Nesne Tabanlı Programlama
Bölüm 3
Kurucu ve Yok Edici Fonksiyonlar

Dr. Öğr. Üyesi Murat TAŞYÜREK (kayubmprogramlama1@gmail.com)

16 Ekim 2023

Kayseri Üniversitesi, Bilgisayar Mühendisliği Bölümü

Kurucu Fonksiyonlar (Constructors)

- Kurucu fonksiyonlar (Constructors) üyesi olduğu sınıftan nesne üretilirken otomatik olarak tetiklenir (canlanır).
- Kurucu fonksiyonlar nesnelerin oluşturularken başlangıç değeri atamak için kullanılır.
- Kurucu fonskiyonlar üyesi oldukları sınıf ile aynı ismi taşır.
- Kurucu fonksiyonlar parametre alabilirler ancak geri dönüş değeri yoktur. Diğer bir ifade ile void yazılmaz veya int string gibi geriye değer döndürmez.
- Kurucu fonksiyonlar nesne oluşturularken dışardan erişildiğinden dolayı public olarak tanımlanmalıdır.

Örnek 1

- Bir kişi (insan) türünde bir nesne oluşturmak istiyoruz.
- Bunun için öncelikle bir sınıf oluşturunuz.
- İnsan çocuk olarak dünya gelir.
- Çocuk olarak dünyaya geldiğinde soyadı bellidir. Babasının soyadını alır.
- Soy ismi değiştirilemez başlangıç değerinde atanır.
- Kodlayalım

Console App Person Class

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        Person p = new Person("taşyürek");
        p.ShowSurname();
    }
}
3 references
public class Person
{
    string surname;
    1 reference
    public Person(string fatherSurname)
    {
        this.surname = fatherSurname;
    }
    1 reference
    public void ShowSurname()
    {
        Console.WriteLine("Kişinin soyadı:" + this.surname);
    }
}
```

nesne oluştururken dışardan erişildiğinden dolayı public olarak tanımlanması gerekiyor.

Çıktı

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        Person p = new Person("taşyürek");
        p.ShowSurname();
    }
}
```

```
C:\WINDOWS\system32\cmd.exe
Kişinin soyadı:taşyürek
Press any key to continue . . .
```

Örnek 2

- Kurucu metodlar birden fazla parametre alabilir.
- Bir sınıfın birden fazla kurucu metodu olabilir.
- Nesne oluşturulurken gönderilen veri türüne ve sayına göre otomatik olarak ilgili kurucu metot çağrılır.
- Bir önceki örnekteki bilgilere ilaveten insanın ismi de olur.
- İnsan dünyaya geldiğinde ismi direk soyismi ile konulabilir veya ismi daha sonra konulabilir.
- Kodlayalım

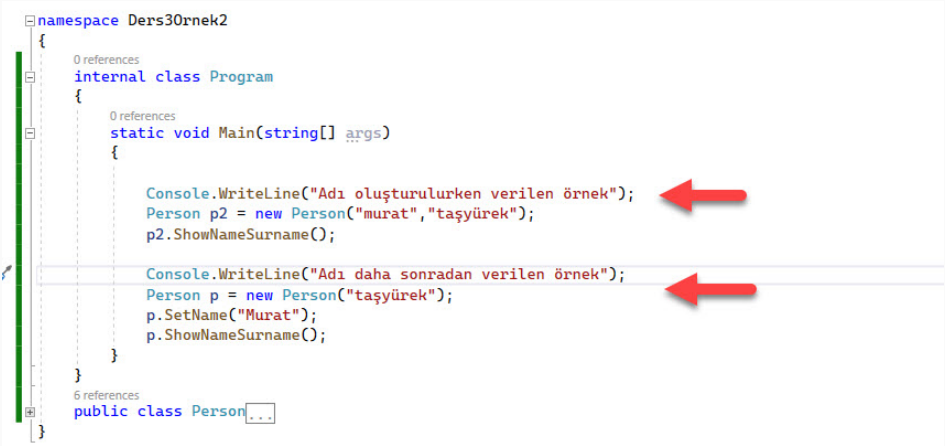
Console App Person Class

```
public class Person
{
    string name,surname;
    1 reference
    public Person(string fatherSurname)
    {
        this.surname = fatherSurname;
    }
    1 reference
    public Person(string name, string fatherSurname)
    {
        this.name = name;
        this.surname = fatherSurname;
    }
    1 reference
    public void SetName(string name)
    {
        this.name = name;
    }
    2 references
    public void ShowNameSurname()
    {
        Console.WriteLine("Kişinin adı:" + this.name);
        Console.WriteLine("Kişinin soyadı:" + this.surname);
    }
}
```

Kurucu metot

Sonradan isim verme için kullanılan metot

Kaynak Kod



```
namespace Ders3Ornek2
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {

            Console.WriteLine("Adı oluşturulurken verilen örnek");
            Person p2 = new Person("murat","taşyürek");
            p2.ShowNameSurname();

            Console.WriteLine("Adı daha sonradan verilen örnek");
            Person p = new Person("taşyürek");
            p.SetName("Murat");
            p.ShowNameSurname();
        }
    }
    6 references
    public class Person...
}
```

Çıktı

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {

        Console.WriteLine("Adı oluşturulurken verilen örnek");
        Person p2 = new Person("murat","taşyürek");
        p2.ShowNameSurname();

        Console.WriteLine("Adı daha sonradan verilen örnek");
        Person p = new Person("taşyürek");
        p.SetName("Murat");
        p.ShowNameSurname();
    }
}
```

```
C:\WINDOWS\system32\cmd.exe
Adı oluşturulurken verilen örnek
Kişinin adı:murat
Kişinin soyadı:taşyürek
Adı daha sonradan verilen örnek
Kişinin adı:Murat
Kişinin soyadı:taşyürek
Press any key to continue . . .
```

Örnek 3

- C# programlama dilinde kurucu metoda benzeyen nesne oluşturma yöntemi mevcuttur.
- Eğer classın özellikleri public ise yani dışardan erişime açık ise nesneyi oluştururken süslü parantez açıp kapatararak $\{\}$ nesnenin özellikleri direk oluştururken belirtebilirsiniz.
- Nesne oluşturulurken nesnenin özellikleri set edildiğiden dolayı kurucu yönteme benzer.
- Kod üzerinde gösterim.

Kaynak Kod

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {

        Console.WriteLine("Örnek 1");
        Person p1 = new Person { name = "murat" };
        p1.ShowNameSurname();

        Console.WriteLine("\nÖrnek 2");
        Person p2 = new Person { surname = "taşyürek" };
        p2.ShowNameSurname();

        Console.WriteLine("\nÖrnek 3");
        Person p3 = new Person { name = "murat", surname = "taşyürek" };
        p3.ShowNameSurname();
    }
}
6 references
public class Person
{
    public string name, surname;

    3 references
    public void ShowNameSurname()
    {
        Console.WriteLine("Kişinin adı:" + this.name);
        Console.WriteLine("Kişinin soyadı:" + this.surname);
    }
}
```

Çıktı

```
namespace Ders3Ornek3
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {

            Console.WriteLine("Örnek 1");
            Person p1 = new Person { name = "murat" };
            p1.ShowNameSurname();

            Console.WriteLine("\nÖrnek 2");
            Person p2 = new Person { surname = "taşyürek" };
            p2.ShowNameSurname();

            Console.WriteLine("\nÖrnek 3");
            Person p3 = new Person { name = "murat", surname = "taşyürek" };
            p3.ShowNameSurname();
        }
    }
    6 referenc
    public
    {
```

```
C:\WINDOWS\system32\cmd.exe
Örnek 1
Kişinin adı:murat
Kişinin soyadı:

Örnek 2
Kişinin adı:
Kişinin soyadı:taşyürek

Örnek 3
Kişinin adı:murat
Kişinin soyadı:taşyürek
Press any key to continue . . .
```

Sınıf Statik fonksiyonları

- Programlama dillerinde sabit yapılan işlemlerde statik fonksiyonlar kullanılır.
- Statik fonksiyonlarda nesne üretmeye gerek yoktur.
- {Sınıf ismi}.{Metod adi} ile çağrılır.
- Statik fonskiyonlar sadece program derlenirken (exe oluşturulken) hafıza da yer ayrılır. Daha sonra çağrılan bütün metotlar aynı yeri kullanır.
- Statik olduğundan ve aynı kod kullanılacağından tekrar tekrar hafızada yer ayırmaya gerek yoktur.
- İşlemler sabit olduğunda, başka bir işlem veya sonuçlardan etkilenmiyor veya etkilemiyor ise bu durumlarda statik fonskiyon kullanılır.

Örnek 4

- Matematik işlemleri için bir sınıf oluşturduğumuzu varsayalım.
- Girilen bir değerin faktöriyeli alan ve geriye değer dönderen fonskiyonu statik ve normal olarak yazalım.
- Sınıf içerisinde metotları çağırlarım.

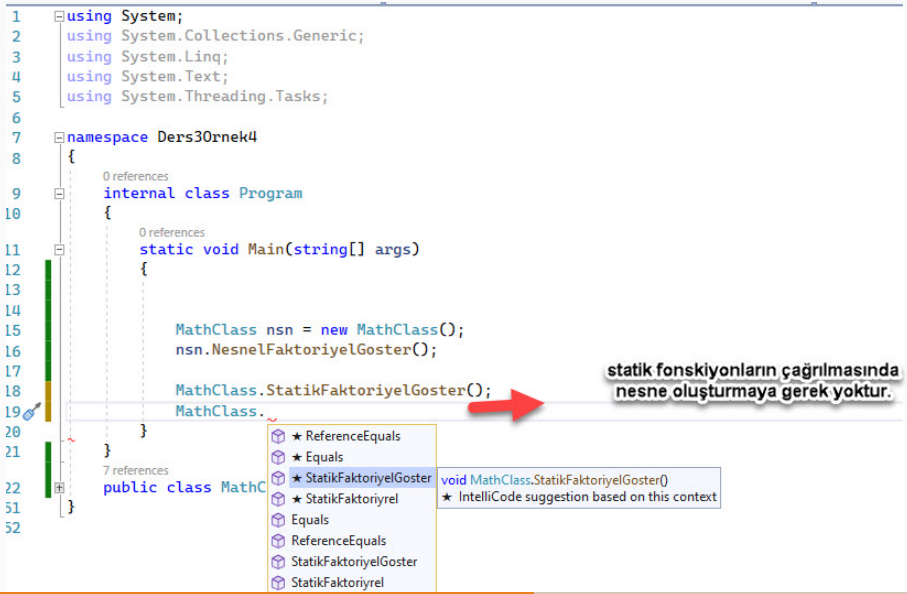
Math Class

```
public class MathClass
{
    1 reference
    public static int StatikFaktoriyrel(int deger)
    {
        int carpim = 1;
        for (var i =1; i <= deger; i++)
            carpim=carpim*i;
        return carpim;
    }
    2 references
    public int NesnelFaktoriyrel(int deger)
    {
        int carpim = 1;
        for (var i = 1; i <= deger; i++)
            carpim = carpim * i;
        return carpim;
    }
    1 reference
    public static void StatikFaktoriyelGoster()
    {
        int deger = MathClass.StatikFaktoriyrel(6);
        Console.WriteLine("6 değerinin faktoriyeli="+deger);
    }
    1 reference
    public void NesnelFaktoriyelGoster()
    {
        int deger = this.NesnelFaktoriyrel(5);
        MathClass cls = new MathClass();
        int deger2 = cls.NesnelFaktoriyrel(5);
        Console.WriteLine("5 değerinin faktoriyeli=" + deger+", nesne üretilen değer:"+deger2);
    }
}
```

fonkstiyon ve procedüre statik olabilir.

Sınıf içerisinde this veya sınıftan nesne üretilerek nesnenin metodu çağıralbilir.

Kaynak Kod


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Ders3Ornek4
{
0 references
internal class Program
{
0 references
static void Main(string[] args)
{
MathClass nsn = new MathClass();
nsn.NesnelFaktoriyelGoster();
MathClass.StatikFaktoriyelGoster();
MathClass.
}
}
7 references
public class MathC
}
★ ReferenceEquals
★ Equals
★ StatikFaktoriyelGoster
★ StatikFaktoriyrel
Equals
ReferenceEquals
StatikFaktoriyelGoster
StatikFaktoriyrel
void MathClass.StatikFaktoriyelGoster()
★ IntelliCode suggestion based on this context
statik fonskiyonların çağrılmasında
nesne oluşturmaya gerek yoktur.

Çıktı

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {


        MathClass nsn = new MathClass();
        nsn.NesnelFaktoriyelGoster();

        MathClass.StatikFaktoriyelGoster();
    }
}
6 refe
pub
```

```
C:\WINDOWS\system32\cmd.exe
5 değerinin faktoriyeli=120, nesne üretilen değer:120
6 değerinin faktoriyeli=720
Press any key to continue . . .
```

Sınıf Statik Değişkenler

- A static variable is declared with the help of static keyword. When a variable is declared as static, then a single copy of the variable is created and shared among all objects at the class level. Static variables are accessed with the name of the class, they do not require any object for access (c# static variable in class).
- Statik değişkenlerde fonksiyonlar gibi sadece 1 defa hazıda yer ayrılır.
- Yapılan bütün değişiklikler aynı yerde yapıldığından dolayı sınıftan oluşturulan bütün nesnelerde aynı değerler okunur.
- Yani bir değişkeni statik tanımladınız ve o sınıftan türetilen nesnelerden eriştiğinizde aslında ortak değişken kullanılır.

Örnek 5

- Toplama işlemi için bir sınıf içerisinde statik değişken tanımlayalım.
- Sınıf içerisinde bu değişkenin değeri artıran bir procedüre olsun.
- Sınıf içerisinde bu değişkenin değerini dönderen bir de fonskiyon olsun.
- Kodlayalım.

Kaynak Kod

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        StaticDegiskenSinifi snf = new StaticDegiskenSinifi();
        Console.WriteLine("Statik değişkenin değeri:" + snf.getNum());
        snf.count();
        snf.count();
        snf.count();
        Console.WriteLine("Statik değişkenin değeri:" + snf.getNum());

        StaticDegiskenSinifi snf2 = new StaticDegiskenSinifi();
        Console.WriteLine("Yeni oluşturulan sınıfın statik değişkenin değeri:" + snf2.getNum());
    }
}
4 references
class StaticDegiskenSinifi
{
    public static int num;

    3 references
    public void count()
    {
        num++;
    }
    3 references
    public int getNum()
    {
        return num;
    }
}
```

Çıktı

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        StaticDegiskenSinifi snf = new StaticDegiskenSinifi();
        Console.WriteLine("Statik değişkenin değeri:" + snf.getNum());
        snf.count();
        snf.count();
        snf.count();
        Console.WriteLine("Statik değişkenin değeri:" + snf.getNum());

        StaticDegiskenSinifi snf2 = new StaticDegiskenSinifi();
        Console.WriteLine("Yeni oluşturulan sınıfın statik değişkenin değeri:" + snf2.getNum());
    }
}
4 references
class S
{
    pub
```

```
C:\WINDOWS\system32\cmd.exe
Statik değişkenin değeri:0
Statik değişkenin değeri:3
Yeni oluşturulan sınıfın statik değişkenin değeri:3
Press any key to continue . . .
```

Yok edici fonksiyonlar

- Destructors in C# are methods inside the class used to destroy instances of that class when they are no longer needed. The Destructor is called implicitly by the .NET Framework's Garbage collector and therefore programmer has no control as when to invoke the destructor. An instance variable or an object is eligible for destruction when it is no longer reachable.
- Bir nesnenin kullanımı bittiği anda nesnenin yok edilmesi ve gerekli hafızanın boşaltılması gerekmektedir.
- .NET Framework's Garbage collector bu işlemi otomatik olarak yapmaktadır.
- Nesnenin kullanımı bittiği anda yok edici fonksiyon otomatik olarak tetiklenmektedir (destructors).
- Bazı programlama dillerinde bu fonksiyon çağrılması gerekebilir.

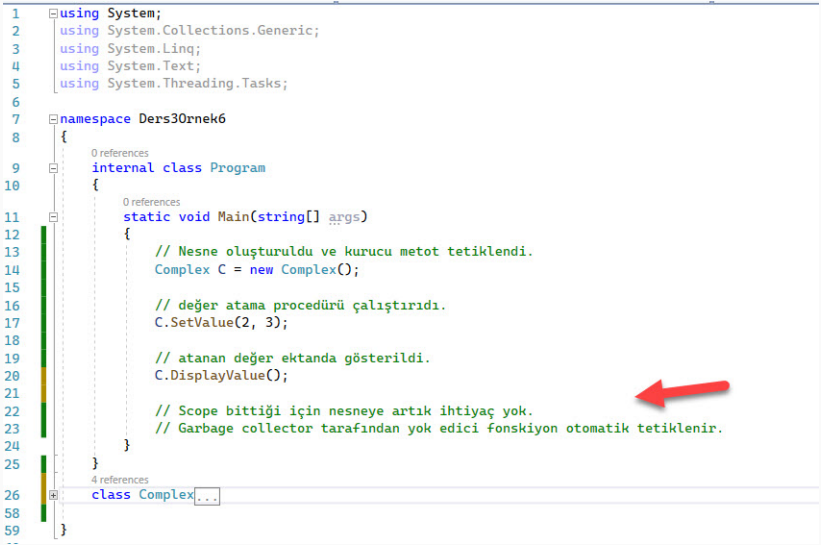
Örnek 6

- İki adet integer değerin gizli olarak tutulduğu Complex isminde bir sınıf tanımlayalım.
- Bu sınıfın içerisine değişkenlere değer atama ve değerleri gösterme için de iki adet procedüre olsun.
- Nesne yok edilirken nesnenin yok edildiğini yakalayalım ve ekrana yok edildiğini yazdıralım.
- Kodlayalım.

Complex Class

```
4 references
class Complex
{
    //  sınıf üyeleri default olarak privatedır.
    int real, img;

    // kurucu fonksiyon
    1 reference
    public Complex()
    {
        real = 0;
        img = 0;
    }

    // değer atama procedürü
    1 reference
    public void SetValue(int r, int i)
    {
        real = r;
        img = i;
    }

    // değerleri gösterme
    1 reference
    public void DisplayValue()
    {
        Console.WriteLine("Real = " + real);
        Console.WriteLine("Imaginary = " + img);
    }

    // yok edici fonskiyon
    0 references
    ~Complex()
    {
        Console.WriteLine("Yok edici fonskiyon tetiklendi");
    }
}
```

Kaynak Kod



```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace Ders3Ornek6
8   {
        0 references
9       internal class Program
10      {
            0 references
11          static void Main(string[] args)
12          {
13              // Nesne oluşturuldu ve kurucu metot tetiklendi.
14              Complex C = new Complex();
15
16              // değer atama procedürü çalıştırıdı.
17              C.SetValue(2, 3);
18
19              // atanan değer ektanda gösterildi.
20              C.DisplayValue();
21
22              // Scope bittiği için nesneye artık ihtiyaç yok.
23              // Garbage collector tarafından yok edici fonskiyon otomatik tetiklenir.
24          }
25      }
            4 references
26      class Complex...
58
59  }
```

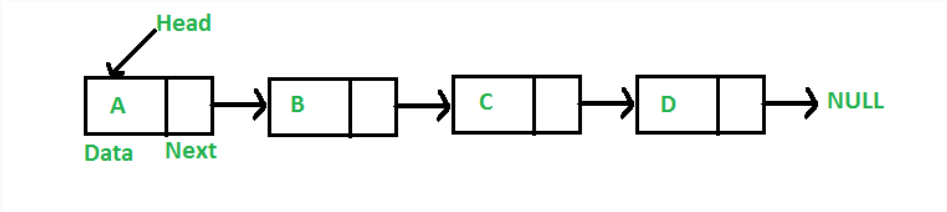
Çıktı

```
0 references
internal class Program
{
    0 references
    static void Main(string[] args)
    {
        // Nesne oluşturuldu ve kurucu metot tetiklendi.
        Complex C = new Complex();

        // değer atama procedürü çalıştırıdı.
        C.SetValue(2, 3);

        // atanan değer ektanda gösterildi.
        C.DisplayValue();

        // Scope bittiği için nesneye artık ihtiyaç yok.
        // Garbage collector tarafından yok edici fonskiyon otomatik tetikler
    }
}
4 referen
class
```

```
C:\WINDOWS\system32\cmd.exe
Real = 2
Imaginary = 3
Yok edici fonskiyon tetiklendi
Press any key to continue . . .
```

Bilgi

- Bağlantılı listede listenin başındaki öğeden diğer öğeye ulaşırsınız. Aradaki öğelere direk erişim yoktur.
- Her öğe kendiyle ilgili özel değerleri ve kendinden sonraki öğenin adresini bilir.
- Son öğenin adresi boştur buradan listenin bittiğini anlarsınız.



C# Bağlantılı liste gösterimi

Ödev - Altın Bulma Oyunu

- Bir belediye otobüsü günlük sefer için yola çaıkmıştır.
- Otobüsünün kapasitesi 40 kişidir.
- Bilet basma cihazı olan validatör giriş kapısında bulunmaktadır.
- 38 adet yolcu 15 ön kapı, 10 orta kapı ve 13 arka kapı olmak üzere otobüse binmiştir. Bir yolcu otobüse binerken boş koltukları kontrol ederek gider ruh haline göre beğendiği yere oturur.
- Ön kapıdan binen yolcuların tamamı bilet basmıs, orta kapıdan binen yolcuların yüzde %50'si bilet bastmış, arka kapıdan binen yolculardan ise 9 tanesi bilet basmamıştır.
- Bağlantılı liste kullanarak hangi koltukta kimin orduğunu (adı, soyadı, yaşı, cinsiyeti vb.), hangi kapıdan biniş yaptığını ve bilet basıp basmadığını gösteriniz.
- 23 Ekim 06.59'a kadar gönderilenler 8 $+2$ puan, vize tarihine kadar gönderenler 8 puan (kayubmprogramlama1@gmail.com).