Masaüstü Uygulama Geliştirme

Hafta-13

Öğr. Gör. Zafer SERİN

Record Yapılanması

- C# 9.0'da herhangi bir nesnenin propertylerine ilk değerlerinin verilmesi ve sonraki süreçte bu değerlerin değiştirilmemesini garanti altına almamızı sağlayan **Init-Only Properties** özelliği gelmiştir.
- Bu özellik sayesinde nesnenin sadece ilk yaratılış anında propertylerine değer atanmakta ve böylece iş kuralları gereği runtime'da değeri değişmemesi gereken nesneler için ideal bir önlem alınmaktadır.

Record Yapılanması

- Init-Only Properties, developer açısından süreç esnasında değiştirilmemesi gereken property değerlerinin yanlışlıkla değiştirilmesinin önüne geçmekte ve böylece olası hata ve buglardan yazılımı arındırmaktadır.

Record Yapılanması

- Aşağıdaki gibi tanımlanmış bir Bitki classından nesne üretirken Yas ve Tur propertylerini başlangıç değerleri verilebilir.

```
class Bitki
{
    0 references
    public int Yas { get; set; }
    0 references
    public string Tur { get; set; }
}
```

```
static void Main(string[] args)
{
    Bitki b1 = new Bitki()
    {
        Yas = 5,
        Tur = "Çam"
    };
    Console.WriteLine(b1.Yas); // 5
    Console.WriteLine(b1.Tur); // Çam
}
```

Record Yapılanması

- Bu propertylerin sadece okunabilir(readonly) olması istenirse set blokları kaldırılarak sadece get blokları bırakılabilir; ancak bu durumda ilgili propertylere nesne başlatılırken başlangıç değeri atanmasına izin verilmeyecektir.
- Nesne başlatılırken(Object Initializers) Propertylere değer atamak için Inıt-Only Properties kullanılabilir.
- Bunun için Property tanımlanırken get yapısına ek olarak **init** yapısı tanımlanmalıdır!

Record Yapılanması

- init get keywordü olmaksızın kullanılamaz!
- Ayrıca yapısı gereği bu semantikte set bloğuda kullanılamaz!
- get; init; konsepti bir araya geldiği zaman bu property readonly(sadece okunabilir) olmakta ve ilk değerlerini constructor(yapıcı metot) veya direk tanımlandığı yerde alabileceği gibi init sayesinde object initializerdan(obje başlatıcı - nesne oluşturma süreci) da alabilmektedir.
- C#'ta bir fieldın(class memberı) sadece okunabilir olması için **readonly** anahtar sözcüğü kullanılabilir!

Record Yapılanması

- readonly bir field üzerinde işlem yapmamız gerekiyorsa eğer aşağıdaki gibi init bizlere eşlik edebilir.

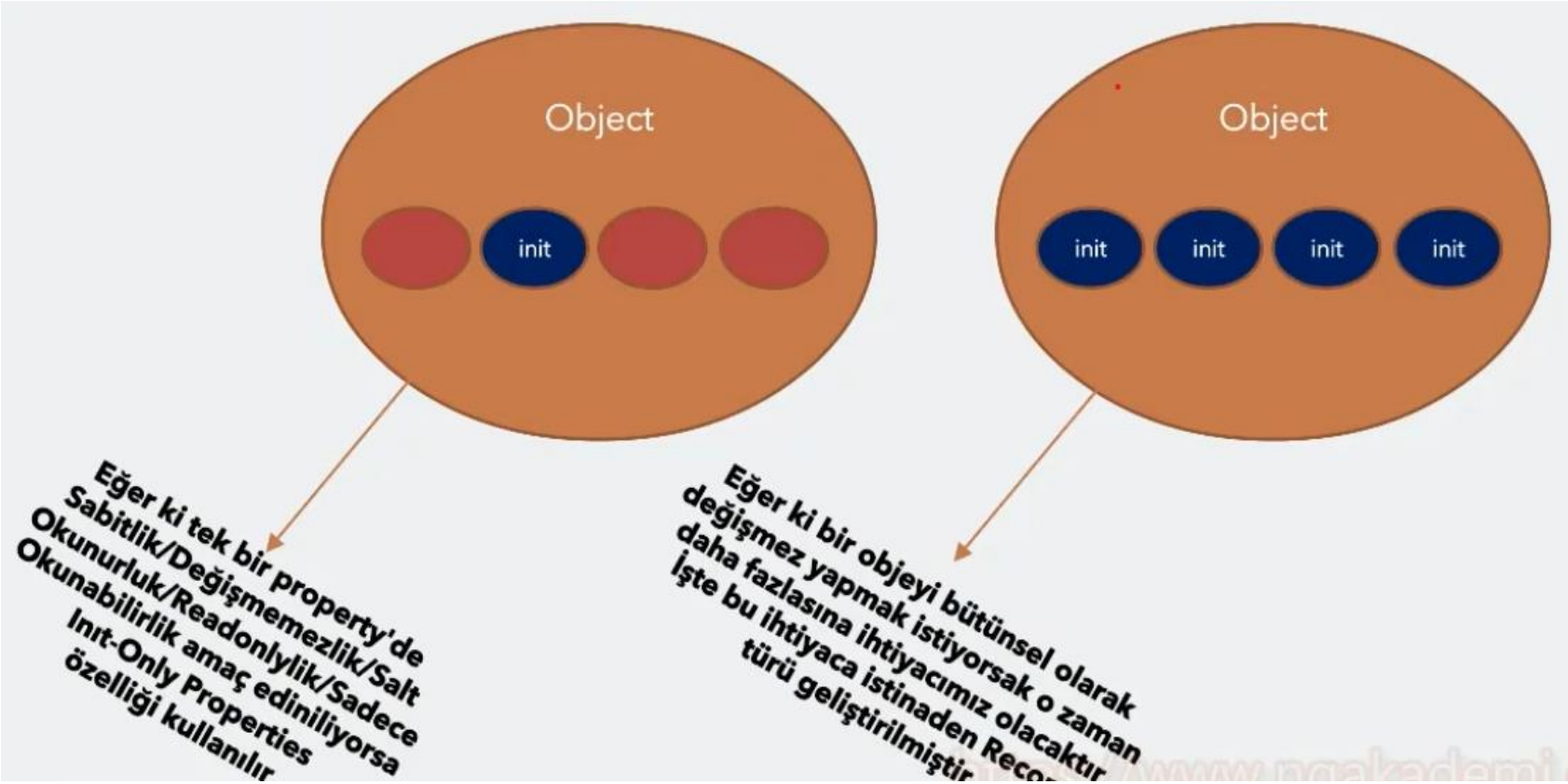
```
class Kedi
{
    private readonly int yas;
    1 reference
    public int Yas
    {
        get
        {
            return yas;
        }

        init
        {
            yas = value;
        }
    }
}
```

```
Kedi k1 = new Kedi()
{
    Yas = 5
};
```

Record Yapılanması

- C# 9.0 ile gelen Init-Only Properties özelliği, nesne üretimi dışında değişmez değerler oluşturulması için constructor ve tanımlama anında işaretleme(auto property initizalizers) yapısının yanında object initializers yapısının kullanılmasını sağlıyordu.

Record Yapılanması

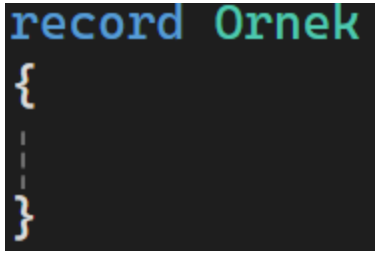


Record Yapılanması

- Record bir objenin topyekün olarak sabit/değişmez olarak kalmasını sağlamakta ve bu durumu güvence altına almaktadır.
- Nesne ön plandaysa bu class, nesnenin değerleri ön plandaysa bu record olur.
- Buna bağlı olarak recordlar içerisinde veri barındıran hafif classlar olarak düşünülebiliri. Recordlar kesinlikle değer türlü değildir! Heapte tutulurlar. Recordlar özünde birer classtır sadece nesnesinden ziyade değerleri ön plana çıkmıştır.

Record Yapılanması

- Class'lardan üretilen nesneler aynı değerlere sahip olsalar dahi farklı referanslar ile işaretlenip eşitlik durumu kontrol edilirse 'False' çıktısı alınırken Recor'lardan üretilen nesneler aynı değerlere sahip olurlar ve farklı referans ile işaretlenip eşitlik durumu kontrol edilirse 'True' çıktısı alınacaktır! Bunun sebebi recordlarda değerlerin ön planda olmasıdır. C#'ta record aşağıdaki gibi tanımlanabilir.



Sanal Yapılar Virtual & Override

- Bir nesne üzerinde var olan tüm memberlar(üyeler) derleme zamanında belirgindir.
- Yani, derleme aşamasında hangi nesne üzerinden hangi metotların çağrılabileceği bilinmektedir.

```
class MyClass
{
    0 references
    public int MyProperty { get; set; }
    0 references
    public void MyMethod() { }
}
```

- Derleme zamanından olaya bakıldığında MyClass sınıfından üretilen tüm nesnelerde MyPproperty ve MyMethod çağrılabilecektir.

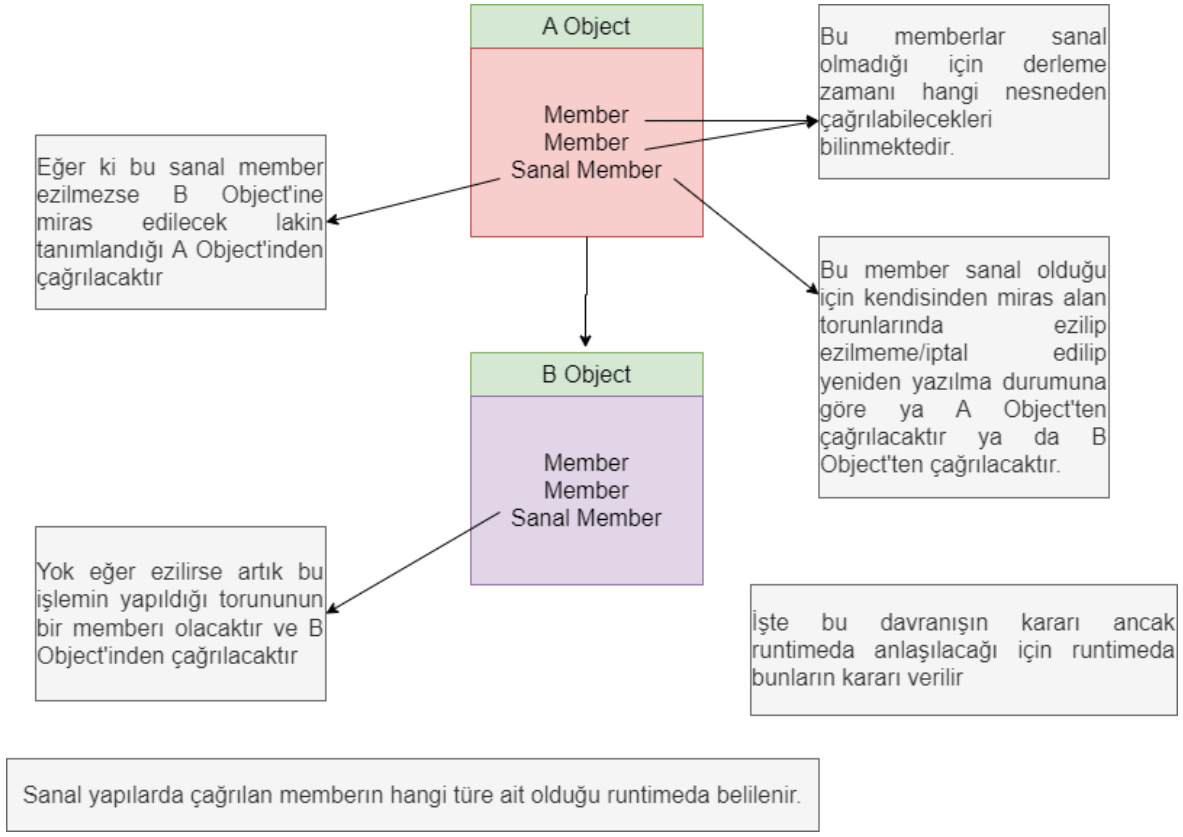
Sanal Yapılar Virtual & Override

- Sanal yapılar ile derleme zamanında kesin olarak bilinen bu bilgi runtime(çalışma zamanı) için de belirlenebilmektedir. Yani ilgili nesnenin hangi metodu kullanacağı runtimeda kararlaştırılır.
- Sanal yapılar, bir sınıf içerisinde bildirilmiş olan ve o sınıftan türeyen alt sınıflarda da tekrar bildirilebilen yapılardır.
- Bu yapılar metot, property veya indexer olabilir.
- Sanal yapılanmalarda üstten gelen bir yapının işlevi iptal edilip yeniden yapılandırılır.

Sanal Yapılar Virtual & Override

- Bir sınıfta tasarlanmış sanal yapının işlevinin iptal edilip edilmeme durumuna göre tanımlandığı sınıftan mı yoksa bu sınıfın torunlarından mı çağrılacağının belirlenmesi runtime da gerçekleşecektir.

Sanal Yapılar Virtual & Override



Sanal Yapılar Ne için Kullanılır?

- Bir sınıfta tanımlanmış olan herhangi bir memberın kendisinden türeyen alt sınıflarda ezilmesi için kullanılır.
- Peki bu zorunlu mudur? Yani bir sanal yapı illaki kendisinden türeyen torunlarda ezilmek zorunda mıdır?
- Tabiki değildir. Yani bir member sanal yapıldı diye illa ki kendisinden türeyen alt sınıflarda ezilmek zorunda değildir! Ama ezilmek istenirse direkt ilgili sınıfın bir memberı olacak şekilde çalışılmasını sağlamış olur.

Bir Sınıfta Sanal Yapı Nasıl Oluşturulur?

- Bir sınıfta sanal yapılar oluşturabilmek için ilgili memberın imzasını **virtual** keywordü ile işaretlemek yeterlidir.
- public virtual ya da virtual public şeklinde kullanılabilir.

Bir Sınıfta Sanal Yapı Nasıl Oluşturulur?

- Bir sınıfta sanal yapılar oluşturabilmek için ilgili memberın imzasını **virtual** keywordü ile işaretlemek yeterlidir.
- public virtual ya da virtual public şeklinde kullanılabilir.

```
class MyClass
{
    // Normal metot ve property tanımlama
    0 references
    public int MyProperty { get; set; }
    0 references
    public void MyMethod() { }
}
```

```
class MyClass
{
    // Sanal metot ve sanal property tanımlama
    0 references
    public virtual int MyProperty { get; set; }
    0 references
    virtual public void MyMethod() { }
}
```

Bir Sınıfta Sanal Yapı Nasıl Oluşturulur?

- Bir class'ta virtual ile işaretlenerek sanal hale getirilmiş bir member, bu classtan miras alan torunlarında ezilmek isteniyorsa eğer ilgili classta imzası **override** keywordü ile işaretlenmiş bir vaziyette tekrardan aynı member oluşturulur.

Bir Sınıfta Sanal Yapı Nasıl Oluşturulur?

```
class MyClass
{
    1 reference
    virtual public void MyMethod() { }
}

0 references
class MyClass2 : MyClass
{
    1 reference
    public override void MyMethod()
    {
    }
}
```

- Base classta virtual ile işaretlenen member ezilmek zorunda değil; ancak bir torun class base classtaki herhangi bir memberı ezecekse o member mutlaka virtual ile işaretlenmiş olmalı!

Bir Sınıfta Sanal Yapı Nasıl Oluşturulur?

- Bir classta virtual ile işaretlenmiş herhangi bir member illa ki direkt o classtan türeyen 1. dereceden classlarda override edilmek zorunda değil! İhtiyaca göre alt seviyedeki torunlardan herhangi birinde override edilebilir.
- Ancak böyle bir durumda virtual member en son override edildiği Object'ten sonra geçerli olur.

Çok Biçimlilik(Polimorfizm) Nedir?

- Polimorfizm esasında kalıtım gibi bir biyolojik terimdir.
- Polimorfizm biyolojik açıdan iki veya daha fazla farklı fenotipin aynı tür popülasyonunda bulunmasıdır.
- Polimorfizm yazılımda iki veya farklı nesnenin aynı tür classlardan(referanslardan) referans almasıdır.(işaretlenebilmesidir).
- Bir diğer tanımda Polimorfizm bir nesneyi birden fazla form ile tarif edebilmektir.

Çok Biçimlilik(Polimorfizm) Nedir?

- OOP'de polimorfizm, iki ya da daha fazla nesnenin aynı tür sınıf tarafından karşılanabilmesi/referans edilebilmesidir.
- Bir başka deyişle, bir nesnenin birden fazla farklı türdeki referans tarafından işaretlenebilmesi polimorfizmdir.
- Bir başka tanım Polimorfizm, bir nesnenin kalıtımsal olarak ilişkisi olan diğer nesnelerin referanslarıyla işaretlenebilmesini sağlar.

Çok Biçimlilik(Polimorfizm) Nedir?

- Polimorfizm, OOP tasarımlarında geliştirilen koda daha manevrasal bir şekilde nitelik kazandıran ve yaklaşım sergilememizi sağlayan bir özelliktir.
- Polimorfizm, programlamada ki temel prensiplerden olan ‘Sol/Sağ’ prensibini aşıp, eldeki nesnenin birden fazla referans ile işaretlenebilmesini sağlar.

Çok Biçimlilik(Polimorfizm) Nedir?

- Bir nesnenin birden fazla referansla işaretlenmesi; o nesnenin, birden fazla türün davranışını gösterebilmesini sağlar.

```
class Test{}
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        Test t1 = new Test();
    }
}
```

Çok Biçimlilik(Polimorfizm) Nedir?

- Kuş deyince aklımıza kartal, papağan, deve kuşu, tavus kuşu vb. gelebilmektedir. Bunların hepsi temelde bir kuştur. Bununla birlikte kartal hem kuş özelliklerine hem de kartal özelliklerine sahip yani çok biçimlidir.
- Üstteki maddeye göre ortak atadan gelen, kalıtımsal olarak ‘Kuş’ tan türeyen tüm hayvanlar kendi türleri yahut ‘Kuş’ türü ile referans edilebilirler.
- Buradan da şunu anlıyoruz ki, yazılımsal açıdan çok biçimliliğin söz konusu olabilmesi için teknik olarak Kalıtım olması gerekmektedir!

Programlama da Polimorfizm Nerede Kullanılır?

- Bilinçsiz tür dönüşümü,
- Object türünün herhangi türdeki bir değeri alabilmesi,

Soyutlama(Abstraction)

- Soyutlama bir mantıktır, bir davranıştır.
- Tanım olarak soyutlama bir iş yaparken sadece ihtiyaç duyulan yapıların bulunması durumudur.
- Programlama açısından kullanıcı işlemleri için kullanılacak bir classta yüzlerce member olabilir, buna karşın sadece kullanıcı kayıt etmek isteyen bir kişiye bu metodun gösterilmesi yeterli olacaktır. İşte bu soyutlamadır.
- Bir başka deyişle soyutlama gerekli olanları göster, gereksiz olanları gösterme demektir.

Soyutlama(Abstraction)

- Soyutlama ile bir sınıfın memberlarından ihtiyaç noktasında alakalı olanları gösterip, alakalı olmayanları göstermemektir.

Abstract Class

- Abstract Class kullanılarak abstraction uygulanabilir.
- Abstraction = abstract class şeklinde bir düşünce yanlıştır. Bunlar aynı kelime kökenine sahip olmasına karşın abstraction interface ile de uygulanabilir. Burada dikkat edilmesi gereken abstraction davranışının abstract class ile uygulanabilecek olmasıdır.
- Bir sınıfın uyması gereken temel yapıyı tanımlamak için abstract class yapısı kullanılabilir ve gerekli modellemeyi gerçekleştirebilirsiniz.

Abstract Class

- Abstract classlardan normal şartlarda nesne üretilemez!