SANAL YAPILAR

VIRTUAL & OVERRIDE

NESNE TABANLI PROGRAMLAMA

- Bir nesne üzerinde var olan tüm member[illegible]amamı derleme zamanında belirgindir.
- Yani, derleme aşamasında han[illegible]angi metotların çağrılabileceği bilinmektedir.

```
class MyClass
{
   public int MyProperty { get; set; }

   public void MyMethod( )
   {

   }
}
```

- Sanal yapılar ile derleme zaman[illegible]bilgi run time(çalışma zamanın)da belirlenebilmektedir. Yani ilgili nes[illegible]odu kullanacağı bilgisi run time'da kararlaştırılır.

Hoca bunu zaten Name Hiding ile yapabiliyorduk!"

Evet, Name Hiding ile bir sınıftaki herhangi bir member'ı ondan türeyen torunlarda oluşturabiliyoruz ve buradaki yaşanan çakışmaya da Name Hiding diyoruz.

Lakin, sanal yapılarda olay bu şekilde değildir. Bir sınıfta bildirilen sanal yapı(metot ya da property) bu sınıftan türeyen torunlarında ezilebilmekte yani devre dışı bırakılıp yeniden oluşturulabilmektedir.

Yani sanal yapılanmalarda Name Hiding'de olduğu gibi bir isimsel çakışmadan ziyade üstten gelen bir yapının işlevini iptal edip yeniden yapılandırmak vardır.

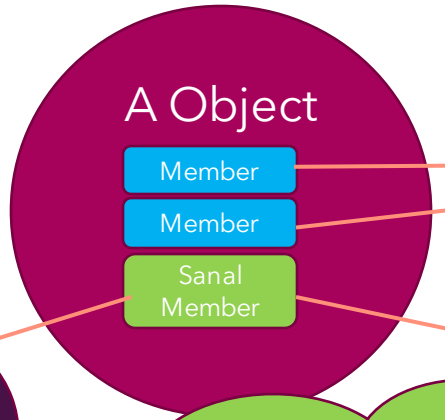
İşte burada bir sınıfta tasarlanmış sanal yapının işlevinin iptal edilip edilmeme durumuna göre tanımlandığı sınıftan mı yoksa bu sınıfın torunlarından mı çağrılacağının belirlenmesi run time'da gerçekleşecektir.

...sinde
...ldiri... ...tan
türeyen al... ...a da tekrar
...lebilen yapılardır.

...bilar metot ya da property
...lir.

SA
YAPIL
GÖZLEYE IM

Vee unutma! Metot ya da property fark etmez! Bir sanal member'ın hangi türe ait olduğunun bu şekilde run time'da belirlenmesine Geç Bağlama(Late Binding) denir!



Bu member'lar sanal olmadığı için derleme zamanı hangi nesneden çağrılabilecekleri bilinmektedir.

Lakin bu member sanal ğu için kendisinden alan torunlarında me/iptal edilip a durumuna bject'ten ya da B rılacaktır.

Sanal yapılarda çağrılan member'ın hangi türe ait olduğu Run Time'da belirlenir...

d davranışın rarı ancak Run ime'da anlaşılacağı için Run Time'da verilmektedir!

k eğer ezilir/iptal edilir ve yeniden yazılırsa, artık bu işlemin apıldığı torununun bir member'ı olacaktır ve B Object'tinden çağrılacaktır.

SANAL YAPILAR NE İÇİN KULLANILIR?

Bir sınıfta tanımlanmış olan herhangi bir member'ın kendisinden türeyen alt sınıflarda Name Hiding durumuna düşmemeksizin ezilip/yeniden yazılıp yapılandırılması için kullanılır.

Peki bu zorunlu mudur?

Yani bir sanal yapı illa ki kendisinden türeyen torunlarda ezilmek/yeniden yazılmak zorunda mıdır?

Tabi ki de hayır! Yani bir member sanal yapıldı diye illa ki kendisinden türeyen alt sınıflarda ezilmez zorunda değildir! Ama ezilmek istenirse de Name Hiding'e bulaşmadan direkt ilgili sınıfın bir member'ı olacak şekilde çalışılmasını sağlamış olur.

BİR SINIFTA SANAL YAPI NASIL OLUŞTURULUR?

VIRTUAL KEYWORDÜ

Bir sınıfta sanal yapı oluşturabilmek için ilgili member'ın(metot ya da property) imzasını **virtual** keywordü ile işaretlemek yeterlidir.

public virtual ya da virtual public

```
class MyClass
{
   public void MyMethod( )
   {

   }
}
```

Normal Metot

```
class MyClass
{
   public virtual void MyMethod( )
   {

   }
}
```

Sanal Metot

```
class MyClass
{
   public int MyProperty { get; set; }
}
```

Normal Property

```
class MyClass
{
   virtual public int MyProperty { get; set; }
}
```

Sanal Property

SANAL YAPILAR NASIL EZİLİR?

OVERRIDE KEYWORDÜ

Bir class'ta **virtual** ile işaretlenerek sanal hale getirilmiş bir member(metot, ya da
torunlarında ez
class't

virtual bir
override ile
direkt olarak
etmek" diyece

"**override**" keywordünü ileride Abstract Class'ların implemantasyonunda da kullanacağız.
Şimdilik ne olduğunu salla :)
Zamanı gelince konuşacağız.
Öylesine not olsun çorba olsun diye söyleyelim dedim...

```
class MyClass2 : MyClass
{
    public override void MyMethod( )
    {

    }
}
```

- da atalard tual ile
lastırılm erhangi
override
- ki
rride
ede n kesinlikle
virtu sı
gerek
- virtual ile iş bir member kesinlikle override EDİLEMEZ!

İKİDEN ÇOK HİYERARŞİK KALITIM DURUMLARINDA OVERRIDE DURUMU

Bir class'ta virtual ile işaretlenmiş herhangi bir member illa ki direkt o class'tan türeyen 1. dereceden class'lar da override edilmek mecburiyetinde değildir!

İhtiyaca binaen alt seviyede ki torunlardan herhangi birinde override edilebilir.

Lakin böyle bir durumda kritik bir durum vardır! O da ilgili soyut member'ın en son override edildiği Object'ten sonra geçerli olduğudur.

Ve o object'ten sonra hiyerarşik olarak türetilen sınıflar varsa onlarda da **override** işlemi gerçekleştirilebilir.

SANAL YAPILAR ÖRNEK 1

SANAL YAPILAR ÖRNEK 2

SANAL YAPILAR HAKKINDA ÖZET

- Sanal yapılar OOP'de Polimorfizm(Çok Biçimlilik)'i uygulayan yapılardır. (İleride göreceğiz)
- Sanal yapıların en önemli özelliği Geç Bağlam(Late Binding)'dir.
- Eğer bir member sanal olarak bildirilmemişse, derleyici nesnelerin tür bilgisinden faydalanarak derleme zamanında hangi nesneden ilgili member'ın çağrılacağını bilir.
- Eğer bir member sanal olarak bildirilmişse, ilgili member'ın hangi nesne üzerinden çağrılacağı run time'da belirlenir.
- Hangi member'ın run time'da belirlenmesine Geç Bağlama(Late Binding) denir.
- Sanal yapı oluşturabilmek için ilgili member'ı virtual keywordü ile işaretlemeliyiz.
- Türeyen sınıflarda sanal yapıyı ezebilmek için override keywordü kullanılır.
- Türeyen sınıflar sanal yapıları override etmek zorunda değildir.
- Static yapılanmalar sanal olarak bildirilemezler! (İleride göreceğiz)