Bölüm 10

10. Karakter Sınıfları

Normalde karakterlere uğraşırken ilkel veri tipi olan char kullanılır.

Örnek:

```
char ch = 'a';

// Unicode for uppercase Greek omega character
char uniChar = '\u039A';

// an array of chars
char[] charArray ={ 'a', 'b', 'c', 'd', 'e' };
```

Ancak geliştirme sırasında ilkel veri tipleri yerine objeleri kullanmamız gereken durumlarla karşılaşabiliriz. Bu durumda Java char için **Character** sınıfını kullanmamızı sağlar.

Character sınıfı karakterleri değiştirip kullanabilmek için bir çok sayıda metot sağlar.

```
Character ch = new Character('a');
```

10.1 Character Metotları

Burada Character sınıfının; bütün alt sınıflarının kulllandığı önemli instance metotları belirtilmiştir.

Methods with Description	Example
isLetter() Belirtilen değerin harf olup olmadığına bakar	`public class Test {` `public static void main(String args[]) {` `System.out.println(Character.isLetter('c'));` `System.out.println(Character.isLetter('5'));` `}` `}` Şu sonuç oluşur: `true`

	`` false ```
isDigit() Belirtilen değerin sayı olup olmadığına bakar.	``` public class Test { public static void main(String args[]) { System.out.println(Character.isDigit('c')); System.out.println(Character.isDigit('5')); } } ``` Şu sonuç oluşur: ``` false true ```
isWhitespace() Bu metot belirtilen char değerlerinin boşluk içerip içermediğine bakar. İncelemeyi; • Boşluk(Space) • Tab • Yeni Satır(new line) Baz alarak yürütür.	``` public class Test{ public static void main(String args[]){ System.out.println(Character.isWhitespace('c')); System.out.println(Character.isWhitespace(' ')); System.out.println(Character.isWhitespace('\n')); System.out.println(Character.isWhitespace('\t')); } } ``` Şu sonuç oluşur: ``` false true true true ```
isUpperCase() Belirtilen char değerinin büyük harf olup olmadığına bakar.	``` public class Test{ public static void main(String args[]){ System.out.println(Character.isUpperCase('c')); System.out.println(Character.isUpperCase('C')); System.out.println(Character.isUpperCase('\n')); System.out.println(Character.isUpperCase('\t')); } } ``` Şu sonuç oluşur: ``` false true false false ```
isLowerCase() Belirtilen char değerinin küçük harf olup olmadığına bakar	``` public class Test{ public static void main(String args[]){ System.out.println(Character.isLowerCase('c')); System.out.println(Character.isLowerCase('C')); System.out.println(Character.isLowerCase('\n')); System.out.println(Character.isLowerCase('\t')); } } ```

	Şu sonuç oluşur: `true` `false` `false` `false`
toUpperCase() Verilen char değerlerini büyük harfe çevirir.	`public class Test{` `public static void main(String args[]){` `System.out.println(Character.toUpperCase('c'));` `System.out.println(Character.toUpperCase('C'));` `}` `}` Şu sonuç oluşur: `C` `C`
toLowerCase() Verilen char değerlerini küçük harfe çevirir.	`public class Test{` `public static void main(String args[]){` `System.out.println(Character.toLowerCase('c'));` `System.out.println(Character.toLowerCase('C'));` `}` `}` Şu sonuç oluşur: `c` `c`
toString() Character değerini String nesnesine çevirir.	`public class Test{` `public static void main(String args[]){` `System.out.println(Character.toString('c'));` `System.out.println(Character.toString('C'));` `}` `}` This produces following result: `c` `C`

Bölüm 11

11. String'ler

Java'da geniş olarak kullanılan String'ler, karakter dizileridir.Java programlama dilinde; String'ler nesnedir.

Java platformu String'leri oluşturmak ve kullanmak için String sınıfı bulundurmaktadır

11.1 String Oluşturma

String oluşturmanın en direkt yolu şu şekildedir:

```
String greeting = "Hello world!";
```

Derleyici kodun içinde String deyimiyle karşılaştığı anda bir String nesnesi yaratır. Bu örnekte bu nesne "Hello world!"

Diğer nesnelerde olduğu gibi, Stringleri de "new" anahtar kelimesini ve constructor kullanarak oluşturabilirsiniz. String sınıfının; ilk değerlerinin atanabilmesi için 11 tane farklı constructor'ı vardır.Örneğin karakterlerden oluşan bir dizi.

```
public class StringDemo{

   public static void main(String args[]){
      char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.'};
      String helloString = new String(helloArray);
      System.out.println( helloString );
   }
}
```

Bu aşağıdaki sonucu üretecektir:

```
hello
```

11.2 String Uzunluğu

Bir nesneyle ilgili bilgi elde etmek isteyen metotlara erişimci metotlar denir. Stringle kullanabileceğiniz erişimci metotlardan birisi length(). Bu metot string nesnesinde kaç karakter olduğunu bulmaktadır.

Aşağıdaki kod çalıştırıldığında uzunluk (len) 17'ye eşit olacaktır.

```
public class StringDemo {

   public static void main(String args[]) {
      String palindrome = "Dot saw I was Tod";
      int len = palindrome.length();
      System.out.println( "String Length is : " + len );
   }
}
```

Bu aşağıdaki sonucu üretecektir.

```
String Length is : 17
```

11.3 String'leri Birleştirmek

String sınıfı; iki stringi birleştirmek için bir metot kullanmaktadır.

```
string1.concat(string2);
```

String2'nin string1'in arkasına eklenerek oluşturduğu yeni bir string dönecektir.Aşağıda olduğu gibi concat() metodunu string dizilerinde kullanabilirsiniz.

```
"My name is ".concat("Zara");
```

Stringler genel olarak "+" operatörüyle birbirlerine bağlanırlar

```
"Hello," + " world" + "!"
```

Sonuç olarak;

```
"Hello, world!"
```

Aşağıdaki örneği inceleyelim.

```
public class StringDemo {

   public static void main(String args[]) {
      String string1 = "saw I was ";
      System.out.println("Dot " + string1 + "Tod");
   }
}
```

Bu aşağıdaki sonucu üretecektir.

```
Dot saw I was Tod
```

11.4 String Methotları

Methods with Description	Example
char charAt(int index) Belirtilen index değerindeki karakteri döndürür. String indexleri 0'dan başlar.	`public class Test {` `public static void main(String args[]) {` `String s = "Strings are immutable";` `char result = s.charAt(8);` `System.out.println(result);` `}` `}` Bu aşağıdaki sonucu oluşturacaktır. `A`
int compareTo(Object o) Compares this String to another Object. Bir stringi diğer objelerle karşılaştırır.Eğer çıkan sonuç 0 ise karşılaştırılan nesneler birbirine eşittir. Eğer 0'dan küçükse karşılaştırılan değer sözlüksel olarak karşılaştırılmak istenenden küçüktür. Eğer 0'dan büyükse karşılaştırılan değer sözlüksel olarak karşılaştırılmak istenenden büyüktür..	`public class Test {` `public static void main(String args[]) {` `String str1 = "Strings are immutable";` `String str2 = "Strings are immutable";` `String str3 = "Integers are not immutable";` `int result = str1.compareTo( str2 );` `System.out.println(result);` `result = str2.compareTo( str3 );` `System.out.println(result);` `result = str3.compareTo( str1 );` `System.out.println(result);` `}` `}` Bu aşağıdaki sonucu oluşturacaktır. `0` `10`

	``-10```
int compareToIgnoreCase(String str) Büyük küçük harf ayrımı yapmadan; iki stringi sözlüksel olarak karşılaştırır.	```public class Test { public static void main(String args[]) { String str1 = "Maltepe"; String str2 = "maltepe"; int result = str1.compareTo(str2); System.out.println(result); result = str1.compareToIgnoreCase(str2); System.out.println(result); } }``` Bu aşağıdaki sonucu oluşturacaktır. ```-32 0```
String concat(String str) Bu metot iki stringi birbirine bağlar.	```public class Test { public static void main(String args[]) { String s = "Strings are immutable"; s = s.concat(" all the time"); System.out.println(s); } }``` Bu aşağıdaki sonucu oluşturacaktır. ```Strings are immutable all the time```
boolean endsWith(String suffix) Stringin belirtilen değerle bitip bitmediğine bakar.	```public class Test{ public static void main(String args[]){ String Str = new String("This is immutable"); boolean retVal; retVal = Str.endsWith("immutable"); System.out.println(retVal); retVal = Str.endsWith("immu"); System.out.println(retVal); } }``` Bu aşağıdaki sonucu oluşturacaktır. ```true false```
boolean equals(Object anObject) Bir stringi belirtilen bir nesneyle karşılaştırır.	```public class Test { public static void main(String args[]) { String Str1 = new String("maltepe"); String Str2 = Str1; String Str3 = new String("maltepe"); boolean retVal;```

	`` retVal = Str1.equals(Str2); System.out.println("Returned = " + retVal); retVal = Str1.equals(Str3); System.out.println("Returned = " + retVal); } } ``` Bu aşağıdaki sonucu oluşturacaktır. ``` Returned = true Returned = true ```
boolean equalsIgnoreCase(String anotherString) Büyük küçük ayrımı yapmadan iki stringi birbiriyle karşılaştırır.	Modify the example given in the equals method description as shown below. ``` String Str4 = new String("Maltepe"); retVal = Str1.equalsIgnoreCase(Str4); System.out.println("Returned = " + retVal); ```
int length() String'in karakter uzunluğunu döndürür.	``` public class Test{ public static void main(String args[]){ String Str2 = new String("Tutorials"); System.out.print("String Length :"); System.out.println(Str2.length()); } } ``` Bu aşağıdaki sonucu oluşturacaktır. ``` String Length :9 ```
String replace(char oldChar, char newChar) oldChar karakterlerinin yerine newChar karakterlerini koyarak Stringi baştan yazar.	``` public class Test{ public static void main(String args[]){ String Str =new String("Maltepe University"); System.out.print("Return Value :"); System.out.println(Str.replace('e', 'i')); } } ``` Bu aşağıdaki sonucu oluşturacaktır. ``` Return Value :Maltipi Univirsity ```
String toLowerCase() Yerel kurallara göre; stringteki bütün harfleri küçük harfe çevirecektir.	``` public class Test{ public static void main(String args[]){ String Str =new String("Maltepe University"); System.out.print("Return Value :"); System.out.println(Str.toLowerCase()); } } ``` Bu aşağıdaki sonucu oluşturacaktır.

	`` Return Value :maltepe university ```
String toLowerCase(Locale locale) Belirtilen dil ailesinin kurallarına göre verilen stringteki bütün harfleri küçük harfe çevirecektir.	``` public class Test{ public static void main(String args[]){ String Str = "ILLAKI"; System.out.println(Str.toLowerCase(new Locale("en"))); System.out.println(Str.toLowerCase()); } } ``` Bu aşağıdaki sonucu oluşturacaktır. ``` illaki ıllakı ```
String toUpperCase() Yerel kurallara göre; stringteki bütün harfleri büyük harfe çevirecektir.	``` public class Test{ public static void main(String args[]){ String Str = new String("Welcome to Java"); System.out.print("Return Value :"); System.out.println(Str.toUpperCase()); } } ``` Bu aşağıdaki sonucu oluşturacaktır. ``` Return Value :WELCOME TO JAVA ```
String toUpperCase(Locale locale) Belirtilen dil ailesinin kurallarına göre verilen stringteki bütün harfleri büyük harfe çevirecektir.	``` public class Test{ public static void main(String args[]){ String Str = "illaki"; System.out.println(Str.toUpperCase(new Locale("en"))); System.out.println(Str.toUpperCase()); } } ``` Bu aşağıdaki sonucu oluşturacaktır. ``` ILLAKI İLLAKİ ```
String trim() Stringin başındaki ve sonundaki boşlukları keserek ekrana basar.	``` public class Test{ public static void main(String args[]){ String Str = new String(" Welcome to Tutorials "); System.out.print("Return Value :"); System.out.println(Str.trim()); } } ```

	Bu aşağıdaki sonucu oluşturacaktır. `Return Value :Welcome to Tutorials`

11.4.1 indexOf()

Bu metodun farklı değişkenleri mevcuttur.

- **public int indexOf(int ch):** Belirtilen karakterin ilk görüldüğü yerdeki index numarasını döndürür. Eğer o karakter hiç kullanılmadıysa -1 döner.
- **public int indexOf(int ch, int fromIndex):** Belirtilen indexten başlayarak bir karakterin ilk görüldüğü yerdeki index numarasını döndürür. Eğer o karakter hiç kullanılmadıysa -1 döner.
- **int indexOf(String str):** Belirtilen string parçasının ilk görüldüğü yerdeki index numarasını döndürür. Eğer o karakter hiç kullanılmadıysa -1 döner.
- **int indexOf(String str, int fromIndex):** Belirtilen indexten başlayarak bir string parçasının ilk görüldüğü yerdeki index numarasını döndürür. Eğer o karakter hiç kullanılmadıysa -1 döner.

  Bu metodun yazımı şu şekildedir.

```
public int indexOf(int ch )

public int indexOf(int ch, int fromIndex)

int indexOf(String str)

int indexOf(String str, int fromIndex)
```

Parametrelerin detayları şu şekildedir;

- **ch** – Bir karakter
- **fromIndex** – Aramaya başlanacak olan index.
- **str** – Bir string

Örnek:

```
public class Test {

   public static void main(String args[]) {
```

```
      String Str = new String("Welcome to Tutorials about Java");
      String SubStr1 = new String("Tutorials");
      String SubStr2 = new String("Sutorials");

      System.out.print("Found Index :" );
      System.out.println(Str.indexOf( 'o' ));
      System.out.print("Found Index :" );
      System.out.println(Str.indexOf( 'o', 5 ));
      System.out.print("Found Index :" );
      System.out.println( Str.indexOf( SubStr1 ));
      System.out.print("Found Index :" );
      System.out.println( Str.indexOf( SubStr1, 15 ));
      System.out.print("Found Index :" );
      System.out.println(Str.indexOf( SubStr2 ));
   }
}
```

Bu aşağıdaki değeri döndürecektir:

```
Found Index :4
Found Index :9
Found Index :11
Found Index :-1
Found Index :-1
```

11.4.2 split()

Bu metodun iki değişkeni vardır ve verilen ifade kurallarına göre stringi belirtildiği şekilde ayırır.

Yazımı şu şekildedir;

```
public String[] split(String regex, int limit)

public String[] split(String regex)
```

Parametrelerin detayları ise;

- **regex** – sınırlandırıcı ifade kuralı
- **limit** – Sonuç sınırı.Yani; ne kadar stringin geri döneceği belirtilir.

Örnek:

```
public class Test{
   public static void main(String args[]){
      String Str = new String("Jim-Jack-Marry-David");

      System.out.println("Return Value :" );
      for (String retval: Str.split("-", 2)){
         System.out.println(retval);
      }
      System.out.println("");
      System.out.println("Return Value :" );
```

```
      for (String retval: Str.split("-", 3)){
         System.out.println(retval);
      }
      System.out.println("");
      System.out.println("Return Value :" );
      for (String retval: Str.split("-", 0)){
         System.out.println(retval);
      }
      System.out.println("");
      System.out.println("Return Value :" );
      for (String retval: Str.split("-")){
         System.out.println(retval);
      }
   }
}
```

Bu aşağıdaki sonucu oluşturur.

```
Return Value :
Jim
Jack-Marry-David

Return Value :
Jim
Jack
Marry-David

Return Value :
Jim
Jack
Marry
David

Return Value :
Jim
Jack
Marry
David
```

11.4.3 startsWith()

Bu metodun iki tane değişkeni vardır. Belirtilen indexin belirtilen şekilde başlayıp başlamadığını veya index göstermeden; belirtilen şekilde başlayıp başlamadığını kontrol eder.

Metodun yazımı aşağıdaki gibidir.

```
public boolean startsWith(String prefix, int toffset)

public boolean startsWith(String prefix)
```

Parametrelerin detayları;

- **prefix** – Uyuşacak parça.

- **toffset** – Stringin içinde aramaya başlanacak yer.

Örnek:

```
import java.io.*;

public class Test{
   public static void main(String args[]){
      String Str = new String("Welcome to Tutorials");

      System.out.print("Return Value :" );
      System.out.println(Str.startsWith("Welcome") );

      System.out.print("Return Value :" );
      System.out.println(Str.startsWith("Tutorials") );

      System.out.print("Return Value :" );
      System.out.println(Str.startsWith("Tutorials", 11) );
   }
}
```

Bu aşağıdaki kodu üretecektir;

```
Return Value :true
Return Value :false
Return Value :true
```

11.4.3 subSequence()

Bu metot; bir karakter dizisinin içinden çekilen yeni bir karakter dizisi oluşturur.

Metodun yazımı şu şekildedir.

```
public CharSequence subSequence(int beginIndex, int endIndex)
```

Parametre detayları;

- **beginIndex** – Başlangıç indexi, dahildir.
- **endIndex** – Bitiş indexi, dahil değildir.

Örnek:

```
public class Test{
   public static void main(String args[]){
      String Str = new String("Welcome to Tutorials");

      System.out.print("Return Value :" );
```

```
      System.out.println(Str.subSequence(0, 10) );

      System.out.print("Return Value :" );
      System.out.println(Str.subSequence(10, 15) );
   }
}
```

Bu aşağıdaki sonucu verecektir.

```
Return Value :Welcome to
Return Value : Tuto
```