Masaüstü Uygulama Geliştirme

Hafta-11

Öğr. Gör. Zafer SERİN

Class Üyeleri(Memberları)

- Class üyeleri temel olarak 4 başlık altında incelenir. Bunlar:

1. Field(Alan)
2. Property(Özellik)
3. Method(Metot)
4. İndexer(İndeksleyici)

Field(Alan)

- Nesne içerisinde veri depoladığımız/tuttuğumuz alanlardır.
- Class içerisindeki değişkenlerdir.
- Herhangi bir türden olabilir.
- Field’lar türüne özgü varsayılan değer alırlar.

Property(Özellik)

- Nesne içerisinde özellik/property sağlar.
- Property esasında özünde bir metottur. Yani programatik/algoritmik kodlarımızı inşa ettiğimiz bir metot olarak ifade edilir.
- Lakin fiziksel olarak metottan farkı parametre almaması ve içerisinde get ve set olmak üzere iki adet blok içermesidir.
- Property'nin değeri okunmak istediğinde get bloğu Property'e değer atanmak istediğinde de set bloğu çalışır.
- Property'nin işlevsel açıdan metottan farkı yoktur, ancak davranışsal olarak nesne üzerinden değer okuma ve atama için kullanılır.

Neden Property(Özellik) Kullanılır?

- Yazılımcılar nesneler içerisindeki fieldlara doğrudan erişilmesini istemez!
- Dolayısıyla field'lar da ki verileri kontrollü bir şekilde dışarı açmak isteriz. İşte böyle bir durumda metotları kullanabiliriz. C#'ta buna Property ile de izin verilmiştir.
- Yani Property yapıları özünde nesne içerisindeki bir fieldın dışarıya kontrollü açılmasını ve kontrollü bir şekilde dışarıdan değer almasını sağlayan yapılardır.
- İşte Property'lerin bu işlevine Encapsulation(Kapsülleme/Sarmalama) denir ve nesne yönelimli programlamanın ortak özelliklerindendir.

Encapsulation(Kapsülleme/Sarmalama)

- Encapsulation, bir nesne içerisindeki dataların(fieldlardaki verilerin) dışarıya kontrollü bir şekilde açılması ve kontrollü bir şekilde veri almasıdır.

Property Tanımlama

[erişim belirleyicisi] [geri dönüş değeri] [property adı]
{
get {} // Property'den veri istendiğinde tetiklenir.
set{} /* Property'e veri gönderildiğinde tetiklenir. Gönderilen veriyi value keywordü yakalar.*/
}

- Eğer set bloğu tanımlanmaz ise sadece okunabilir(readonly), sadece get bloğu tanımlanmaz ise sadece yazılabilir(writeonly) property olacaktır.

Metot

- Nesne üzerinde, fieldlardaki yahut dışarıdan parametreler eşliğinde gelen değerler üzerinde işlemler yapmamızı sağlayan temel programatik parçalardır.

Indexer

- Nesneye indexer özelliği kazandıran onun dışında propety ile birebir aynı olan elemandır. Şu şekilde tanımlanır:

[erişim belirleyicisi] [geri dönüş değeri] this[parametreler]
{
get{} // Indexerdan veri istenildiğinde tetiklenir.
set{} /*Indexer'a veri gönderildiğinde tetiklenir. Gönderilen veriyi value keywordüyle yakalar*/
}

this Keywordü

- C#'ta this keywordü genel olarak 3 farklı amaç için kullanılır. Bunlar:

1. Sınıfın(Classın) nesnesini temsil etmek için kullanılır.
2. Aynı isimde field ile metot parametrelerini ayırmak için kullanılır.
3. Bir Constructor'dan(Yapıcı Metot) başka bir bir constructorı çağırmak için kullanılır.

Not: this keywordü class memberları(üyeleri) içerisinde kullanılır!

this Keywordü

```
class Ogrenci
{
    public int vize_notu { get; set; }
    public string isim { get; set; }

    public void HarfNotuHesapla(string isim, int vize_notu)
    {
        isim = "Zafer";
        this.vize_notu = 100;
    }
}
```

this Keywordü

- this keywordü ilgili class yapılanmasının o anki nesnesine karşılık geliyorsa önceden bahsedilen 3 durum dışında da this kullanmak gerekmez mi?
- C#'ta bahsedilen bu 3 durum dışında this keywordünü kullanma zorunluluğu yoktur. Bununla birlikte compiler(derleyici) seviyesinde yine this kullanılmaktadır. Python'da(self keywordü) veya typescriptte(this keywordü) bu keyword mutlaka yazılmalıdır!

Constructor(Yapıcı Metot)

- Constructor bir nesne üretimi sürecinde ilk tetiklenen(çalışan) metottur.
- Constructor, nesne oluşturma sürecinde tetiklenmek zorundadır!
- Nesne ilk oluşturulduğunda yapılmak istenen işlemler Constructor içerisinde belirtilir. Zorunlu olmamakla birlikte genel olarak class fiedlarına ve propertylerine başlangıç değerleri atamak için kullanılır. Nesne ile ilgili başlangıç konfigürasyonlarını(ayarlamalarını) yapmak için kullanılır.
- Constructor, new ile nesne oluşturma talebi geldikten ve ilgili nesneye hafızada yer ayrıldıktan sonra tetiklenir.

Constructor(Yapıcı Metot) Nasıl Oluşturulur?

- Constructor özel bir sınıf elemanıdır; ancak temelde bir metottur.
- Bir metot olmasına karşın imzası bir nebze farklıdır ve Constructorlar şu özelliklere sahiptir:

1. Constructor adı sınıfın adıyla aynı olmalıdır! (Özel sınıf elemanlarının dışında hiçbir class memberı(üyesi) sınıf adıyla aynı adda olamaz!)
2. Geri dönüş değeri olmaz/belirtilmez!
3. Erişim belirleyicisi public olmalıdır!(private olduğu durumu ayrıca inceleyeceğiz.)

Constructor(Yapıcı Metot) Nasıl Oluşturulur?

```
class Ogrenci
{
    public Ogrenci()
    {
        // Constructor metot içerisinde yapılacaklar
    }
    public void X()
    {
        // Normal metot içerisinde yapılacaklar. Bu constructor değildir!
    }
}
```

Default Constructor

- Her sınıfın içerisinde tanımlamasak dahi default bir constructor mevcuttur.
- Şimdiye kadar yaptığımızı örneklerde bir nesne oluştururken şu kodu yazdık: ‘Ogrenci ogrenci1 = new Ogrenci();’ buradaki () biz oluşturmasak dahi compiler(derleyici) seviyesinde varsayılan olarak bir constructor oluşturulduğunu belirtir.
- Eğer ki bir classa constructor eklersek bu default constructorı ezmiş(override) oluruz.

Parametreli Constructor ve Constructor Overloadı

- Constructorlar parametre alabilmektedirler.
- Constructorlar overload(aşırı yükleme) işlemine tabi tutulabilirler. Overload aynı isimde ki bir metodun farklı sayıda veya tipte parametreler ile çağrılarak birbirinden ayrılması ama aynı isimle çağrılmasıdır.

Private Constructor

- Bir constructorın erişim belirleyicisi private yapılırsa o class dışından nesne üretimi engellenmiş olur!
- Bu durum Singleton design pattern(tasarım deseni) yapısında çok sık kullanılır. Bu durumda ilgili constructor dışarıdan erişilemez lakin classın içerisinden erişilebileceği için ilgili işlemler burada yapılabilir.

this keywordüyle Constructorlar Arası Geçiş

```
class Ogrenci
{
    public Ogrenci() : this(42)
    {
        Console.WriteLine("Parametresiz yapıcı metot.");
    }
    public Ogrenci(int a)
    {
        Console.WriteLine("Tek parametreli yapıcı metot.");
    }
    public Ogrenci(int a, int b) : this()
    {
        Console.WriteLine("2 parametreli yapıcı metot.");
    }

}
```

Destructor(Yıkıcı Metot)

- Bir classtan üretilmiş olan nesne imha edilirken otomatik çağrılan metottur.
- Destructor özellikleri

1. Destructor sadece class yapılanmasında kullanılabilir.
2. Bir classta sade ve sadece bir Destructor bulunabilir.
3. Destructor parametre alamaz, erişim belirteci olmaz.
4. Destructor overload işlemlerine tabi tutulamaz.
5. Destructor class ile aynı isimde olmalı ancak başında ~(tilda) işareti olmalıdır.

Nesne Hangi Şartlarda Kim Tarafından İmha Edilir?

- Bir nesnenin imha edilmesi için;

1. İlgili nesne herhangi bir referans tarafından işaretlenmemelidir.
2. Nesnenin oluşturulduğu ve kullanıldığı scope(faaliyet alanı) sona ermiş olmalıdır.
3. Yani ilgili nesne bir daha erişilemez hale gelmelidir.

- Bu durumlarda nesne Garbage Collector(Çöp Toplayıcısı) tarafından imha edilir.

Garbage Collector(Çöp Toplayıcısı)

- Uygulamada lüzumsuz olan nesneleri toplamak için Garbage Collector isimli bir mekanizma devreye girer.
- Esasında Garbage Collector c#'ta bellek optimizasyonunu üstlenen bir yapılanmadır.
- C#'ta Garbage Collector'ın ne zaman iş göreceği tahmin edilemez.
- Dolayısıyla geliştiricilerin bu mekanizmaya müdahale etmesi pek önerilmez.

Destructor Tanımlama Kuralları

```
class Ogrenci
{
    ~Ogrenci()
    {
        // işlemler
    }
}
```

static Constructor

- static anahtar sözcüğü kullanıldığı yapıyı nesne üretilmeden, doğrudan sınıf üzerinden de erişilebilir yapmak için kullanılır.
- Örneğin bir metot static olarak işaretlenirse bu metodun tanımlandığı classtan bir nesne üretilmese dahi bu metoda class adı ile erişilebilir. Aynı durum class memberlarının tamamı için geçerlidir.
- Static Constructor bir classtan nesne üretildiğinde normal constructordan bile önce tetiklenen bir metottur. Normal constructordan farkı ise ilgili classtan 50 tane nesne oluşturulursa normal constructor 50 kez tetiklenirken static constructor yalnızca 1 kez tetiklenir.