Bölüm 15

String Sınıfı

String Sınıfı

`String`, `.NET Framework` içinde `System` aduzayına (namespace) ait bir sınıftır. Dolayısıyla `System.String` sınıfını C# olduğu gibi alır ve kullanır. Klâsik dillerde *string* karakterlerden oluşan bir *array* olarak tanımlanır. `System.String` sınıfı da o geleneği sürdürür, ama çok farklı nitelikler ekleyerek. Bir string (nesnesi) unicode karakterlerden oluşan metindir (text). Başka bir deyişle `System.Char` nesnelerinin bir dizisidir. Ama bu dizi bir bütündür, parçalanamaz, değiştirilemez. Bir stringe *ekleme, birleştirme, ayırma, kırpma (trim), silme, yerine koyma (replace)* gibi klâsik metin işlemleri uygulanamaz. Bu işlemlerden herhangi birisi uygulandığında başka bir string yaratılmış olur. Gerçekten, aşağıda göreceğimiz ve string üzerinde değişiklik yapan metotlar, söz konusu stringi (değerini) değiştirmezler, onun yerine her değişiklik için yeni bir string yaratırlar.

`String` sınıfı, girdi/çıktı eylemlerinde yaşamsal role sahiptirler. Metinler, sayılar, tarihler gibi farklı veri tipleri bilgisayara string olarak girer, bilgisayardan string olarak çıkarlar. Kullanıcı ile bilgisayarın etkileşim içinde olabilmesi için, giriş işleminde *string 'den veri tipine*, çıkışta ise *veri tipinden string'e* gerekli dönüşümler yapılır. `String` 'in bu önemli işlevleri nedeniyle, `.NET Framework` string işlemleri yapmaya yarayan çok sayıda metoda sahiptir. Bunlara ek olarak çok sayıda arayüz ile `StringBuilder`, `String.Format`, `StringCollection`, vb. sınıflar da string işlemlerinin yapılmasını kolaylaştırırlar. Bunların hepsini sistematik olarak açıklamak bu kitabın amacı dışındadır. Konuyu aydınlatmaya yetecek örnekler vermekle yetineceğiz. Zaten önceki bölümlerde `string` kavramını çok kullandık. Sonraki bölümlerde de farklı niteliklerini kullanmaya devam edeceğiz. Bütün bunların ötesinde `string` 'in sistematik yapısını incelemek isteyenler msdn web sitesine bakabilirler.

.NET Framework, `System.String` sınıfına kısa bir takma ad veriris `string`. Biz de bu ikisini eşanlamlı kullanacağız; yani `System.String` ve `string` aynı anlama sahiptirler.

Öte yandan, string çok özel bir sınıftır. Diğer sınıflarda olmayan nitelikleri vardır. Çok sayıdastatik öğesi vardır; dolayısıyla, ona ait bir nesne yaratmaya gerek kalmaksızın onları kullanabiliriz. Örneğin,

```
string str = "Ankara";
```

deyimini yazdığımızda `str` referansı tarafından işaret edilen bir string nesnesi yaratıp ona değer atamış oluruz; ayrıca `new` operatörünü kullanmaya gerek yoktur. Böyle olduğu için `string` bir referans tipidir, *string tipi* bir nesneyi işaret eder. Bir nesne işaret etmediği zaman değeri `null` olur; yani hiç bir adresi göstermez. Bu demektir ki,

```
string str = null ;
```

deyimi geçerlidir.

String Sınıfının Özgenleri (properties)

String.Chars : Bir string içinde belirlenen bir yerdeki karakteri söyler.

String.Length : string'in uzunluğunu; yani kaç karakterden oluştuğunu söyler.

Aşağıdaki program, bir stringin uzunluğunu ve buluyor ve stringi bir karakter arrayi gibi alıp öğelerini sırayla yazıyor

StrLength.cs

```
using System;

class StrLength
{
    public static void Main()
    {
        string s = "Bilgimiz gücümüzdür!";
        for (int i = 0; i < s.Length; i++)
        {
            Console.Write("s[" + i + "]  = " + s[i]);
        }
    }
}
```

String Sınıfının Metotları

String sınıfının 50 den çok metodu vardır. Bu metotlar, stringlerle yapılabilecek *mukayese, ekleme, insert, dönüştürme, copyalama, formatlama, indexleme, birleştirme, ayırma, doldurma, kırpma, silme, yerine koyma ve arama* gib eylemleri yapmamızı sağlarlar. Aşağıda bunların bazı örneklerini göreceğiz.

String Bildirimi

`string` sınıfı, .NET Framework içinde öntanımlı bir sınıftır. Anımsayacağınız gibi, öntanımlı sınıflar klâsik dillerdeki veri tipi gibi kullanılabilirler; yani onlara ait nesneleri bizim yaratmamıza gerek yoktur; kullanacağımız metotlarının çoğu statiktir. Dolayısıyla, *string bildirimi* için şu yolları izleyebilirizis

```
string birMetin = "Ankara başkenttir"; // metin
String birsayı = "123456789"; // sayı
System.String birTarih = "15.08.2008"; // tarih
string kod = "\\\u0084\n";  // backslash,  ? işareti, satırbaşı
```

Metot Örnekleri

Yukarıdaki bildirimlerin tiplerinin `System.String` olduğunu görmek için `GetType()` metodunu kullanabilirizis

String01.cs

```
using System;

namespace Stringler
{
    class String01
    {
        static void Main(string[] args)
        {
            string birMetin = "Ankara başkenttir";
            String birSayı = "123456789";
            System.String birTarih = "15.08.2008";

            Console.WriteLine(birMetin.GetType());
            Console.WriteLine(birSayı.GetType());
            Console.WriteLine(birTarih.GetType());
        }
    }
}
```

Çıktı

```
System.String
System.String
System.String
```

String nesnedir, değiştirilemez (immutable)

Bildirimi yapılan bir string üzerinde herhangi bir değişiklik yapılamayacağını, yapılan her değişikliğin başka bir string yarattığını söylemiştik. Bunu aşağıdaki örnekten görebiliriz. `str1` stringi `StringDeğiştir()` metodu ile değiştirilmek isteniyor, ama değişiklik olmuyor, ona dokunulmadan başka bir string yaratılıyor; orijinal `str1` değişmeden kalıyor.

Değişmez.cs

```
using System;

class Değişmez
```

```
{
    static void Main(string[] args)
    {
        string str1 = "Doğayı koru!" ;
        Console.WriteLine("str1 in ilk değeriis {0}", str1);
        Console.WriteLine(StringDeğiştir(str1));
        Console.WriteLine("str1 in son değeriis {0}", str1);
    }

    static string StringDeğiştir(string str)
    {
        str = "Doğayı kimler yokediyor?";
        return str;

    }
}
```

Çıktı

```
str1 in ilk değeriis Doğayı koru!
Doğayı kimler yokediyor?
str1 in son değeriis Doğayı koru!
```

Stringleri Birleştirme

Stringleri birleştirmek için değişik yöntemler kullanabiliriz. Sayılar için tanımlanan (+) ve (+=) operatörleri string tipi için *override* edilmişlerdir. Bunlar kullanılabileceği gibi, `String` sınıfının bir öğesi olan `Concat()` metodu da kullanılabilir.

Aşağıdaki üç deyim aynı işi yaparis

```
string s1 = "Bu " + "gün " + "hava " + "çok sıcak" + ".";
```

```
string s2 = "Bu ";
       s2 += "gün ";
       s2 += "hava ";
       s2 += "çok sıcak ";
       s2 += ".";
```

```
string s3 = String.Concat("Bu ", "gün ", "hava ", "çok sıcak", ".");
```

Gerçekten aynı olduklarını görmek istersek, bunları bir program içine alıp sonuçları yazdırabilirizis

Birleştir01.cs

```
using System;

class Birleştir01
{
    public static void Main()
    {
        string s1 = "Bu " + "gün " + "hava " + "çok sıcak" + ".";
```

```
        string s2 = "Bu ";
        s2 += "gün ";
        s2 += "hava ";
        s2 += "çok sıcak ";
        s2 += ".";

        string s3 = String.Concat("Bu ", "gün ", "hava ", "çok sıcak", ".");
        Console.WriteLine(s1);
        Console.WriteLine(s2);
        Console.WriteLine(s3);

    }
}
```

Çıktı

```
Bu gün hava çok sıcak.
Bu gün hava çok sıcak .
Bu gün hava çok sıcak.
```

Benzer işi, değişkenleri kullanarak da yapabiliriz. `GetType()` metodu veri tipini belirtir.

Birleştir02.cs

```
using System;
namespace Stringler
{
    class Birleştir02
    {
        static void Main(string[] args)
        {
            string s1 = "Ankara";
            Console.WriteLine(s1);
            System.String s2 = "başkenttir";
            String boşluk = "  ";

            Console.WriteLine("s1 = " + s1);
            Console.WriteLine("s2 = " + s2);
            Console.WriteLine("s1 + boşluk + s2 =  " + s1 + boşluk + s2);
            Console.WriteLine("s1.GetType()      = "  + s1.GetType());
            Console.WriteLine("s2.GetType()      = "  + s2.GetType());
            Console.WriteLine("boşluk.GetType() = " + boşluk.GetType());
        }
    }
}
```

Çıktı

```
Ankara
s1 = Ankara
s2 = başkenttir
s1 + boşluk + s2 =  Ankara  başkenttir
s1.GetType()      = System.String
s2.GetType()      = System.String
boşluk.GetType() = System.String
```

Stringleri Karşılaştırma

Genel olarak, iki referans tipinin (reference type) karşılaştırılması demek, işaret ettikleri *bellek adreslerinin* aynı olup olmadığının belirlenmesi demektir. Bellek adreslerinde yazılı veriye bakılmaz. Tersine olarak, iki değer tipinin (value type) karşılaştırılması demek, *bellek adreslerindeki verilerin* aynı aynı olup olmadığının belirlenmesi demektir. Zaten değer tipinden farklı iki değişkenin adresleri farklı olacağı için, onların adreslerini karşılaştırmanın bir anlamı olamaz; onun yerine o farklı adreslerde yazılı verilerin eşit olup olmadıklarına bakılır.

`String` de referans tipi olduğu için, iki stringin karşılaştırılmasında, işaret ettikleri adreslerin aynı olup olmadığı belirlenir kanısına varmak gerekiyor. Oysa, iki stringin karşılaştırılmasında bellek adreslerinin aynı olup olmadığına değil, onları oluşturan karakter dizilerinin aynı olup olmadığına bakılır, yani değer tipinde olduğu gibi bellek adreslerinde yazılı olan değerler karşılaştırılır. Stringlerin karşılaştırılmasında ortaya çıkan bu özelik, onların değer tipi imiş gibi algılanmasına neden olabilir. Ancak, bu özeliğin bilinçli olarak yapıldığını söylemeliyiz. Stringleri karşılaştırırken, kullanıcı onların bellek adresleriyle değil, değerleriyle ilgilidir.

Stringleri karşılaştırmak için `==` ile `!=` ilişkisel operatörlerini ya da `Compare()` metodunu kullanabiliriz. İlşkisel operatörler yalnızca stringlerin eşit olup olmadığını `True` veya `False` değerleriyle belirtir. `Compare()` metodu ise çok daha fazla işler yapar. Aşağıdaki iki örnek bunları göstermektedir.

Mukayese01.cs

```
using System;
namespace Stringler
{
    class Mukayese01
    {
        static void Main(string[] args)
        {
            string s1 = "Ankara";
            string s2 = "istanbul";
            string s3 = "ANKARA";
            string s4 = "Ankara";

            Console.WriteLine(s1 == s2); // False
            Console.WriteLine(s1 != s2); // True
            Console.WriteLine(s1 == s4); // True
            Console.WriteLine(s1 == s3); // False
            Console.WriteLine("Ankara" == "ANKARA"); // False
            Console.WriteLine("Ankara" != "ANKARA"); // True

        }
    }
}
```

Aşağıdaki programda `s1` ve `s2` stringlerinin içerikleri aynı, adresleri farklıdır. `s1 == s2` karşılaştırmasının `True`, ama object olarak `(object)s1 == (object)s2` karşılaştırmasının `False` sonuç verdiğine dikkat ediniz.

Mukayese02.cs

```
using System;
namespace Stringler
```

```
{
    class Mukayese02
    {
        static void Main(string[] args)
        {
            string s1 = "Ankara";
            string s2 = "An";
            s2 += "kara"; // s1, s2 farklı adreste
            Console.WriteLine(s1.ToString());
            Console.WriteLine(s2);
            Console.WriteLine(s1 == s2);
            Console.WriteLine((object)s1 == (object)s2);
            Console.WriteLine((object)s1) ;
            Console.WriteLine((object)s2);
        }
    }
}
```

Çıktı

```
Ankara
Ankara
True
False
Ankara
Ankara
```

Compare() Metodu

Stringleri karşılaştırmak için `string` sınıfının bir öğesi olan `Compare()` metodunu kullanarak `==` ile `!=` ilişkisel bağıntılarının verdiğinden fazlasını elde edebiliriz. Gerçekten bu metot, farklı diller, farklı alfabeler ve farklı kültürlerde yazılabilecek stringleri karşılaştırır; eşit olup olmadıklarını; eşit değillerse hangisinin büyük, hangisinin küçük olduğunu bildirir. Dünyadaki bütün diller, bütün alfabeler ve bütün kültürler için işlevselliği olan bu metot default olarak Windows İşletim Sisteminin diline ve kültürüne bağlıdır. Türkçe Windows kullanıyorsak, Türk alfabesine göre karşılaştırma yapacaktır.

Yapılan karşılaştırma, ilgili kültürün alfabetik sıralamasına (sözlük sıralaması) göredir.

`Compare()` metodunun değer kümesi `Int32` veri tipidir. Bu demektir ki, `s1` ile `s2` stringlerini mukayese etmek için

```
String.Compare(s1,s2) ;
```

deyimini yazarsak, metodun alacağı değer işaretli bir `int` tipi sayıdır. Eğer bu sayı negatif ise `s1 < s2` dir, sıfır ise `s1 = s2` dir, pozitif ise `s1> s2` dir. Bu söylediklerimizi aşağıdaki tabloda gösterebiliriz.

Compare(s1,s2)	Anlamı
Negatif	`s1 < s2`

Sıfır	`s1 = s2`
Pozitif	`s1 > s2`

Karşılaştırılan stringlerden birisi veya her ikisi de `null` ya da `boş` (`""`) string olabilir. `null` bütün stringlerden küçüktür. *Boş string* ise, boş olmayan her stringden küçüktür. İki `null` ya da iki *boş string* karşılaştırılırsa 0 (eşit) değeri çıkar.

Mukayese03.cs

```
using System;
namespace Stringler
{
    class Mukayese03
    {
        static void Main(string[] args)
        {
            string s1 = "Ankara";
            string s2 = "istanbul";
            string s3 = "ANKARA";
            string s4 = "Ankara";

            Console.WriteLine(String.Compare(s1,s2));       // -1
            Console.WriteLine(String.Compare(s1, s3));      // -1
            Console.WriteLine(String.Compare(s1, s4));      //  0
            Console.WriteLine(String.Compare(null, s2));    // -1
            Console.WriteLine(String.Compare(s1, null));    //  1
            Console.WriteLine(String.Compare(null, null));  //  0
            Console.WriteLine(String.Compare("", s2));      // -1
            Console.WriteLine(String.Compare(s1, ""));      //  1
            Console.WriteLine(String.Compare("", ""));      //  0

        }
    }
}
```

Equals() Metodu

İki stringin eşit olup olmadığını anlamak için `Equals()` metodunu da kullanabiliriz. Aşağıdaki programda bu metodun *overload* edilmiş iki sürümünü göreceksiniz. Programı satır satır çözümleyiniz.

Equal01.cs

```
using System;

class Equal01
{
    public static void Main()
    {
        bool b;
        string s1 = "abcd";
        string s2 = "abcde";
```

```
        b = String.Equals("abc", "abc");
        Console.WriteLine("String.Equals(\"abc\", \"abc\") = " + b);

        b = s1.Equals(s2);
        Console.WriteLine("s1.Equals(s2) = " + b);

        b = s1 == s2;
        Console.WriteLine("s1 == s2 = " + b);
    }
}
```

Çıktı

```
String.Equals("abc", "abc") = True
s1.Equals(s2) = False
s1 == s2 = False
```

Copy() metodu

Bir stringi başka bir stringe kopyalamak için `Copy()` metodunu kullanırız. Aşağıdaki program `s2` stringini `s1` stringine kopyalıyor.

Kopya01.cs

```
using System;

class Kopya01
{

    public static void Main()
    {
        string s1 = "Ankara";
        string s2 = "C# öğreniyorum";
        Console.WriteLine("s1 = " + s1);
        Console.WriteLine("s2 = " + s2);
        Console.WriteLine("s2 stringi s1 'e kopyalanıyor");
        s1 = String.Copy(s2);
        Console.WriteLine("s1 = " + s1);
    }
}
```

Çıktı

```
s1 = Ankara
s2 = C# öğreniyorum
s2 stringi s1 'e kopyalanıyor
s1 = C# öğreniyorum
```

Insert(), Remove(), Replace() metotları

`Insert()` metodu bir string'in belirlenen bir yerine başka bir string'i yerleştirir. İki parametresi vardır. Birinci parametrede string'in nereye yerleştirileceğini belirten bir tamsayıdır. İkinci parametre ise birinci stringe yerleştirilecek olan string'dir.

Yerleştir01.cs

```
using System;

class Yerleştir
{
    public static void Main()
    {
        string str1 =  "C# kolaydır.";
        string str2 = " dilini öğrenmek çok ";
        Console.WriteLine("str1 = "  + str1);
        Console.WriteLine("str2 = " + str2);
        str1 = str1.Insert(3, str2);
        Console.WriteLine("str1 =  " + str1);
    }
}
```

Çıktı

```
str1 =  C# kolaydır.
str2 =  dilini öğrenmek çok
str1 =  C#  dilini öğrenmek çok kolaydır.
```

Replace() metodu

Bir string içindeki bir karakter veya bir alt-stringi bulur ve onun yerine istenen başka bir harf veya stringi koyar. İki parametresi vardır. Birincisi aranacak harf veya alt-string, ikincisi bulunanın yerine konulacak harf veya string.

Aşağıdaki program l harfi yerine h harfini koyar; sonuçta *Ceylan* stringi *Ceyhan* stringine dönüşür.

Replace01.cs

```
using System;

class Replace01
{
    public static void Main()
    {
        string str = "Ceylan";
        Console.WriteLine(str);
        str = str.Replace('l', 'h'); // l yerine h koy
        Console.WriteLine(str);
    }
}
```

Aşağıdaki program *"C# dili çok uzun zamanda öğrenilir."* stringinde *"uzun"* alt-stringi yerine *"kısa"* stringini koyar.

Replace02.cs

```
using System;

class Replace02
```

```
{
    public static void Main()
    {
        string str1 = "C# dili çok uzun zamanda öğrenilir.";
        string str2 = " kısa ";
        Console.WriteLine("str1 = " + str1);
        Console.WriteLine("str2 = " + str2);
        str1 = str1.Replace("uzun", str2);
        Console.WriteLine("str1 =  " + str1);
    }
}
```

`Remove()` Metodu

Bir stringden bir karakter veya bir alt stringi siler. İki tamsayı parametresi vardır. Birinci parametre silme eyleminin kaçıncı karakterden başlayacağını, ikinci parametre kaç karakter silineceğini gösterir.

Aşağıdaki program *"C# dili çok ama çok kısa zamanda öğrenilir."* stringinden *"ama çok"* alt-stringini siler ve *"C# dili çok kısa zamanda öğrenilir."* stringine dönüştürür.

Yoket01.cs

```
using System;

class Yoket01
{
    public static void Main()
    {
        string str1 = "C# dili çok ama çok kısa zamanda öğrenilir.";
        Console.WriteLine("str1 = " + str1);
        str1 = str1.Remove(12, 8);
        Console.WriteLine("str1 = " + str1);
    }
}
```

Çıktı

```
str1 = C# dili çok ama çok kısa zamanda öğrenilir.
str1 = C# dili çok kısa zamanda öğrenilir.
```

ToUpper() ve ToLower() Metotları

ToUpper() metodu bir stringdeki bütün harfleri büyük harf karşılıklarına dönüştürür. ToLower() metodu ise tersini yapar, bir stringdeki bütün büyük harfleri küçük harf karşılıklarına dönüştürür.

Aşağıdaki program bu iki dönüşümün kullanılışını göstermektedir.

UpperLower.cs

```
using System;

class UpperLower
{
```

```
    public static void Main()
    {
        string str1 =  "C# dili çok kısa zamanda öğrenilir.";
        Console.WriteLine("str1 = " + str1);
        // büyük harfe dönüştür
        string str2 = str1.ToUpper();
        Console.WriteLine("str1.ToUpper() =  " + str2);

        // küçük harfe dönüştür
        str2 = str2.ToLower();
        Console.WriteLine("str2.ToLower() =  " + str2);
    }
}
```

Çıktı

```
str1 = C# dili çok kısa zamanda öğrenilir.
str1.ToUpper() =  C# DİLİ ÇOK KISA ZAMANDA ÖĞRENİLİR.
str2.ToLower() =  c# dili çok kısa zamanda öğrenilir.
```

Split() ve Join() Metotları

Başka bir sistem veya programın girdi/çıktısı ile bağlantılı bir iş yaparken, farklı biçemdeki (format) verileri kullanmamız gerekebilir. Örneğin, bir çok programın çıktısı Excel biçemine dönüşebilir ve bir çok program Excel biçemindeki verileri okuyabilir. Excel *Comma Separated Value (CSV) format* denilen virgüllerle ayrılmış verileri girdi/çıktı olarak kabul eder. C# ile Excel arasında veri alışverişi yapabilmemiz için, örneğin, CVS biçemindeki verileri bir array'e dönüştürmek ya da tersine olarak bir array'i CVS biçemine dönüştürmek gerekecektir. Bu iş için Split ve Join() metotlarını kullanırız. Aşağıdaki örnekler, bu metotların nasıl kullanıldığını göstermektedir.

Aşağıdaki program bir string array'ini CVS biçemine dönüştürmektedir. [Array'in öğelerini (veriler) birbirinden ayırmak için virgül veya başka bir karakter kullanılabilir. Bu örnekte `'*'` kullanılmıştır.]

Join01.cs

```
  using System;

class MainClass
{
    public static void Main()
    {
        string[] dizi1 = { "C#", "ile", "Nesne", "Yönelimli",
"programlama", "öğreniyorum" };
        string dizi2 = String.Join("*", dizi1);
        Console.WriteLine("dizi2 = " + dizi2);
    }
}
```

Çıktı

```
dizi2 = C#*ile*Nesne*Yönelimli*programlama*öğreniyorum
```

Aşağıdaki program bir arrayi önce CVS biçemine dönüştürüyor, sonra onun virgülle birbirlerinden ayrılmış öğelerini (veriler) listeliyor.

Split01.cs

```
using System;

class MainClass
{
    public static void Main()
    {
        string[] dizi1 = { "C#", "ile", "Nesne", "Yönelimli",
"programlama", "öğreniyorum" };
        string dizi2 = String.Join(",", dizi1);
        Console.WriteLine("dizi2 = " + dizi2);

        dizi1 = dizi2.Split(',');
        foreach (string s in dizi1)
        {
            Console.WriteLine("s = " + s);
        }
    }
}
```

Çıktı

```
dizi2 = C#,ile,Nesne,Yönelimli,programlama,öğreniyorum
s = C#
s = ile
s = Nesne
s = Yönelimli
s = programlama
s = öğreniyorum
```

Aşağıdaki program CVS olarak verilen ayları öğelerine ayırıyor, sonra onları (;) ile birleştirip yazıyor. Ayrıca istenen bir mevsime ait ayları buluyor. Programı satır satır çözümleyiniz.

StrJoinSplit.cs

```
using System;

namespace SplitJoin
{
    class StrJoinSplit
    {
        static void Main(string[] args)
        {
            // CVS biçemindeki string
            string cvs =
"Ocak,Şubat,Mart,Nisan,Mayıs,Haziran,Temmuz,Ağustos,Eylül,Ekim,Kasım,Aral
ık";

            Console.WriteLine("Original CVS Stringiis \n{0}\n", cvs);

            // separate individual items between commas
            string[] sene = cvs.Split(new char[] { ',' });

            Console.WriteLine("Ayrılmış öğeleris ");

            foreach (string ay in sene)
```

```
            {
                Console.Write("{0} ", ay);
            }
            Console.WriteLine("\n");

            // Arrayin öğelerini ; ile birleştir
            string cvs2 = String.Join(";", sene);

            Console.WriteLine("; ile birleşen arrayis \n{0}\n", cvs2);

            string[] mevsim = cvs.Split(new Char[] { ',' }, 3);

            Console.WriteLine("İlk üç öğeis ");

            foreach (string ay in mevsim)
            {
                Console.Write("{0} ", ay);
            }
            Console.WriteLine("\n");

            string güz = String.Join("/", sene, 6, 3);

            Console.WriteLine("Güz mevsimiis \n{0}\n", güz);
        }
    }
}
```

IndexOf() Metodu

Bir string içindeki bir harf veya alt-stringin kaçıncı karakterden başladığını belirtir. Metodun birisi zorunlu üç parametresi vardır. Zorunlu olan birincisi aranan harf veya stringdir. İkincisi kaçıncı harften başlayarak arama yapılacağını, üçüncü parametre ise kaç harf boyunca arama yapılacağını gösterir. Metod aranan bulamazsa -1 değerini alır.

IndexOf01.cs

```
using System;

class IndexOf01
{
    public static void Main()
    {
        string str1 =  "C# dili çok kısa zamanda öğrenilir.";
        Console.WriteLine("str1 = " + str1);
        // "kısa" stringinin yerini bulur
        Console.WriteLine("str1.IndexOf(\"kısa\") =  " +
str1.IndexOf("kısa"));
    }
}
```

Çıktı

```
str1 = C# dili çok kısa zamanda öğrenilir.
str1.IndexOf("kısa") =  12
```

Aşağıdaki program *"C# dili çok kısa zamanda öğrenilir."* stringi içinde `'i'` harfini 10-uncu karakterden sonra aramaya başlıyor ve aramayı 5 karakter boyunca sürdürüyor. Bulamadığı için -1 değerini alıyor. Programda üçüncü parametre olarak 21 ile 25 arasında bir sayı yazarsanız metodun değeri 30 olur; yani aradığı `'i'` harfinin yeri stringin 30-uncu karakteridir. Üçüncü parametre 25 den büyük olursa, string uzunluğu dışına taşıldığını belirten şu hata mesajını verir.

Unhandled Exceptionis System.ArgumentOutOfRangeExceptionis Count must be and count must refer to a location within the string/array/collection.

IndexOf02.cs

```
using System;

class IndexOf02
{
    public static void Main()
    {
        string str1 =  "C# dili çok kısa zamanda öğrenilir.";
        Console.WriteLine("str1 = " + str1);
        // "kısa" stringinin yerini bulur
        Console.WriteLine("str1.IndexOf('i',10,5 ) =  " +
str1.IndexOf('i',10,5));
    }
}
```

IndexOfAny() Metodu

Bir metinde istenmeyen harflerin olup olmadığını bilmek isteyebiliriz. Bu durumda `IndexOfAny()` metodunu kullanırız. Bu metot bir string'in içerisinde bir karakter array'ine ait herhangi bir karakterin olup olmadığını denetler; varsa bulduğu karakterin string içinde kaçıncı karakterden sonra geldiğini belirtir. Parametreleri `IndexOf()` fonksiyonu ile aynıdır. Metot aradığını bulduğu ilk yeri belirten tamsayı değerini alır; bulamazsa -1 değerini alır.

Aşağıdaki örnek, aranan (#) karakterinin *"Çok kısa zamanda C# dili öğrenilir."* stringinin 18-inci karakter olduğunu bildirir.

```
using System;

class IndexOf02
{
    public static void Main()
    {
        string str1 =  "Çok kısa zamanda C# dili öğrenilir.";
        Console.WriteLine("str1 = " + str1);
        char[] dizi = {'$','#','%','*'};
        int n = str1.IndexOfAny(dizi, 10, 24);
        Console.WriteLine(n);
    }
}
```

LastIndexOf()

`public int LastIndexOf(char, int, int)` metodu bir string içinde bir karakterin en sondaki yerini bulur. Metodun üç parametresi vardır. Birinci parametre aranan, ikinci parametre aramaya nereden başlanacağını, üçüncü parametre aramanın kaç karakter süreceğini belirtir.

LastIndexOf01.cs

```
using System;

class LastIndexOf01
{
    public static void Main()
    {
        string str = "Bu bir denemedir.";

        Console.WriteLine(str.LastIndexOf(' ', 5, 6)); //6
        Console.WriteLine(str.LastIndexOf(' ', 6, 6)); //6
        Console.WriteLine(str.LastIndexOf(' ', 7, 6)); //6
        Console.WriteLine(str.LastIndexOf(' ', 8, 6)); //6
        Console.WriteLine(str.LastIndexOf(' ', 9, 6)); //6
        Console.WriteLine(str.LastIndexOf(' ',10, 6)); //6
        Console.WriteLine(str.LastIndexOf(' '));       //6
    }
}
```

PadLeft() ve PadRight() metotları

Bir stringin soluna ya da sağına istenildiği kadar başka karakter veya boşluk koyarlar.

Aşağıdaki örnekte stringin soluna ve saına birer kez boş karakter, birer kez de '*' karakteri konulmuştur.

Padding01.cs

```
using System;

class Padding01
{

    public static void Main()
    {
        string str = "C# ile Nesne Programlama";
        Console.WriteLine("str.PadLeft(30)       = |{0}|" ,
str.PadLeft(30));
        Console.WriteLine("str.PadLeft(30, '*')  = |{0}|" ,
str.PadLeft(30, '*'));
        Console.WriteLine("str.PadRight(30)      = |{0}|" ,
str.PadRight(30));
        Console.WriteLine("str.PadRight(30,'*')  = |{0}|" ,
str.PadRight(30,'*'));
    }
}
```

Çıktı

```
str.PadLeft(30)       = |       C# ile Nesne Programlama|
```

```
str.PadLeft(30, '*')   = |******C# ile Nesne Programlama|
str.PadRight(30)       = |C# ile Nesne Programlama      |
str.PadRight(30,'*')   = |C# ile Nesne Programlama******|
```

Trim(), TrimStart(), and TrimEnd() metotları

`Trim()` metodu parametresiz olarak kullanıldığında string'in önündeki ve gerisindeki boşluk karakterlerini (white space) siler. `TrimStart()` metodu string'in önündeki boşluk karakterlerini siler. `TrimEnd()` metodu string'in gerisindeki boşluk karakterlerini siler. Silinmesi istenen karakterler arrayi bir parametre olarak kullanılabilir.

Aşağıdaki program çeşitli kırpma yöntemlerini göstermektedir.

Trim01.cs

```
using System;

public class Trim01
{
    public static void Main()
    {
        string s1 = "    abc     ";
        Console.WriteLine("s1        = |{0}|" , s1);
        Console.WriteLine("s1.Trim() = |{0}|" , s1.Trim());

        string s2 = "__...,, ABC,..._";
        Console.WriteLine("s2 = " + s2);
        char[] trimEdilecek = new char[] { '.', ',', '_' };
        Console.WriteLine("trimEdilecek = { '.', ',', '_' }");
        Console.WriteLine("s2.Trim(trimEdilecek)      = |
{0}|",s2.Trim(trimEdilecek));
        Console.WriteLine("s2.TrimStart(trimEdilecek) = |{0}|"
,s2.TrimStart(trimEdilecek));
        Console.WriteLine("s2.TrimEnd(trimEdilecek)   = |{0}|",
s2.TrimEnd(trimEdilecek));
    }
}
```

Çıktı

```
s1           = |    abc     |
s1.Trim()    = |abc|
s2           = __...,, ABC,..._
trimEdilecek = { '.', ',', '_' }
s2.Trim(trimEdilecek)      = | ABC|
s2.TrimStart(trimEdilecek) = | ABC,..._|
s2.TrimEnd(trimEdilecek)   = |__...,, ABC|
```

Substring() metodu

`Substring()` metodu bir string'in içerisinde başlangıç yeri ve uzunluğu belitilen alt-stringi seçer. İki parametresinden ilki string'den seçilecek alt-stringin başlangıç yerini, ikinci parametre ise seçilecekstringin uzunluğunu belirtir. İkinci parametre yazılmazsa string'in sonuna kadar seçilir.

Aşağıdaki program *"Çok kısa zamanda C# dili öğrenilir."* stringinden *"C# dili"* alt-stringini seçer.

Substring01.cs

```
using System;

class Substring01
{
    public static void Main()
    {
        string str1 =  "Çok kısa zamanda C# dili öğrenilir.";
        Console.WriteLine("str1 = " + str1);
        string altString = str1.Substring(17,7);
        Console.WriteLine("str1.Substring(17,7) = " + altString);
    }
}
```

Alıştırmalar

1. Aşağıdaki program if-else yapısıyla iki metni mukayese eder. Programı satır satır çözümleyiniz.

String02.cs

```
using System;

namespace Stringler
{
    public class String02
    {
        public static void Main(string[] args)
        {
            string p = "Ankara";
            string q = "ankara";
            if (p == q)         // farklıdır
            {
                Console.WriteLine("Aynı");
            }
            else
            {
                Console.WriteLine("Farklı");
            }
        }
    }
}
```

Çıktı

```
Farklı
```

2. Aynı mukayeseyi **`public static int Compare(`**string `a` **`,`** string *`b`* **`)`** metodu ile daha kolay yapabiliriz. Bu metot `a` ve `b` yi karşılaştırır. Sözlük sıralamasına göre `a < b` ise negatif bir değer, `a ==b` ise 0 değerini, `a > b` ise pozitif bir değer alır.

   Aşağıdaki Programı satır satır çözümleyiniz.

String03.cs

```
using System;

namespace Stringler
{
    public class String01
    {
        public static void Main(string[] args)
        {
            string p = "Ankara";
            string q = "ankara";
            Console.WriteLine(String.Compare(p, q)); // çıktıis "1"
        }
    }
}
```

Sözlük sıralamasına göre "Ankara" > "ankara" olduğu için Compare(p,q) metodu "1" değerini almaktadır.

3. Aşağıdaki Programı satır satır çözümleyiniz.

IndexOf.cs

```
using System;

class IndexOf
{

    public static void Main()
    {
        string[] myStrings = { "To", "be", "or", "not", "to", "be" };
        string myString = String.Join(".", myStrings);

        char[] myChars = { 'b', 'e' };
        int index = myString.IndexOfAny(myChars);
        Console.WriteLine("'b' and 'e' occur at index " + index + " of
myString");
        index = myString.LastIndexOfAny(myChars);
        Console.WriteLine("'b' and 'e' last occur at index " + index + "
of myString");

    }

}
```

4. Aşağıdaki Programı satır satır çözümleyiniz.

```
using System;

class MainClass
{
    public static void Main()
    {
        string[] myStrings = { "To", "be", "or", "not", "to", "be" };
        string myString = String.Join(".", myStrings);

        string myString21 = myString.Substring(3);
        Console.WriteLine("myString.Substring(3) = " + myString21);
        string myString22 = myString.Substring(3, 2);
        Console.WriteLine("myString.Substring(3, 2) = " + myString22);
    }
}
```

5. Aşağıdaki deyimlerin sonunda `str` stringinin değeri nedir?

```
string str = "Sevimli küçük tavşan";
str += " tembel karabaşın üstünden atladı; ";
str += " ama karabaş farketmedi.";
```

6. Aşağıdaki kodlar çalışınca `str` ne olur?

```
string str = "Merhaba";
str.Replace ("m", "M");
```

7. Aşağıdaki kodlar çalışınca str ne olur?

```
string str = "Merhaba";
str.Replace ("M", "m");
```

8. Aşağıdaki kodlar çalışınca `sonuç` ne olur?

```
string kitap = "C# ile Nesne Programlama";
int sonuç;

sonuç = ad.Length;
sonuç = "Nesne Programlama".Length;
```

9. Aşağıdaki deyimlerin karşılarına sonuçlarını yazınız.

```
string str = "Sevimli küçük tavşan tembel karabaşın üstünden atladı";
int sonuç;

sonuç = str.IndexOf("küçük");           // sonuç :
sonuç = str.LastIndexOf("karabaş");     // sonuç :
sonuç = str.IndexOf("tembel");          // sonuç :
sonuç = str.LastIndexOf("üstü");        // sonuç :
```

10. Aşağıdaki deyimlerin karşılarına sonuçlarını yazınız.

```
string str = "Sevimli küçük tavşan tembel karabaşın sırtından atladı";
int sonuç;

sonuç = str.IndexOf("an");              // sonuç :
sonuç = str.IndexOf("an", 1);           // sonuç :
sonuç = str.IndexOf("an", 2);           // sonuç :
sonuç = str.LastIndexOf("an");          // sonuç :
sonuç = str.LastIndexOf("an",33);       // sonuç :
sonuç = str.LastIndexOf("an",32);       // sonuç :
```

11. Aşağıdaki deyimlerin karşılarına sonuçlarını yazınız.

```
string str = "Sevimli küçük tavşan tembel karabaşın sırtından atladı";
int sonuç;

sonuç = str.IndexOf("küçük");           // sonuç :
sonuç = str.IndexOf("tembel", 5, 4);    // sonuç :
sonuç = str.LastIndexOf("at");          // sonuç :
sonuç = str.LastIndexOf("sır", 25, 5);  // sonuç :
```

12. Aşağıdaki deyimlerin karşılarına sonuçlarını yazınız.

```
string str = "Sevimli küçük tavşan tembel karabaşın sırtından atladı";
bool sonuç;

sonuç = str.StartsWith("tembel karabaş"); // sonuç :
sonuç = str.StartsWith("sırtından");      // sonuç :
sonuç = str.EndsWith("Sevimli");          // sonuç :
sonuç = str.EndsWith("küçük tavşan");     // sonuç :
```

13. "Sevimli küçük tavşan tembel karabaşın sırtından atladı";

```
string str = "Sevimli küçük tavşan tembel karabaşın sırtından atladı";
bool sonuç;

sonuç = str.Contains("küçük tavşan");      // sonuç :
sonuç = str.Contains("tembel köpek");      // sonuç :
```

14. `String` ile `string` arasında fark olup olmadığını gösteren aşağıdaki programı çözümleyiniz.

```
using System;

class Uygulama
{
    public static void Main()
    {
        string s = "a";
        String S = "A";

        if (s.GetType() == S.GetType())
        {
        Console.WriteLine("string ve String aynı tiptir")
else
Console.WriteLine("string ve String aynı tip değildir");
        }
    }
}
```

15. Aşağıdaki programın kodlarını çıktı ile karşılaştırarak çözümleyiniz.

```
using System;

namespace Test
{
    class Program
    {
        static void Main(string[] args)
        {
            string test1 = "Merhaba";
            string test2 = "Dünya";

           System.Console.WriteLine(test1 + " " + test2);
            System.Console.WriteLine("Test1[0]: " + test1[0]);

            string[] test3 = new string[2];
            test3[0] = test1;
            test3[1] = test2;

            System.Console.WriteLine("Test3[0]: " + test3[0]);
            System.Console.WriteLine("Test3[1]: " + test3[1]);
            System.Console.WriteLine("Test3[0][0]: " + test3[0][0]);
        }
    }
}
```

Çıktı

```
Merhaba Dünya
Test1[0]: M
Test3[0]: Merhaba
Test3[1]: Dünya
Test3[0][0]: M
```