumda derleyici hatası verecektir. Yok eğer kalıtsal olarak C'nin altında A'nın torunu ise fiziksel C nesnesinin kendisinden küçük olan D referansı ile işaretlenmesi Polimorfizm mantığı gereği mümkün olamayacağı için run time hatası verecektir.

```
A a = new C() { X = "1", Y = "2", Z = "3" };|
```

```
D d = (D)a;
```



Exception Unhandled

System.InvalidCastException: 'Unable to cast object of type 'polimorfizm.C' to type 'polimorfizm.D'.'

[View Details](#) | [Copy Details](#) | [Start Live Share session...](#)

▸ Exception Settings

Tersine olarak, kalıtsal ilişkide alt türden üst türe cast operatörü ile de bir dönüşüm sağlamaktadır.

```
C c = new C();
```

```
A a = (A)c;
```

Burada da yine kalıtsal ilişki gerekmekte aksi takdirde derleyici hatası ile karşılaşılabilmektedir.

Polimorfizm Durumlarında Tür Dönüşümleri

as

Cast gibi kalıtımsal ilişki olan türler arasında referans dönüşümü yapabilmemizi sağlayan operatördür.

```
A a = new C();
```

```
C c = a as C;
```

Dönüşüm esnasında hiyerarşik olarak tüm türlere dönüşüm sağlar. Lakin kalıtımsal ilişkide bulunmayan türlerde derleyici hatası verecektir.

Ya da kalıtımsal ilişkide olup fiziksel nesnenin türünden daha alt hiyerarşide olan nesnelere dönüştürülmeye çalışıldığında Polimorfizm mantığı gereği ilgili referans o nesneyi karşılayamayacağından **run time hatası VERMEYECEK!** geriye **null** dönecektir.

```
A a = new C() { X = "1", Y = "2", Z = "3" };
D c = a as D;
```

c null

Cast operatörünün as operatöründen farkı; biri dönüşüm sağlanamıyorsa hata fırlatırken(Cast), diğeri null dönmektedir(as)

Polimorfizm Durumlarında Tür Dönüşümleri

is

is operatörü kalıtımsal ilişkiye sahip nesnelerin Polimorfizm özelliğine nazaran fiziksel olarak hangi türde olduğunu veren bir operatördür.

```
A a = new C() { X = "1", Y = "2", Z = "3" };  
Console.WriteLine(a is B);  
Console.WriteLine(a is C);  
Console.WriteLine(a is A);  
Console.WriteLine(a is D);
```

Microsoft Visual Studio Debug Console

```
True  
True  
True  
False
```

Haliyle dikkat ederseniz fiziksel nesnenin kalıtım hiyerarşisine uygun olan türlere 'true' olmayan türlere ise 'false' sonucunu döndürmektedir. Kalıtımsal ilişki olmayan sınıflarla yapılacak kontrolde de beklenildiği gibi 'false' değeri döndürecektir.

Haliyle çok biçimlilik uygulanmış bir nesnenin ihtiyaç doğrultusunda (uygun olan) farklı bir türe dönüştürülebilmesi için işi garantiye alabilmek adına önce **is** kontrolü ardından **Cast** ya da **as** operasyonu sağlanması kafiidir.