

6. 1. Sınıf Üyeleri

1. Alanlar (field)

string, int, double vb. olabileceği gibi sınıflar da bir başka sınıfın içerisinde kullanıldıklarında alan olarak kabul edilir.

2. Metodlar

Uygulamalar içerisindeki belirli işleri yapan küçük programlardır. Veriler üzerindeki her türlü operasyonlarda bu yapı içerisinde yapılır. Her ne kadar yapısal programlama kavramı olsa da NDP bu yapılardan vazgeçmemiş hatta sınıfı oluşturan yapı taşlarından kabul etmiştir. Kodun merkezi yönetimi, kodun tekrar kullanılabilirliği ve soyutlama sağlar.

Genel yapısı

erişim belirteci geridonustipi metod_ismi(parametre listesi)

şeklindedir.

Örnek 1: 2 tam sayıyı toplayıp sonucu geri döndüren metod.

```
public class A
{
    public int Hesapla(int a, int b)
    {
        int toplam; toplam = a + b;
        return toplam;
    }
}
class Program
{
    static void Main(string[] args)
    {
        A yeniSinif = new A();
        int sonuc = yeniSinif.Hesapla(8, 9);
        Console.WriteLine(sonuc);
    }
}
```

Örnek 2: Ekrana bilgi yazdıran metot

```
public class A
{
    private int Hesapla(int a, int b) // sarmala gereği erişim düzeyi
private olarak değiştirilmiştir.
    {
        int toplam;
        toplam = a + b;
        return toplam;
    }
    public void EkranaYazdir()
    {
        Console.WriteLine(Hesapla(5, 6));
    }
}
class Program
{
    static void Main(string[] args)
    {
        A yeniSinif = new A();
        yeniSinif.EkranaYazdir();
    }
}
```

Dışarıdan değer almayan ve geri değer döndürmeyen metot tanımı:

erişim_belirteci **void** metot_ismi()

Dışarıdan değer alan ve geri değer döndürmeyen metot tanımı:

erişim_belirteci **void** metot_ismi(parametre listesi)

Dışarıdan değer almayan ve geri değer döndüren metot tanımı:

erişim_belirteci **geriDonusTipi** metot_ismi()

Dışarıdan değer alan ve geri değer döndüren metot tanımı:

erişim_belirteci **geriDonusTipi** metot_ismi(parametre listesi)

Metotların geri değer döndürmesi durumunda **return** anahtar kelimesi ile metodun işlevi geri döndürülür.

Örnek 3: Öğrencilerin ad, soyad, doğum tarihi, memleketi, öğrenci numarası, bölümü, fakültesi bilgilerini tutan sınıfı oluşturalım.

```
class Ogrenci
{
    public string adi; public string soyadi;
    public DateTime dogumTarihi; public string memleket; public string
numara; public string bolum;
    public string fakulte;
    public void EkranaYazdir()
    {
        Console.WriteLine("Adı          : " + adi);
        Console.WriteLine("Soyadı       : " + soyadi);
        Console.WriteLine("Doğum Tarihi : " +
dogumTarihi.ToShortDateString());
        Console.WriteLine("Memleket     : " + memleket);
        Console.WriteLine("Numarası     : " + numara);
        Console.WriteLine("Fakültesi    : " + fakulte);
        Console.WriteLine("Bölümü      : " + bolum);
    }
}

class Program
{
    static void Main(string[] args)
    {
        // o1 isimli ilk öğrenci nesnesi oluşturuluyor
        Ogrenci o1 = new Ogrenci();

        //o1 nesnesine değerleri atanıyor
        o1.adi = "Barış"; o1.soyadi = "Çalışkan";
        o1.dogumTarihi = new DateTime(1981, 3, 1, 7, 0, 0); o1.memleket =
"Sakarya"; o1.numara = "0001.10001";
        o1.bolum = "Bilgisayar Mühendisliği"; o1.fakulte = "Mühendislik";

        //o1 nesnesi ekrana yazdırılıyor
        o1.EkranaYazdir();
    }
}
```

```
C:\Windows\system32\cmd.exe
Adı      : Barış
Soyadı   : Çalışkan
Doğum Tarihi : 01.03.1981
Memleket : Sakarya
Numarası : 0001.10001
Fakültesi : Mühendislik
Bölümü   : Bilgisayar Mühendisliği
Devam etmek için bir tuşa basın . . .
```

Programımıza 2. öğrenciyi ekleyelim.

```
// o2 isimli ikinci öğrenci nesnesi
oluşturuluyor
Ogrenci o2 = new Ogrenci();

//o2 nesnesine değerleri
atanıyor
o2.adi = "Ahmet";
o2.soyadi = "Şanslı";
o2.dogumTarihi =new DateTime(1983,7, 19, 7, 0, 0);

o2.memleket = "Sakarya";
o2.numara = "0101.10001";
o2.bolum = "Bilgisayar Mühendisliği";
o2.fakulte = "Mühendislik";

//o2 nesnesi ekrana yazdırılıyor
o2.EkranaYazdir();
```

```
C:\Windows\system32\cmd.exe
Adı      : Barış
Soyadı   : Çalışkan
Doğum Tarihi : 01.03.1981
Memleket   : Sakarya
Numarası   : 0001.10001
Fakültesi  : Mühendislik
Bölümü     : Bilgisayar Mühendisliği
Adı        : Ahmet
Soyadı     : Şanslı
Doğum Tarihi : 19.07.1983
Memleket   : Sakarya
Numarası   : 0101.10001
Fakültesi  : Mühendislik
Bölümü     : Bilgisayar Mühendisliği
Devam etmek için bir tuşa basın . . .
```

Ogrenci sınıfının içerisindeki **EkranaYazdir()** metodunda yapacağımız değişiklikler bu sınıftan oluşturulmuş tüm nesneleri etkileyecektir.

Ogrenci sınıfını aşağıdaki şekilde değiştirelim.

```
class Ogrenci
{
    public string adi;
    public string soyadi;
    public DateTime dogumTarihi;
    public string memleket;
    public string numara;
    public string bolum;
    public string fakulte;
    public void EkranaYazdir()
    {
        Console.WriteLine("Numarası      : " + numara);
        Console.WriteLine("-----");
        Console.WriteLine("Adı      : " + adi);
        Console.WriteLine("Soyadı   : " + soyadi);
        Console.WriteLine("Doğum Tarihi : " +
dogumTarihi.ToShortDateString());
        Console.WriteLine("Memleket   : " + memleket);
        Console.WriteLine("Fakültesi  : " + fakulte);
        Console.WriteLine("Bölümü     : " + bolum);
        Console.WriteLine();
        Console.WriteLine("*****");
        Console.WriteLine();
    }
}
```



```
C:\Windows\system32\cmd.exe

Numarası      : 0001.10001
-----
Adı      : Barış
Soyadı   : Çalışkan
Doğum Tarihi : 01.03.1981
Memleket   : Sakarya
Fakültesi  : Mühendislik
Bölümü     : Bilgisayar Mühendisliği

*****

Numarası      : 0101.10001
-----
Adı      : Ahmet
Soyadı   : Şanlı
Doğum Tarihi : 19.07.1983
Memleket   : Sakarya
Fakültesi  : Mühendislik
Bölümü     : Bilgisayar Mühendisliği

*****
```

6.2 Metodların aşırı yüklenmesi (Overload)

Aynı isme sahip, dışardan değer alma sayıları ya da tipleri farklı olan metotlardır. Aşırı yüklenmiş metotlar kullanıldıkları sınıflarda, bu sınıflara ait nesne örnekleri için aynı isme sahip fakat farklı görevleri yerine getirebilen (benzer görevi farklı sayı veya tipte parametre ile yerine getirebilen) fonksiyonellikler kazandırır. Burada dikkat edilmesi gereken başlıca husus geri dönüş tipinin değişmesi ile aşırı yüklenmiş metot oluşturulamamasıdır.

Örnek: `integer`, `double` ve `string` değerler ile toplama yapan metot.

```
class Toplama
{
    public int Sonuc(int a)
    {
        return a + a;
    }
    public double Sonuc(double a, double b)
    {
        return a + b;
    }
    public string Sonuc(string a)
    {
        return a;
    }
}
```

Sonuc isimli metot 3 defa kullanılmış ancak her birinde dışardan aldığı değerlerin tipleri ve sayıları değişmiştir. Main içerisinde t isimli nesne ekrana yazdırılırken oluşan görünüm bize bu metodun 3 farklı şekilde kullanılabileceğini söylemektedir.

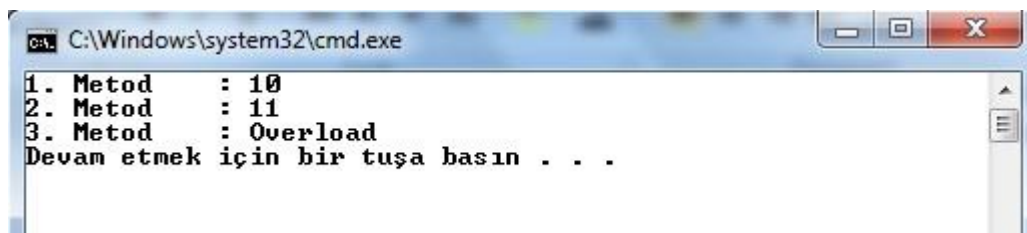
```
class Program
{
    static void Main(string[] args)
    {
        Toplama t = new Toplama();
        Console.WriteLine(t.Sonuc(
    }
}
```

▲ 1 of 3 ▼ int Toplama.Sonuc(int a)

Main kodlarımız;

```
class Program
{
    static void Main(string[] args)
    {
        Toplama t = new Toplama();

        Console.WriteLine("1. Metod      : " + t.Sonuc(5));
        Console.WriteLine("2. Metod      : " + t.Sonuc(5, 6));
        Console.WriteLine("3. Metod      : " + t.Sonuc("Overload"));
    }
}
```



6.3. this Anahtar Kelimesi

this anahtar kelimesi bir sınıfta 4 farklı amaçla kullanılır.

- Varsayılan sınıfın örneğini referans eder.
- Benzer isimdeki alanların kullanımını sağlar.
- Başka bir metoda nesneyi parametre olarak aktarır.
- Sınıfın indexer(dizi) olarak kullanılmasını sağlar.

Örnek: 2 string ifadeyi birleştiren program.

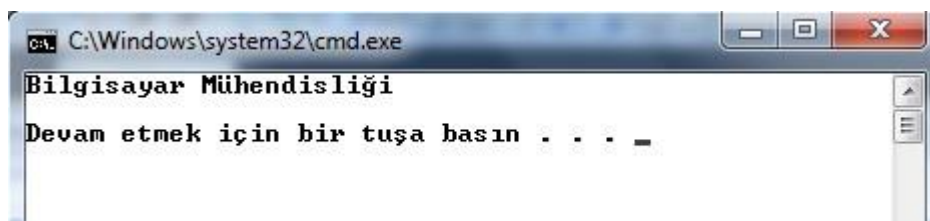
Islem sınıfı;

```
class Islem
{
    public string a;
    public string b;
    public string Birlestir()
    {
        return (this.a + " " + this.b);
    }
    public void EkranaYazdir()
    {
        Console.WriteLine(this.Birlestir());
        Console.WriteLine();
    }
}
```

Main kodları;

```
class Program
{
    static void Main(string[] args)
    {
        Islem i = new Islem(); i.a = "Bilgisayar";
        i.b = "Mühendisliği";
        i.EkranaYazdir();
    }
}
```

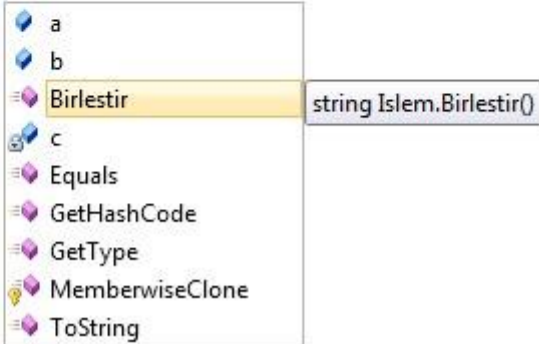
Ekran görünümü;



`this` anahtar kelimesinden sonra (.) nokta operatörü ile sınıfın tüm üyelerine ulaşılabilir.

```
class Islem
{
    public string a;
    public string b;
    private string c;

    public string Birlestir()
    {
        return (this.
```



6.4. Kurucular (Constructor)

Sınıfa ait bir nesne oluşturan sınıf ögesidir. Bir sınıfın kurucusu, sınıf içerisinde tanımlı olan, sınıfın adını alan ancak değer kümesi olmayan özel metotlardır. Bir nesnenin oluşturulduğu anda otomatik olarak çalıştırılan metotlardır. Bir sınıf içinde birden fazla kurucu olabilir. Nesne oluşturulurken hangi kurucu çağrılacağı parametrelerle belirlenir.

Bazen sınıflardan nesne oluşturulurken belirli şartlar yada değerler altında oluşturulması gerekebilir. Bu tür durumlarda sınıfla aynı ismi taşıyan ancak geri dönüş değeri olmayan metotlar kullanılır. Her sınıfın ister belirtilsin ister belirtilmesin en az 1 kurucusu vardır.

Örnek 1: tabanı 6, tavanı 4 ve yüksekliği 7 birim olan yamuğun alanını hesaplayan program.

Yamuk sınıfı;

```
class Yamuk
{
    private double taban;
    private double tavan;
    private double yukseklik;

    // Yamuk constructor
    public Yamuk()
    {
        taban = 6;
        tavan = 4;
        yukseklik = 7;
    }

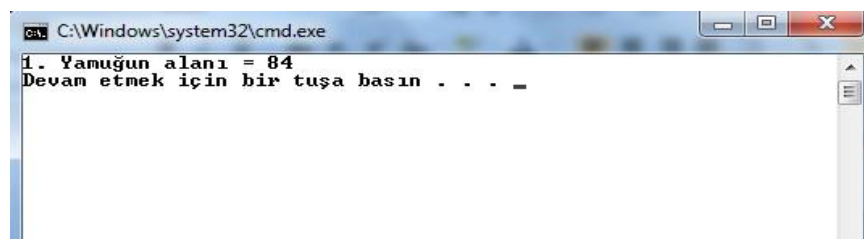
    // yamuğun alanını hesapla ve sonucu gönder
    public double alan()
    {
        return (taban * tavan) / 2 * yukseklik;
    }
}
```

Main kodları;

```
class Program
{
    static void Main(string[] args)
    {
        // Yamuk sınıfından nesne oluşturacaktır
        Yamuk ym1 = new Yamuk();

        // ym1 yamuğunun alanı yazdırılır
        Console.WriteLine("1. Yamuğun alanı = " + ym1.alan());
    }
}
```

Ekran görünümü;



6.5. new anahtar kelimesi

Sınıflardan nesne oluşturulurken new anahtar kelimesi kullanılmaktaydı.

```
Yamuk ym1 = new Yamuk();
```

new anahtar kelimesinden sonra Yamuk() şeklindeki ifade sınıftan nesne oluştururken hangi kurucunun çağrılacağı belirtilir.

6.6. Kurucuların Özellikleri

- Kurucular sınıf ismi ile aynı isimdedir.
- Geri dönüş değeri olmayan metotlardır (Gerçekte geri dönüş değerleri sınıfın kendisidir).
- Geri dönüş değeri olmamak haricinde metotların tüm özelliklerini (overload vb.) taşırlar.
- Nesne oluşturulurken ilk çalışan yapılardır.
- C# ve Java dillerinden belirtilmese bile her sınıfın varsayılan (parametresiz) kurucusu vardır.

6.7. Parametrelili Kurucular ve Kurucuların Aşırı Yüklenmesi

Parametresiz kurucular, her nesne oluşturulduğunda aynı değerleri verdiği için pek kullanışlı değildir.

Değişik değerlerdeki nesneleri oluşturmak için parametrelili kurucular programı daha kullanışlı yapar.

Örnek 2: tabanı 6, tavanı 4 ve yüksekliği 7 birim olan yamuk ile taban, tavan, yükseklik değerleri dışardan girilen yamuğun alanını hesaplayan program.

Yamuk sınıfı

```
public class Yamuk
{
    private double taban;
    private double tavan;
    private double yukseklik;
    // Yamuk constructor
    public Yamuk()
    {
        this.taban = 6;
        this.tavan = 4;
        this.yukseklik = 7;
    }
    // parametrelili constructor
    public Yamuk(double taban, double tavan, double yukseklik)
    {
        this.taban = taban; this.tavan = tavan;
        this.yukseklik = yukseklik;
    }
    // yamuğun alanını hesapla ve sonucu gönder
    public double alan()
    {
        return (this.taban * this.tavan) / 2 * this.yukseklik;
    }
}
```

Main kodları;

```
class Program
{
    static void Main(string[] args)
    {
        Yamuk ym1 = new Yamuk();
        Yamuk ym2 = new Yamuk(8, 3, 6);
        // ym1 yamuğunun alanı yazdırılır
        Console.WriteLine("1. Yamuğun alanı = " + ym1.alan());
        // ym2 yamuğunun alanı yazdırılır
        Console.WriteLine("2. Yamuğun alanı = " + ym2.alan());
    }
}
```

Örnek 3: Öğrenci bilgilerini tutan program.

Oğrenci sınıfı;

```
class Öğrenci
{
    private string adi;
    private string soyadi;
    private DateTime dogumTarihi;
    private string memleket;
    private string numara;
    private string bolum;
    private string fakulte;

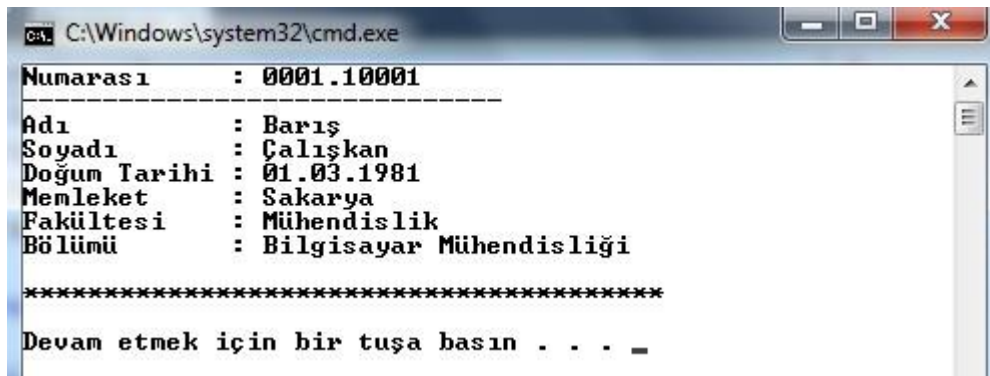
    public Öğrenci(string adi, string soyadi, DateTime dogumTarihi,
string memleket, string numara, string bolum, string fakulte)
    {
        this.adi = adi;
        this.soyadi = soyadi;
        this.dogumTarihi = dogumTarihi;
        this.memleket = memleket;
        this.numara = numara;
        this.bolum = bolum;
        this.fakulte = fakulte;
    }
    public void EkranaYazdir()
    {
        Console.WriteLine("Numarası      : " + this.numara);
        Console.WriteLine("-----");
        Console.WriteLine("Adı      : " + this.adi);
        Console.WriteLine("Soyadı    : " + this.soyadi);
        Console.WriteLine("Doğum Tarihi : " +
this.dogumTarihi.ToShortDateString());
        Console.WriteLine("Memleket   : " + this.memleket);
        Console.WriteLine("Fakültesi  : " + this.fakulte);
        Console.WriteLine("Bölümü     : " + this.bolum);
        Console.WriteLine();
        Console.WriteLine("*****");
        Console.WriteLine();
    }
}
```

Main kodları;

```
class Program
{
    static void Main(string[] args)
    {
        // o1 isimli ilk öğrenci nesnesi oluşturuluyor
        Öğrenci o1 = new Öğrenci("Barış", "Çalışkan", new DateTime(1981,
3, 1), "Sakarya", "0001.10001", "Bilgisayar Mühendisliği", "Mühendislik");

        //o1 nesnesi ekrana yazdırılıyor
        o1.EkranaYazdir();
    }
}
```

Ekran görünümü;



```
C:\Windows\system32\cmd.exe
Numarası      : 0001.10001
-----
Adı           : Barış
Soyadı        : Çalışkan
Doğum Tarihi  : 01.03.1981
Memleket      : Sakarya
Fakültesi     : Mühendislik
Bölümü        : Bilgisayar Mühendisliği
*****
Devam etmek için bir tuşa basın . . . _
```

İkinci öğrencimizi ekleyelim;

Main kodlarımız;

```
// o2 isimli ilk öğrenci nesnesi oluşturuluyor
Öğrenci o2 = new Öğrenci("Ahmet", "Şanslı", new DateTime(1983, 8, 17),
"Sakarya", "0101.10001", "Bilgisayar Mühendisliği", "Mühendislik");

//o2 nesnesi ekrana yazdırılıyor
o2.EkranaYazdir();
```

şeklinde olacaktır. Her seferinde eğer Öğrenci sınıfından oluşacak nesneyi ekrana yazdıracaksak eğer EkranaYazdir() metodunu kurucunun içerisine çekebiliriz.

```
class Ogrenci
{
    private string adi;
    private string soyadi;
    private DateTime dogumTarihi;
    private string memleket;
    private string numara;
    private string bolum;
    private string fakulte;
    public Ogrenci(string adi, string soyadi, DateTime dogumTarihi,
string memleket, string numara, string bolum, string fakulte)
    {
        this.adi = adi;
        this.soyadi = soyadi;
        this.dogumTarihi = dogumTarihi;
        this.memleket = memleket;
        this.numara = numara;
        this.bolum = bolum;
        this.fakulte = fakulte;

        EkранаYazdir();
    }
    public void EkранаYazdir()
    {
        Console.WriteLine("Numarası      : " + this.numara);
        Console.WriteLine("-----");
        Console.WriteLine("Adı      : " + this.adi);
        Console.WriteLine("Soyadı   : " + this.soyadi);
        Console.WriteLine("Doğum Tarihi : " +
this.dogumTarihi.ToShortDateString());
        Console.WriteLine("Memleket   : " + this.memleket);
        Console.WriteLine("Fakültesi  : " + this.fakulte);
        Console.WriteLine("Bölümü     : " + this.bolum);
        Console.WriteLine();
        Console.WriteLine("*****");
        Console.WriteLine();
    }
}
```

Main kodlarımızı ise oldukça sade bir hale gelecektir.

```
class Program
{
    static void Main(string[] args)
    {
        // o1 isimli ilk öğrenci nesnesi oluşturuluyor
        Ogrenci o1 = new Ogrenci("Barış", "Çalışkan", new DateTime(1981,
3, 1), "Sakarya", "0001.10001", "Bilgisayar Mühendisliği", "Mühendislik");

        // o2 isimli ilk öğrenci nesnesi oluşturuluyor
        Ogrenci o2 = new Ogrenci("Ahmet", "Şanslı", new DateTime(1983, 8,
17), "Sakarya", "0101.10001", "Bilgisayar Mühendisliği", "Mühendislik");

    }
}
```

4.4. Static Kurucular

C# ile ortaya çıkmış bir kavramdır. Sınıfa ait ilk dinamik nesne kurulmadan önce çalışan kuruculardır. Ancak sınıf içerisindeki sadece static¹ üyelerine erişebilir.

Özellikleri:

1. Her sınıfın ancak ve ancak tek bir static kurucusu olabilir.
2. Static kurucuların erişim belirteci olamaz.
3. Static kurucular parametresizdir.
4. Sınıfın ancak statik üyelerine erişebilir.
5. Bir sınıfta hem static hem de normal kurucu varsa öncelik static kurucularındır.

¹ Static kavramı ilerleyen derslerde işlenecektir.


```

class StatikSinif
{
    static DateTime zaman = DateTime.Now;

    public StatikSinif()
    {
        Console.WriteLine("ilk bu kurucu çalışacak mı?");
    }

    static StatikSinif()
    {
        Console.WriteLine(zaman);
    }
}

```

```

class Program
{
    static void Main(string[] args)
    {
        StatikSinif s = new StatikSinif();
    }
}

```

6.7 Yıkıcılar (Destructor)

Uygulama içerisinde işi biten nesnelerin hafızadan silinmesi gerekir. Çok çok nadir olarak da nesneler hafızadan silerken bazı işlemlerin gerçekleşmesi gerekebilir. Tam bu noktada sınıfların yıkıcıları devreye girmektedir. ~(tilda) işareti ile gösterilir, parametre almazlar.

Örnek:

```

class Personel
{
    ~Personel()
    {
        /// Yıkıcı çalışırken yapılması istenilenler
        ///
    }
}

```

Yıkıcılar, C++ programlarında nesnelerin hafızadan silinmesi programcı tarafından yapılmak zorundadır. C# ve Java'da programlarında hafıza yönetimi Garbage Collector adı verilen yapıların kontrolünde olduğundan, nesnenin işi bittikten sonra otomatik olarak kaldırılır. Böylece sınıflar için yıkıcı metotları yazmak zorunluluğu ortadan kalkar. Ne zaman kaldırılacağı tamamen GC'nin kontrolündedir.

Nesneler, çok özel durumlarda GC'nin çalışma algoritmasına bırakmadan, programdan kaldırılmak istendiğinde kurucuların tersi olan yıkıcılar kullanılmalıdır.

Özellikleri:

1. Bir sınıfın ancak ve ancak bir yıkıcısı olabilir.
2. Yıkıcılar aşırı yüklenemez.
3. Yıkıcılar kalıtım yolu ile geçmez.
4. Yıkıcılar çağrılmaz, otomatik olarak çalışır.
5. Yıkıcılar parametre alamaz.
6. Yıkıcılar değiştirilemez.
7. Yıkıcılar yalnızca sınıflarda geçerlidir. Yapılarda (struct) yoktur.