

BBS515 Nesneye Yönelik Programlama

Ders 9 - String, StringBuffer, StringTokenizer

Zümra Kavafoğlu

<https://zumrakavafoglu.github.io/>

STRING NESNESİ OLUŞTURMA

- ❧ `String str1 = "This is a test!";`
`//literally`
- ❧ `String str1 = new String("This is a test!");`
`// Başka bir string nesnesinden`
- ❧ `String str3 = new String(charArray);`
`// Bir karakter dizisinden`
- ❧ `str1.length()` komutu, `str1` String nesnesini oluşturan karakter sayısını verir. (`str1` için 15'dir)

ALT STRINGLER (SUBSTRINGS)

❧ Bir altstring stringin bir parçasıdır. substring yöntemiyle bir stringin alt stringine ulaşılabilir. Bu yöntemin iki çeşidi vardır:

❧ **s.substring(int st)**

Bu yöntem argüman olarak sadece alt stringin başlangıç indeksini alır ve bu indeksten başlayıp stringin sonuna kadar olan bütün karakterleri kapsayan yeni bir String oluşturur.

❧ **s.substring(int st , int en)**

Bu yöntem ise alt stringin başlangıç ve bitiş indekslerini argüman olarak alır. Eğer bu argümanlar 0 dan küçük ya da orijinal dizinin boyundan büyükse program *StringIndexOutOfBoundsException* hatası verir.

ÖNEMLİ NOT



☞ substring yönteminde dikkat edilmesi gereken bazı durumlar vardır:

☞ Javada her zaman olduğu gibi String nesnesi için de indeks değeri 0 dan başlar. Dolayısıyla yöntemde st değeri 0 olarak girilirse String nesnesinin ilk karakteri döner.

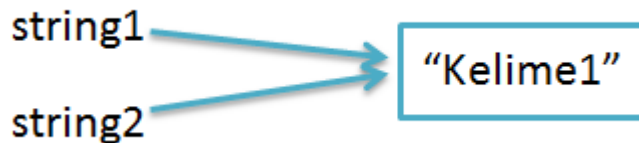
String Karşılaştırma



String karşılaştırma:

$string1 == string2$ ifadesi $string1$ ve $string2$ 'nin aynı nesneyi gösterip göstermediğini kontrol eder, $string1$ ile $string2$ 'nin aynı içeriğe sahip olup olmadığını kontrol etmez.

```
String string1 = new String("Kelime1");  
String string2 = string1;
```



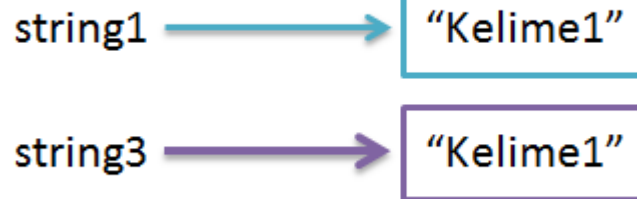
$string1 == string2$

TRUE

String Karşılaştırma



```
String string1 = new String("Kelime1");  
String string3 = new String("Kelime1");
```



`string1 == string3` → FALSE

String Karşılaştırma



- ❧ İki String nesnesinin içeriklerini karşılaştırmak için equals ya da compareTo metotları kullanılmalıdır.
- ❧ equals metodu, iki stringin içeriği birbirine eşitse true, değilse false döndürür.

```
String string1 = new String("Kelime1");  
String string3 = new String("Kelime1");
```

string1.equals(string3)



TRUE

String Karşılaştırma

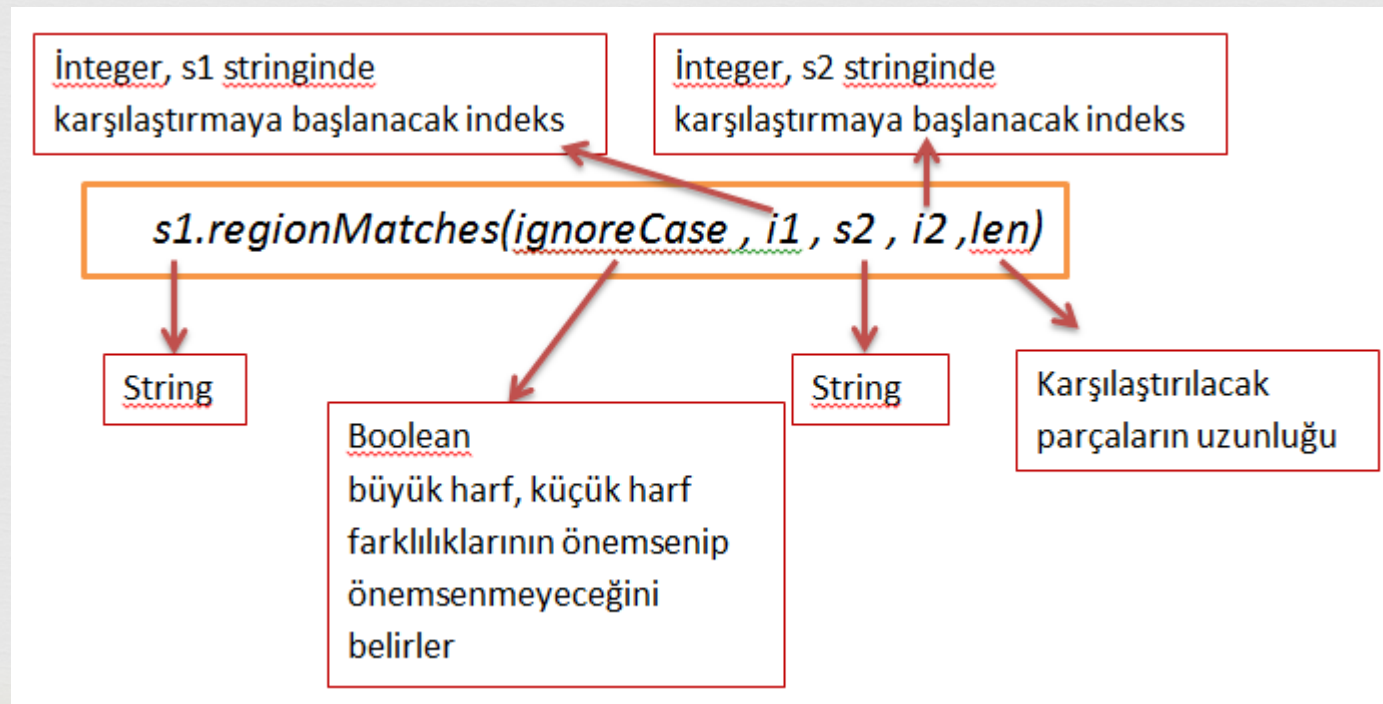


- ❧ s1 ve s2 iki String olmak üzere, s1.compareTo(s2) metodu
 - ❧ s1 ile s2 eşitse 0 döndürür.
 - ❧ s1, s2'den lexicographically (ascii kodlarına göre sözlük sıralaması) küçükse negatif bir değer döndürür. (İlk farklı karaktere kadar olan karakter sayısının eksilisi)
 - ❧ s1, s2'den lexicographically (ascii kodlarına göre sözlük sıralaması) büyükse pozitif bir değer döndürür. (İlk farklı karaktere kadar olan karakter sayısı)
- ❧ **String nesnelerini <, >, <=, >= gibi karşılaştırma operatörleriyle karşılaştırmak bir yazım hatasıdır.**

String Karşılaştırma



☞ *regionMatches* yöntemi iki string nesnesinin belirlenen bölgelerini karşılaştırır.



String Karşılaştırma



☞ *equalsIgnoreCase* yöntemi *equals* yöntemine benzerdir. Ancak bu yöntem karşılaştırma yaparken harflerin büyük ya da küçük olmasını dikkate almaz.

```
String string1 = new String("Kelime1");  
String string3 = new String("kElimE1");
```

string1.equals(string3)

FALSE

string1.equalsIgnoreCase(string3)

TRUE

String Birleştirme



- ❧ İki string concat yöntemini kullanarak birleştirilebilir.
- ❧ concat yöntemi, parametre olarak bir String nesnesi alır ve dönüş tipi de yine String'dir.
- ❧ `String s3 = s1.concat (s2)` komutu, s2 stringinin karakterlerini s1 stringinin sonuna ekleyerek yeni bir s3 stringi oluşturur. s1 ve s2 stringleri bu işlemde etkilenmez.
- ❧ String birleştirme direkt + operatörü kullanılarak da yapılabilir.
 - ❧ `String s3 = s1 + s2;`

indexOf, lastIndexOf



- ❧ Bir alt stringin başka bir stringin içinde yer alıp almadığına bakmak için *indexOf* ve *lastIndexOf* yöntemleri kullanılır.
- ❧ *indexOf* yöntemi aşağıdaki şekillerde kullanılabilir:
- ❧ *s1.indexOf(String s2) ;*
- ❧ *s1.indexOf(String s2 , int start) ;*
- ❧ *s1.indexOf(char c) ;*
- ❧ *s1.indexOf(char c , int start) ;*
- ❧ Bu yöntem belirtilen başlangıç indeksinden (start) başlayarak(eğer belirtilmediyse 0 olarak alınır) s1 stringini soldan sağa doğru tarar. s2 alt strinigini ya da c karakterini ilk bulduğu indeksi döndürür , eğer bulamadıysa -1 döndürür.

indexOf, lastIndexOf



- ❧ *lastIndexOf* yöntemi ise aşağıdaki şekillerde kullanılabilir:
- ❧ *s1.lastIndexOf(String s2) ;*
- ❧ *s1.lastIndexOf(String s2 , int start) ;*
- ❧ *s1.lastIndexOf(char c) ;*
- ❧ *s1.lastIndexOf(char c , int start) ;*
- ❧ Bu yöntem belirtilen başlangıç indeksinden başlayarak(eğer belirtilmediyse stringin sonundan başlanır) s1 stringini sağdan sola doğru tarar. s2 alt stringini ya da c karakterini ilk bulduğu indeksi döndürür , eğer bulamadıysa -1 döndürür.

replace metodu



- ❧ replace yöntemi stringdeki belirlenen karakteri yeni bir karakterle değiştiren yöntemdir.
- ❧ Yöntem iki karakteri parametre olarak alır ve dönüş tipi String'dir.
- ❧ Yöntem `s1.replace(old , new)` biçimindedir. Burada `s1` bir string , `old` değiştirilmesi istenen karakter , `new` ise istenen karakterin yerine konulacak yeni karakterdir.
- ❧ Örneğin : `s1 = "This is a test."` olsun.
 - ❧ O halde `s1.replace('i' , 'I')` sonucunda `"ThIs Is a test"` string nesnesi döndürülecek.

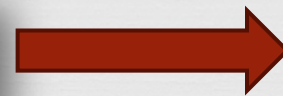
charAt metodu



☞ `char charAt(int index)` metodu parametre olarak bir indeks alır ve `String` nesnesinin o indeksteki karakterini döndürür.

```
String s = "abcdefg";
```

```
c = s.charAt(3);  
System.out.println(c);
```



d

```
char c = s.charAt(0);  
System.out.println(c);
```



a

String nesnesi değişmezdir (immutable)



- ❧ Bahsettiğimiz bütün bu String sınıfı metotlarının ortak özelliği ne?
- ❧ Hiç biri birlikte çağırıldığı String nesnesini(nesnelerini) değiştirmiyor, gerekli değişiklikler yapılmış yeni bir String nesnesi oluşturuyor.

String nesnesi değişmezdir (immutable)

```
String s = "Hello ";  
s+="World";
```

Bu komutların sonucunda başlangıçta oluşturulan s nesnesinin üzerinde değişiklik yapılmaz. Başlangıçtaki «Hello» değerli s nesnesi silinir «Hello World» değerli yeni bir s nesnesi oluşturulur.

Çünkü Stringler aslında «immutable» yani «değişmez» dirler!!!

StringBuffer sınıfı

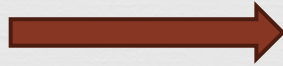


- ❧ Bu durumda bir String nesnesine çok sayıda ardarda değer eklemek hiç de etkili bir işlem değildir.
- ❧ Bunun için StringBuffer sınıfı vardır!!!
- ❧ StringBuffer değiştirilebilir(yani mutable) bir String gibi düşünülebilir.

length() ve capacity()

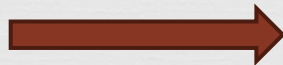


```
public int length()
```



StringBuffer
nesnesindeki
karakter sayısını
döndürür

```
public int capacity()
```



StringBuffer nesnesinin
hafızada yeni yer
açılmadan sahip olabileceği
maksimum karakter
sayısını döndürür.
(performans amaçlı bir
özelliktir.)

StringBuffer yapılandırıcıları



```
public StringBuffer()
```



16 karakter kapasiteli ve hiç karakter içermeyen bir string buffer nesnesi oluşturur.

```
public StringBuffer(int length)
```



length kapasiteli ve hiç karakter içermeyen bir string buffer nesnesi oluşturur.

```
public StringBuffer(String s)
```



s String nesnesinden bir StringBuffer nesnesi oluşturur. Kapasitesi s nesnesinin uzunluğu + 16'dır.

Append metodu



☞ StringBuffer sınıfının, nesnesinin sonuna boolean, char, char dizisi, double, float, int, long, String eklemek için overloaded *append* yöntemleri vardır.

```
StringBuffer strBuf = new StringBuffer();
```

```
strBuf.append("Welcome");  
strBuf.append(" ");  
strBuf.append("to");  
strBuf.append(" ");  
strBuf.append("Java");
```

Welcome

Welcome to

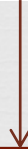
Welcome to Java

Insert metodu



- StringBuffer sınıfının, nesnesinin içine boolean, char, char dizisi, double, float, int, long, String eklemek için overloaded *insert* yöntemleri vardır.

```
strBuf.insert(11, "magnificent ");
```



Welcome to magnificent Java

StringTokenizer sınıfı



- ❧ StringTokenizer sınıfı, bir String nesnesini parçalara bölmek için kullanılan sınıftır.
- ❧ StringTokenizer ile oluşturulan parçaların her birine *token* denir.
- ❧ Stringin hangi ayrıca göre ayrılacağı da StringTokenizer ile belirlenebilir.

StringTokenizer yapılandırıcıları



```
public StringTokenizer(String s, String delim, boolean returnTokens)
```



s – Parçalanacak String
delim – ayraçların bir listesi
returnTokens – ayraçların da bir token olarak
döndürülmesini belirler

```
public StringTokenizer(String s, String delim)
```



returnTokens default olarak false olur.

StringTokenizer yapılandırıcıları



```
public StringTokenizer(String s)
```



returnTokens default olarak false olur.
delimiter default olarak « \t\n\r » stringi
alınır.

StringTokenizer metotları



- ❧ `public boolean hasMoreTokens()`
 - ❧ Eğer stringde token kaldıysa true döndürür.
- ❧ `public String nextToken()`
 - ❧ Stringdeki bir sonraki tokenı döndürür.
- ❧ `public String nextToken(String delim)`
 - ❧ delim ayracına göre Stringdeki bir sonraki tokenı döndürür.
- ❧ `public int countTokens()`
 - ❧ Stringde kalan tokenların sayısını döndürür.

DÖNÜŞTÜRÜCÜLER

Wrapper Class	Method	Description	Example	Returned Value
Integer	<code>parseInt(string)</code>	Converts a string to a primitive type <code>int</code> .	<code>Integer.parseInt("1234")</code>	1234 (an <code>int</code> value)
Integer	<code>toString(x)</code>	Converts the primitive <code>int x</code> value to a string object.	<code>Integer.toString(345)</code>	"345" (a string object)
Long	<code>parseLong(string)</code>	Converts a string to a primitive type <code>long</code> .	<code>Long.parseLong("128365489")</code>	128365489L (a <code>long</code>)
Long	<code>toString(x)</code>	Converts the primitive <code>long x</code> value to a string object.	<code>Long.toString(128365489)</code>	"128365489" (a string object)
Float	<code>parseFloat(string)</code>	Converts a string to a primitive type <code>float</code> .	<code>Float.parseFloat("345.89")</code>	345.89f (a <code>float</code> value)
Float	<code>toString(x)</code>	Converts the primitive <code>float x</code> value to a string object.	<code>Float.toString(345.873)</code>	"345.873" (a string object)
Double	<code>parseDouble(string)</code>	Converts a string to a primitive type <code>double</code> .	<code>Double.parseDouble("2.3456789")</code>	2.3456789 (a <code>double</code> value)
Double	<code>toString(x)</code>	Converts the primitive <code>double x</code> value to a string object.	<code>Double.toString(345.873)</code>	"345.873" (a string object)