برنامه سازی پیشرفته

کار با رشته‌ها

کلاس String متدهای کاربردی بسیاری دارد که در اینجا چندتا از آنها را بررسی خواهیم کرد.

`charAt(int i)`

محتویات کاراکتر موجود در اندیس i ام رشته.

```
//charAt,length
String a = "hello world";
for(int i=a.length()-1;i>=0;i--){
    System.out.print(a.charAt(i));
}
```

```
dlrow olleh
```

`equals(String another)`

در صورت تساوی محتویات دو رشته مقدار true و در غیر اینصورت مقدار false را بر می‌گرداند.

`equalsIgnoreCase(String another)`

بزرگی و کوچکی حروف در مقایسه دو رشته تاثیری ندارد.

```
//equals vs equalsIgnoreCase
String a = "hello world";
String b = "Hello World";
String c = "Hello";
System.out.println("a.equals(b): " + a.equals(b));
System.out.println("a.equalsIgnoreCase(b): " + a.equalsIgnoreCase(b));
System.out.println("b.equalsIgnoreCase(c): " + b.equalsIgnoreCase(c));
```

```
a.equals(b): false
a.equalsIgnoreCase(b): true
b.equalsIgnoreCase(c): false
```

compareTo(String another)

دو رشته را با یکدیگر مقایسه می کند اگر رشته داده شده در آرگمان بزرگتر باشد (مبنای مقایسه به همان شکلی است که در هنگام مرتب سازی لیست اسامی انجام می شود) عدد منفی اگر هردو مقدار مساوی باشند صفر و اگر آرگمان ورودی کوچکتر از رشته ما باشد عدد مثبت برمی‌گردد.

compareToIgnoreCase(String another)

کوچکی و بزرگی حروف در مقایسه بدون تاثیر می‌شود.

```
//compareTo vs compareToIgnoreCase
String a = "hello world";
String b = "Hello World";
String c = "Hello";
System.out.println("a.compareTo(b): " + a.compareTo(b));
System.out.println("a.compareToIgnoreCase(b): " + a.compareToIgnoreCase(b));
System.out.println("b.compareToIgnoreCase(c): " + b.compareToIgnoreCase(c));
System.out.println("c.compareToIgnoreCase(b): " + c.compareToIgnoreCase(b));
```

```
a.compareTo(b): 32
a.compareToIgnoreCase(b): 0
b.compareToIgnoreCase(c): 6
c.compareToIgnoreCase(b): -6
```

split(String regex)

رشته را بر اساس الگوی تعیین شده می‌شکاند و به آرایه ای از رشته ها تبدیل می‌کند.

```
//split
String a = "i love java. java loves me."
System.out.println("Split by space: " + Arrays.toString(a.split(" ")));
System.out.println("Split by java: " + Arrays.toString(a.split("java")));
```

```
Split by space: [i, love, java., java, loves, me.]
Split by java: [i love , . ,  loves me.]
```

concat(String another)

یک رشته را به رشته‌ای دیگر می‌چسباند و رشته‌ای جدید را بر می‌گرداند.

```
//concat
String a = "Hello";
String b = "World";
System.out.println("a.concat(b): "+a.concat(" ").concat(b));
```

```
a.concat(b): Hello World
```

trim()

حروف سفید (مثل اسپیس و تب) اضافی در ابتدا و انتهای رشته را حذف می‌کند.

```
//trim
String beginSpace = "     java";
String trailSpace = "java     ";
String spaces = "  j  a  v  a  ";
System.out.println("beginSpace.trim(): \""+beginSpace.trim()+"\"");
System.out.println("trailSpace.trim(): \""+trailSpace.trim() +"\"");
System.out.println("spaces.trim(): \""+spaces.trim()+"\"");
```

```
beginSpace.trim(): "java"
trailSpace.trim(): "java"
spaces.trim(): "j  a  v  a"
```

toCharArray()

آرایه‌ای شامل محتویات رشته را بر می‌گرداند. توجه شود که با اعمال تغییر در این آرایه رشته اصلی تغییری نمی‌کند.

```
//toCharArray
String a = "abcdefg";
char[] aAsArray = a.toCharArray();
for(int i=0;i<aAsArray.length;i++){
    aAsArray[i]+=1;
}
System.out.println("result: "+ new String(aAsArray));
```

```
result: bcdefgh
```

toUpperCase(), toLowerCase()

همانطور که از اسم آن پیدا است حروف را تماما کوچک یا تماما بزرگ می‌کند.

```
//toUpperCase, toLowerCase
String a = "java";
System.out.println("a.toUpperCase(): "+ a.toUpperCase());
System.out.println("a.toLowerCase(): "+ a.toLowerCase());
```

```
a.toUpperCase(): JAVA
a.toLowerCase(): java
```

```
startsWith(String prefix, int fromIndex),
startsWith(String prefix)
endsWith(String prefix , int fromIndex),
endsWith(String prefix)
```

آیا رشته (از اندیس مشخص شده) با حرف مشخص شده شروع شده؟

آیا رشته (از اندیس مشخص شده به سمت چپ) با حرف مشخص شده تمام شده؟

```
//startsWith, endsWith
String a = "i love java. java loves me. java is awesome.";
//          01234567
System.out.println("a.startsWith(\"java\"): "+ a.startsWith("java"));
System.out.println("a.startsWith(\"java\",6): "+ a.startsWith("java",6));
System.out.println("a.startsWith(\"java\",7): "+ a.startsWith("java",7));
System.out.println("a.endsWith(\"java\"): "+ a.endsWith("java"));
System.out.println("a.endsWith(\"java\"): "+ a.endsWith("."));
```

```
a.startsWith("java"): false
a.startsWith("java",6): false
a.startsWith("java",7): true
a.endsWith("java"): false
a.endsWith("java"): true
```

```
indexOf(String prefix),
indexOf(String prefix, int fromIndex),
lastIndexOf(String prefix)
lastIndexOf(String prefix, int fromIndex)
```

```java
//indexOf, lastIndexOf
String a = "i love java. java loves me. java is awesome.";
//          012345678
System.out.println("a.indexOf(\"java\"): "+ a.indexOf("java"));
System.out.println("a.indexOf(\"java\",7): "+ a.indexOf("java",7));
System.out.println("a.indexOf(\"java\",8): "+ a.indexOf("java",8));
System.out.println("a.lastIndexOf(\"java\"): "+ a.lastIndexOf("java"));
System.out.println("a.lastIndexOf(\"java\",28): "+ a.lastIndexOf("java",28));
System.out.println("a.lastIndexOf(\"java\",27): "+ a.lastIndexOf("java",27));
```

```
a.indexOf("java"): 7
a.indexOf("java",7): 7
a.indexOf("java",8): 13
a.lastIndexOf("java"): 28
a.lastIndexOf("java",28): 28
a.lastIndexOf("java",27): 13
```

format(String format, Object … args)

مشابه printf به شکل فرمت شده رشته‌ای می‌سازد. برای آشنایی با انواع فرمت دهی به این لینک مراجعه کنید.

```java
//String.format
String formated = String.format("%d %d %5d %-5d %05d",12,234,45,45,45);
System.out.println(formated)
System.out.println("** *** ***** ***** *****");
formated = String.format("%f %.2f %2.5f %s",12.1234,12.1234,12.1234,"Hello");
System.out.println(formated)
System.out.println("**.****** **.** **.***** *****");
```

```
12 234    45 45    00045
** *** ***** ***** *****
12.123400 12.12 12.12340 Hello
**.****** **.** **.***** *****
```

join(String delimeter, String … args)

رشته‌های تعریف شده را به وسیله جدا کننده معرفی شده به هم می‌چسباند.

```
//String.join
System.out.println("join by comma: "+ String.join(",","i","love","java"));
System.out.println("join by ##: "+ String.join("##","i","love","java"));
```

```
join by comma: i,love,java
join by ##: i##love##java
```

matches(String regex)

آیا رشته مطابقت با یک الگوی تعریف شده را دارد یا خیر؟

```
System.out.println("09121234567".matches("09(12|33)\\d{7}"));
System.out.println("09201234567".matches("09(12|33)\\d{7}"));
System.out.println("1396-12-21".matches("\\d{4}-\\d{2}-\\d{2}"));
System.out.println("1396:12:21".matches("\\d{4}-\\d{2}-\\d{2}"));
```

```
true
false
true
false
```

توضیح عبارات منظم مفصل‌تر و گسترده‌تر است. برای بررسی بیشتر قابلیت‌های عبارات منظم به این لینک یا این لینک مراجعه کنید.

https://github.com/mhrimaz/string-demo