BTK AKADEMİ

Programlama - II

Doç. Dr. Zafer CÖMERT

BTK AKADEMİ

Bölüm 6

String Veri Yapısı

Giriş

İçerik

- String Veri Yapısı
- String ve Char İlişkisi
- String Oluşturma ve Temel İşlemler
- Hazır String Fonksiyonları
- String Karşılaştırma
- String Formatlama
- StringBuilder Sınıfı

String

- String, bir veya daha fazla karakterin (harf, rakam, sembol, boşluk) sıralı biçimde tutulduğu veri yapısıdır.

- Uygulamaların neredeyse tamamında metin işleme (kullanıcı girdisi, dosya adı, URL, JSON/XML, log mesajları vb.) kritik öneme sahiptir.

Referans tip

Unicode desteği

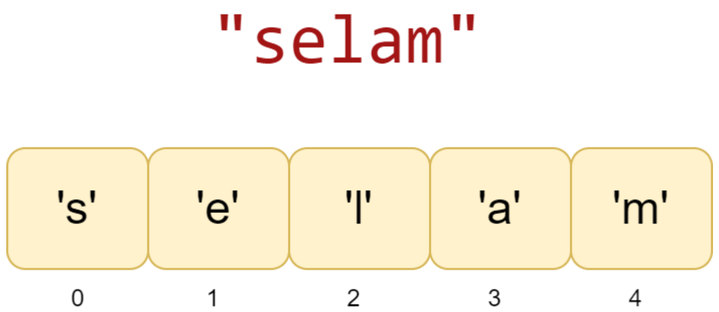
Immutable

Dizi benzeri erişim

Zengin metot desteği

String Pool (interning)

String ve Char İlişkisi



- char veri tipi tek bir karakteri saklamak için kullanılır.
- Unicode desteği sayesinde Türkçe karakterler dahil pek çok dilin sembollerini barındırabilir.
- String veri tipi ise birden fazla karakterin birleşiminden oluşur.

String Birleştirme Yöntemleri

Yöntem	Örnek Kod	Çıktı
+ operatörü	"Merhaba" + " " + "C#"	Merhaba C#
String.Concat()	String.Concat("Hoş", "geldiniz")	Hoşgeldiniz
Interpolation	$"Bugün {DateTime.Now.Year} yılı"	Bugün 2025 yılı

Kaçış Karakterleri

Kaçış Karakteri	Açıklama	Örnek Kullanım	Çıktı
\\	Ters eğik çizgi (backslash)	"C:\\Users\\Admin"	C:\Users\Admin
\'	Tek tırnak	'\''	'
\"	Çift tırnak	"Merhaba \"Ali\""	Merhaba "Ali"
\n	Yeni satır	"Merhaba\nDünya"	Merhaba Dünya
\t	Tab boşluğu	"Ad\tSoyad"	Ad Soyad
\r	Satır başı (Carriage return)	"Merhaba\rDünya"	Dünya
\b	Backspace (önceki karakteri siler)	"ABC\bD"	ABD
\0	Null karakter (boş karakter)	"A\0B"	AB (görünmez boş karakter içerir)
\uXXXX	Unicode karakter (4 haneli hex kod)	"\u03A9"	Ω
\xXX	Hexadecimal karakter	"\x41"	A

Verbatim String (@ Kullanımı)

- C#’ta uzun metinler veya dosya yolları yazarken @ sembolü kullanılır.
- Bu sayede kaçış karakterlerine gerek kalmaz.

```
1   string path = @"C:\Users\Student\Documents";
```

Yerleşik String Fonksiyonları

Üye	Amaç / Tipik Kullanım	Orijinali Değiştirir mi?	Örnek Kullanım	Dönüş
Length	Karakter sayısını öğrenme	Hayır (özellik okuma)	var n = s.Length;	int
Substring	Alt dize çıkarma	Hayır (yeni string)	s.Substring(2, 5)	string
IndexOf	Alt dize/karakter konumunu bulma	Hayır	s.IndexOf("@")	int (yoksa -1)
Replace	Desen ikame (eski→yeni)	Hayır (yeni string)	s.Replace("-", "")	string
ToUpper	Büyük harfe dönüştürme	Hayır (yeni string)	s.ToUpper()	string
ToLower	Küçük harfe dönüştürme	Hayır (yeni string)	s.ToLower()	string
Trim	Baştaki/sondaki boşlukları atma	Hayır (yeni string)	s.Trim()	string

String Karşılaştırma

==
C#’ta string için içerik eşitliği yapar (referans değil). Varsayılan davranış kültürden bağımsız ve büyük/küçük harfe duyarlı (ordinal, case-sensitive) bir karşılaştırmadır.

Equals()
Equals() metodu ise aşırı yüklemeleri sayesinde daha esnek bir kullanıma sahiptir; örneğin string.Equals(a, b, StringComparison.CurrentCultureIgnoreCase) biçiminde kültüre duyarlı ve/veya harf duyarsız seçenekler sunar.

CompareTo()
CompareTo() iki string’i karşılaştırır ve -1/0/1 döndürür. Döndürülen değer, sıralama ve karşılaştırma mantığının temelidir.

String Formatlama

Biçim	Açıklama	Örnek (en-US)	Örnek (tr-TR)
N2	Sayı, 2 ondalık	1,234.56	1.234,56
F2	Sabit nokta	1234.56	1234,56
C2	Para birimi	$1,234.56	₺1.234,56
P1	Yüzde (0–1 arası)	85.3 % (0.853)	85,3 %
D6	Sayıyı doldur (tam sayı)	000255	000255
X	Hex (tam sayı)	FF	FF

- Burada çok önemli bir nokta da kültür farklarıdır.
- Aynı sayı İngilizce (en-US) formatında 1,234.56 şeklinde yazılırken, Türkçe (tr-TR) formatında 1.234,56 olarak gösterilir.

Tarih Saat Formatlama

Biçim	Açıklama	Örnek (tr-TR)
d	Kısa tarih	17.08.2025
D	Uzun tarih	17 Ağustos 2025 Pazar
t	Kısa saat	14:35
T	Uzun saat	14:35:45
f	Uzun tarih + kısa saat	17 Ağustos 2025 Pazar 14:35
g	Kısa tarih + kısa saat	17.08.2025 14:35
o	Yuvarlatmasız ISO-8601	2025-08-17T14:35:45.0000000

- Burada özellikle kültür farkı devreye girer.
- Türkçe (tr-TR) kültüründe 17 Ağustos 2025 Pazar ifadesini görürken, İngilizce (en-US) kültüründe aynı tarih Sunday, August 17, 2025 şeklinde çıkar.

String Builder

- C#'ta string tipi immutabledır.
  - Bir kez oluşturulduktan sonra içerik değiştirilemez.
  - Bu yüzden döngü içinde çok sayıda + ile birleştirme yaptığınızda, her adımda yeni bir nesne üretilir ve ara kopyalar oluşur. Bu hem zaman hem de bellek açısından maliyeti artırır.

- StringBuilder, değiştirilebilir (mutable) bir karakter arabelleği kullanarak aynı işlemleri kopyalama yapmadan (amortize) gerçekleştirir.
  - Özellikle büyük metinler üretilirken, log veya rapor gibi adım adım büyüyen içeriklerde belirgin performans ve bellek kazanımı sağlar.

String Builder

Metod	Amaç	Kısa Not
Append(value)	Sona ekleme	Zincirleme çağrılabilir (Append(...).Append(...)).
Insert(index, value)	Araya ekleme	Geçerli aralıkta indeks ister.
Remove(startIndex, length)	Belirli kısmı silme	Arabellekten çıkarır, kaydırma yapar.
Replace(old, new)	Metin içinde dönüştürme	Tüm eşleşmeleri değiştirir (opsiyonel aralıkla).
Clear()	İçeriği temizler	Kapasiteyi korur.
ToString()	Son metni üretir	Çıktıyı alırken performanslı tek adım.

Teşekkürler

ZAFER CÖMERT
Öğretim Üyesi