BİL 201
NESNE YÖNELİMLİ PROGRAMLAMA (NYP)

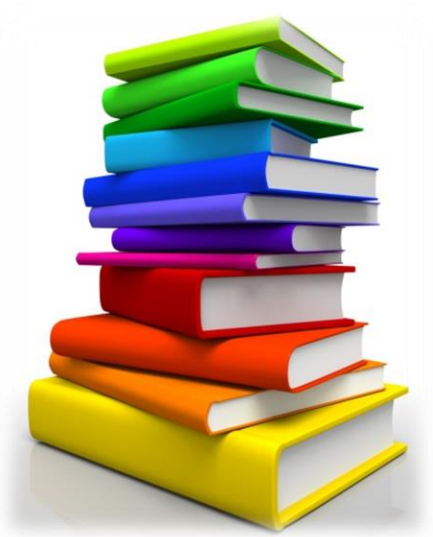
DERS #5

Öğretim Üyesi: Doç. Dr. Deniz Kılınç

BAKIRÇAY ÜNİVERSİTESİ

BÖLÜM 5 – Kurucular ve Örnekler

- Bu bölümde aşağıdaki konular anlatılacaktır
  - this referansı
  - Yapıcılar/Kurucular/Kurucu metotlar (Constructors)
  - Kurucu metotların aşırı yüklenmesi
  - Yıkıcılar (Destructors)



this Referansı

- Bir sınıf yaratıldığında, sınıfın kaynak kodunun bir **kopyası** **bilgisayarın hafızasında** (memory) saklanmaktadır.
- Bununla birlikte, er ya da geç bu sınıftan defalarca **nesne** oluşturulacaktır.
- Her yeni nesne oluşturduğunuzda, nesnenin her üye değişkeni için **bellekte saklama alanı** *sağlanır*.
- **this** referansı ile *sınıfın o anki nesnesine* ulaşabilirsiniz.
- this referansı kodda karışıklığı engellemek için de kullanılabilir.
- **Örneğin:** Sınıfın bir *üyesi* veya *özelliği* ile sınıfın metoduna aktarılan parametrenin aynı isimde olması durumunda karışıklığı önlemek için this referansı kullanılabilir.

Örnek 1: Kitap Sınıfında this Referansının Kullanılması

- Sizden **Kitap** isimli bir sınıf oluşturmanız istenmektedir. Bu sınıf decimal veri türünde, read-only ve public olan Fiyat ve VergiTutari özelliklerine sahiptir.
- Yine decimal türündeki VERGIORANI özelliği ise private olup aynı zamanda sabittir ve *değeri 0.18'dir*.
- Sınıfın VergiTutariHesapla() metodu bulunmakta ve bu metot Fiyat isimli değişkeni parametre olarak almaktadır.
- Metot ilk olarak; Fiyat özelliğine parametreyi atayacak daha sonra VergiTutarini hesaplayacaktır.

Kitap
+Fiyat: decimal {Read-Only} +VergiTutari: decimal {Read-Only} -VERGIORANI: decimal
+VergiTutariHesapla(decimal Fiyat)

Örnek 1: Kitap Sınıfında this Referansının Kullanılması

- Direk olarak Fiyat = Fiyat eşitliğini kullanırsa derleyici uyarı verir. Bunun gibi ve buna benzer ***karışıklıkları önlemek*** için this referansı kullanılır.
- `this.Fiyat = Fiyat` eşitliğinin sol tarafındaki fiyat değişkeni sınıfın özelliği olduğunu belirtmektedir. Sağındaki Fiyat değişkeni ise bu metoda aktarılan fiyat parametresidir.

```
public class Kitap
{
    private static decimal VERGIORANI = 0.18M;
    2 references
    public decimal Fiyat { get; private set; }
    1 reference
    public decimal VergiTutari { get; private set; }

    0 references
    public void VergiTutariHesapla(decimal Fiyat)
    {
        this.Fiyat = Fiyat;
        VergiTutari = this.Fiyat * VERGIORANI;
    }
}
```

Kurucular / Kurucu Metotlar / Yapıcılar (Constructors)

- Herhangi bir sınıftan bir nesne yarattığınızda aşağıdaki gibi bir ifade kullanılır:

```
Isci isc = new Isci();
```

- Aslında bu ifadede **Isci()** adında C#'ın sağladığı bir *metod çağırılmaktadır*.
- Kurucu metotlar (Constructor) bir nesnenin oluşmasını sağlayan metotlardır.
- Bir sınıf için **yazmamış olsanız bile**, yarattığınız sınıflar için *otomatik olarak* ***parametre almayan*** public bir kurucu metot sağlanır.

Kurucular (Constructors) (devam…)

- Parametresi olmayan kurucu metotlara sınıfın varsayılan kurucu metodu (default constructor) denir.
- Otomatik olarak yaratılan Isci() kurucu metodu, isc adında Isci sınıfından bir nesne oluşturur ve devamında Isci'nin ***başlangıç değerlerinin*** atanmasını sağlar.
- Sınıfın üye değişkenleri varsayılan olarak:
  - Sayısal fieldlar için 0 değeri,
  - Karakter fieldları için '\0' değeri,
  - Boolean field'lar için false değeri,
  - String ya da diğer referans tipli nesne fieldlarına NULL değeri atanmaktadır.

Kurucular (Constructors) (devam…)

- Isci nesnesinin üyelerinin ve özelliklerinin başlangıç değerlerinin varsayılan değerler olması **istenmiyorsa** veya
  - Isci sınıfından bir *nesne yaratılırken* ek işlemler yapmak istediğinizde varsayılan kurucu metot yerine **kendi kurucu metodunuzu** oluşturabilirsiniz.
- Sınıfın kurucu metotları **sınıf ile aynı isme** sahip olmak zorundadır.
- Kurucu metotların geri dönüş türleri **yoktur**. (int , float veya void olarak tanımlanmazlar)

Örnek 2: Isci Sınıfının Kurucu Metodunu Yaratmak

- Maas özelliğine sahip bir Isci sınıfı yaratınız.
- Bu sınıfın kurucu metodunu *yaratıp*, sınıftan yaratılan Isci nesnelerinin Maas özelliğinin varsayılan değerinin **1800.00** ve CalismaDurumu özelliğinin **Calisiyor** olması sağlayınız.
- Form üzerinde Isci sınıfından bir nesne yaratarak **Maas ve CalismaDurumu özelliklerinin** değerini ekranda gösteriniz. **DEBUG** işlemi gerçekleştiriniz.

ECalismaDurumu Değerleri
Emekli,
Calisiyor,
Ayrildi

Isci
+CalismaDurumu: ECalismaDurumu +Maas: decimal
<<Constructor>>+Isci()

Örnek 2: Isci Sınıfının Kurucu Metodunu Yaratmak (devam...)

```
public class Isci
{
    2 references
    public enum ECalismaDurumu
    {
        Emekli,
        Calisiyor,
        Ayrildi
    }
    2 references
    public decimal Maas { get; set; }
    2 references
    public ECalismaDurumu CalismaDurumu { get; set; }
    1 reference
    public Isci()
    {
        this.Maas = 1800;
        this.CalismaDurumu = ECalismaDurumu.Calisiyor;
    }
}
```

Kurucu Metotlara Parametre Geçirmek

- Kurucu metotlar, **parametre** de alabilirler.
- Aktarılan parametrelerin değerlerini kullanarak;
  - Yaratılan nesnenin özelliklerinin ya da üyelerinin değerlerini her bir nesne için ayarlamak mümkündür.

Örnek 3: Isci Sınıfının Kurucu Metoduna Parametre Geçirmek

- Örnek 2'deki Isci sınıfının kurucu metoduna geçirilen cocuk parametresine bağlı olarak AsgariGecimIndirimi ücretini aşağıdaki tabloya göre hesaplayınız. **DEBUG** işlemi gerçekleştiriniz.

Çocuk Sayısı	Asgari Geçim İndirimi
Çocuk Yok	80,33 ₺
1 ve Üzeri Çocuk	104,42 ₺

Isci
+CalismaDurumu: ECalismaDurumu +Maas: decimal +AsgariGecimIndirimi: decimal {Read-Only} +CocukSayisi: short {Read-Only}
<<Constructor>>+Isci(short cocuk) -AsgariGecimIndirimiHesapla()

Örnek 3: Isci Sınıfının Kurucu Metoduna Parametre Geçirmek (devam…)

```
public decimal Maas { get; set; }
2 references
public ECalismaDurumu CalismaDurumu { get; set; }
3 references
public decimal AsgariGecimIndirimi { get; private set; }
3 references
public short CocukSayisi { get; private set; }
1 reference
private void AsgariGecimIndirimiHesapla()
{
    if (this.CocukSayisi == 0)
        AsgariGecimIndirimi = 80.33M;
    else if (this.CocukSayisi >= 1)
        AsgariGecimIndirimi = 104.42M;
}
1 reference
public Isci(short cocukSayisi)
{
    this.Maas = 1800;
    this.CalismaDurumu = ECalismaDurumu.Calisiyor;
    this.CocukSayisi = cocukSayisi;
    AsgariGecimIndirimiHesapla();
}
```

```
private void BtnTest_Click(object sender, EventArgs e)
{
    Isci calisan = new Isci(1);
    MessageBox.Show("Çalışma durumu: " + calisan.CalismaDurumu +
                    "\nMaaş: " + calisan.Maas.ToString() +
                    "\nAsg.Geç.İnd: " + calisan.AsgariGecimIndirimi);
}
```

Kurucu Metotların Aşırı Yüklenmesi

- C#'da yaratılan sınıflar için otomatik olarak bir kurucu metot oluşturulmaktadır.
- Sınıf için yeni bir kurucu metot yarattığınızda C# 'ın **otomatik** olarak yarattığı kurucu metot erişilemez olur.
- Fakat otomatik yaratılan kurucu *metodun* **aynısı tanımlanabilir**.
- **Hatta:** bir sınıf için farklı parametrelerle istediğiniz kadar (anlam kargaşasına neden olmayacak şekilde) **kurucu metot** oluşturabilirsiniz.
- C# ın diğer metotları gibi, kurucu metotları da **aşırı yüklenebilir**.

Örnek 4: Dortgen Sınıfının Kurucu Metotlarının Aşırı Yüklenmesi

- UML diagramında görüldüğü gibi bir **Dortgen** sınıfı tanımlamanız istenmektedir.
- Dortgen sınıfının farklı parametreler alan iki adet kurucu metodu bulunmaktadır.

Dortgen
+Uzunluk: int {Read-Only} +Genislik: int {Read-Only} +Alan: int {Read-Only}
<<Constructor>>+Dortgen(int uzunluk, int genislik) <<Constructor>>+Dortgen(int tekuzunluk) +AlanHesapla()

Örnek 4: Dortgen Sınıfının Kurucu Metotlarının Aşırı Yüklenmesi

```
public class Dortgen
{
    3 references
    public int Uzunluk { get; private set; }
    3 references
    public int Genislik { get; private set; }
    1 reference
    public int Alan { get; private set; }
    1 reference
    public Dortgen(int uzunluk, int genislik)
    {
        this.Uzunluk = uzunluk;
        this.Genislik = genislik;
    }
    1 reference
    public Dortgen(int tekUzunuk)
    {
        this.Uzunluk = this.Genislik = tekUzunuk;
    }
    0 references
    public void AlanHesapla()
    {
        Alan = this.Uzunluk * this.Genislik;
    }
}
```

İki parametreli kurucu metot çalışır

```
private void BtnTest_Click(object send , EventArgs e)
{
    Dortgen dikdortgen = new Dortgen(4, 5);
    dikdortgen.AlanHesapla();
    MessageBox.Show(dikdortgen.Alan.ToString());

    Dortgen kare = new Dortgen(4);
    kare.AlanHesapla();
    MessageBox.Show(kare.Alan.T String());
}
```

Tek parametreli kurucu metot çalışır

Nesne Başlatıcıları (Object Initializers)

- Nesne başlatıcısı, bir sınıftan nesne yaratırken o nesnenin yaratılması sırasında erişilebilir üyelerine veya özelliklerine değer atanmasına izin verir.
- Örnek: Parametresiz bir kurucu metot içeren ve KimlikNumarasi adında public bir property içeren bir sınıfta, aşağıdaki ifadede olduğu gibi nesne başlatıcısı (object initializers) kullanılabilir:

```
Isci isc = new Isci { KimlikNo = 104 };
```

Nesne Başlatıcıları (Object Initializers) (dvam…)

- Örnekteki ifadede **104** değeri Isci sınıfından yaratılmış olan isc nesnesinin KimlikNo property'sine değer olarak atanmıştır.
- Değer atama küme parantezleri içerisinde yapılmıştır.
- Bu ifade çalıştırıldığında,
  - **İlk olarak** sınıfın **varsayılan kurucu metodu çalışır, sonra**
  - Nesne başlatıcısı ile KimlikNo property'sine **değer ataması** yapılır.

Örnek 5: Nesne Başlatıcısı Kullanma

- KimlikNo ve Maas özelliklerine sahip, Maas özelliğine 1500 ve KimlikNo özelliğine -1 başlangıç değerini atayan bir varsayılan kurucu metodu içeren bir Isci yaratınız.
- Öncelikle kurucu metottaki ifadelerin mi yoksa nesne başlatıcısındaki ifadelerin mi uygulandığını tespit ediniz.

Isci
+KimlikNo: int +Maas: decimal +Mesaj: string
<<Constructor>>+Isci()

Örnek 5: Nesne Başlatıcısı Kullanma (devam…)

```
public class Isci
{
    4 references
    public int KimlikNo { get; set; }
    4 references
    public decimal Maas { get; set; }
    2 references
    public string Mesaj { get; set; }
    1 reference
    public Isci()
    {
        this.KimlikNo = -1;
        this.Maas = 1500;
        this.Mesaj = "KimlikNo: " + this.KimlikNo +
                     "Maaş: " + this.Maas;
    }
}
```

```
private void BtnTest_Click(object sender, EventArgs e)
{
    Isci calisan = new Isci
    {
        KimlikNo = 222,
        Maas = 3000
    };
    MessageBox.Show(calisan.Mesaj);
    MessageBox.Show("KimlikNo: " + calisan.KimlikNo +
                    "Maaş: " + calisan.Maas);
}
```

Örnek 5: Nesne Başlatıcısı Kullanma (devam…)

```
Isci calisan = new Isci {KimlikNo = 222, Maas = 3000};
```

- Bu ifade aşağıdaki birer birer değer ataması yapılan ifadeler ile aynı işlemi yapmaktadır:

```
Isci calisan = new Isci();
calisan.KimlikNo = 222;
calisan.Maas = 3000;
```

Yıkıcı Metotlar

- Yıkıcı metot, o sınıftan yaratılmış bir nesne yok edildiğinde *yapılacak olan eylemleri içeren* metotlara denir.
- Genellikle sınıftan yaratılmış olan nesne **kapsam dışında kaldığında** *yok edilmektedir*.
- Kurucu metotlarda olduğu gibi, sınıf için bir yıkıcı metot tanımlanmadığında C# sizin için otomatik olarak bir yıkıcı metot sağlamaktadır.

Yıkıcı Metotlar (devam...)

- Yıkıcı metot tanımlamak için, “ **~** ” (Tilda) işareti ile sınıfın adı olarak tanımlanır.

```
~ [sınıfınAdı] ()
{
}
```

- Yıkıcı metotlara herhangi bir parametre **geçirilemez**.
- Yıkıcı metotlar aşırı yüklenemezler(overload).
- Geri dönüş değerine **sahip olmazlar**.

Örnek 6: Isci Sınıfında Yıkıcı Metotların Kullanımı

- KimlikNo property'sine sahip olan Isci sınıfını yıkıcı metodunu oluşturunuz.
- Oluşturduğunuz Isci sınıfından farklı kimlik numaralarına sahip **iki adet** nesne oluşturup çalıştırınız.

Isci
+KimlikNo: int
<<Constructor>>+Isci(int kimlikno)

Örnek 6: Isci Sınıfında Yıkıcı Metotların Kullanımı

```
public class Isci
{
    3 references
    public int KimlikNo { get; set; }
    1 reference
    public Isci(int kimlikNo)
    {
        this.KimlikNo = kimlikNo;
        Console.WriteLine(this.KimlikNo + " kimlik numaralı nesne yaratıldı");
    }
    0 references
    ~Isci()
    {
        Console.WriteLine(this.KimlikNo + " kimlik numaralı nesne yok edildi");
    }
}
```

Örnek 6: Isci Sınıfında Yıkıcı Metotların Kullanımı

```
private void BtnTest_Click(object sender, EventArgs e)
{
    Isci calisan = new Isci(2);
    calisan = null;
    GC.Collect();
}
```

```
'Orn6-Destruc.exe' (CLR v4.0.30319: Orn6-Destruc.exe): Loaded 'C:\Windows\Mi
'Orn6-Destruc.exe' (CLR v4.0.30319: Orn6-Destruc.exe): Loaded 'C:\Windows\Mi
2 kimlik numaralı nesne yaratıldı
2 kimlik numaralı nesne yok edildi
'Orn6-Destruc.exe' (CLR v4.0.30319: Orn6-Destruc.exe): Loaded 'C:\Windows\Mi
The program '[6404] Orn6-Destruc.exe' has exited with code 0 (0x0).
```

Garbage Collector

- GC otomatik hafıza yönetimi ile uygulama yazımını kolaylaştırmakta, kodlama süresini kısaltmaktadır.
- Yine GC, makine koduna derlenen dillere göre kodlama hatalarından dolayı karşılaşılan hafıza alanlarının leak edilmesi problemi ile de bir seviyeye kadar baş edebilmektedir.

https://medium.com/@gokhansengun/garbage-collector-nas%C4%B1l-%C3%A7al%C4%B1%C5%9F%C4%B1r-3bdf2fb20282

Yıkıcı Metotlar (devam…)

- Yıkıcı metotlar **çağırılmaya gerek kalmadan** otomatik olarak uygulanırlar.
- En son yaratılan nesne ilk olarak yok edilir.
- Sınıftan yaratılmış bir nesne, kodda onunla yapılacak bir işi olmadığında ***yok edilmeye uygun hale gelir***.
- Yıkıcı metotlar program kapatılırken yapmak istediklerimizi (bağlantı kapatma, geçici dosya silme vb.) yapmamızı sağlarlar.

Yararlanılan Kaynaklar

- Sefer Algan , HER YÖNÜYLE C# , Pusula Yayıncılık, İstanbul, 2003
- Volkan Aktaş, HER YÖNÜYLE C# 5.0 , Kodlab Yayıncılık, İstanbul, 2013
- Milli Eğitim Bakanlığı "Nesne Tabanlı Programlama", 2012

İyi Çalışmalar...

Doç. Dr. Deniz Kılınç

deniz.kilinc@bakircay.edu.tr

drdenizkilinc@gmail.com

www.denizkilinc.com