Java String replace

The **java string replace()** method returns a string replacing all the old char or CharSequence to new char or CharSequence.

Since JDK 1.5, a new replace() method is introduced, allowing you to replace a sequence of char values.

Signature

There are two type of replace methods in java string.

**public** String replace(**char** oldChar, **char** newChar)

and

**public** String replace(CharSequence target, CharSequence replacement)

The second replace method is added since JDK 1.5.

Parameters

**oldChar** : old character

**newChar** : new character

**target** : target sequence of characters

**replacement** : replacement sequence of characters

Returns

replaced string

Java String replace(char old, char new) method example

**public** **class** ReplaceExample1{

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

String s1="javatpoint is a very good website";

String replaceString=s1.replace('a','e');//replaces all occurrences of 'a' to 'e'

System.out.println(replaceString);

}

}

jevetpoint is e very good website

Java String replace(CharSequence target, CharSequence replacement) method example

**public** **class** ReplaceExample2{

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

String s1="my name is khan my name is java";

String replaceString=s1.replace("is","was");//replaces all occurrences of "is" to "was"

System.out.println(replaceString);

}

}

my name was khan my name was java

Java String replaceAll

The **java string replaceAll()** method returns a string replacing all the sequence of characters matching regex and replacement string.

Signature

**public** String replaceAll(String regex, String replacement)

Parameters

**regex** : regular expression

**replacement** : replacement sequence of characters

Returns

replaced string

Java String replaceAll() example: replace character

Let's see an example to replace all the occurrences of **a single character**.

**public** **class** ReplaceAllExample1{

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

String s1="javatpoint is a very good website";

String replaceString=s1.replaceAll("a","e");//replaces all occurrences of "a" to "e"

System.out.println(replaceString);

}

}

jevetpoint is e very good website

Java String replaceAll() example: replace word

Let's see an example to replace all the occurrences of **single word or set of words**.

**public** **class** ReplaceAllExample2{

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

String s1="My name is Khan. My name is Bob. My name is Sonoo.";

String replaceString=s1.replaceAll("is","was");//replaces all occurrences of "is" to "was"

System.out.println(replaceString);

}

}

My name was Khan. My name was Bob. My name was Sonoo.

Java String replaceAll() example: remove white spaces

Let's see an example to remove all the occurrences of **white spaces**.

**public** **class** ReplaceAllExample3{

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

String s1="My name is Khan. My name is Bob. My name is Sonoo.";

String replaceString=s1.replaceAll("\\s","");

System.out.println(replaceString);

}

}

MynamewasKhan.MynamewasBob.MynamewasSonoo.

Java String split

The **java string split()** method splits this string against given regular expression and returns a char array.

Signature

There are two signature for split() method in java string.

**public** String split(String regex)

and,

**public** String split(String regex, **int** limit)

Parameter

**regex** : regular expression to be applied on string.

**limit** : limit for the number of strings in array. If it is zero, it will returns all the strings matching regex.

Returns

array of strings

Throws

**PatternSyntaxException** if pattern for regular expression is invalid

Since

1.4

Java String split() method example

The given example returns total number of words in a string excluding space only. It also includes special characters.

**public** **class** SplitExample{

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

String s1="java string split method by javatpoint";

String[] words=s1.split("\\s");//splits the string based on string

//using java foreach loop to print elements of string array

**for**(String w:words){

System.out.println(w);

}

}

}

java

string

split

method

by

javatpoint

Java String split() method with regex and length example

**public** **class** SplitExample2{

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

String s1="welcome to split world";

System.out.println("returning words:");

**for**(String w:s1.split("\\s",0)){

System.out.println(w);

}

System.out.println("returning words:");

**for**(String w:s1.split("\\s",1)){

System.out.println(w);

}

System.out.println("returning words:");

**for**(String w:s1.split("\\s",2)){

System.out.println(w);

}

}

}

returning words:

welcome

to

split

world

returning words:

welcome to split world

returning words:

welcome

to split world

Java String startsWith

The **java string startsWith()** method checks if this string starts with given prefix. It returns true if this string starts with given prefix else returns false.

Signature

The syntax or signature of startWith() method is given below.

**public** **boolean** startsWith(String prefix)

**public** **boolean** startsWith(String prefix, **int** offset)

Parameter

**prefix** : Sequence of character

Returns

true or false

Java String startsWith() method example

**public** **class** StartsWithExample{

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

String s1="java string split method by javatpoint";

System.out.println(s1.startsWith("ja"));

System.out.println(s1.startsWith("java string"));

}

}

Output:

true

true

Java String substring

The **java string substring()** method returns a part of the string.

We pass begin index and end index number position in the java substring method where start index is inclusive and end index is exclusive. In other words, start index starts from 0 whereas end index starts from 1.

There are two types of substring methods in java string.

Signature

1. **public** String substring(**int** startIndex)
2. and
3. **public** String substring(**int** startIndex, **int** endIndex)

If you don't specify endIndex, java substring() method will return all the characters from startIndex.

Parameters

**startIndex** : starting index is inclusive

**endIndex** : ending index is exclusive

Returns

specified string

Throws

**StringIndexOutOfBoundsException** if start index is negative value or end index is lower than starting index.

Java String substring() method example

**public** **class** SubstringExample{

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

String s1="javatpoint";

System.out.println(s1.substring(2,4));//returns va

System.out.println(s1.substring(2));//returns vatpoint

}

}

va

vatpoint

Java String toCharArray

The **java string toCharArray()** method converts this string into character array. It returns a newly created character array, its length is similar to this string and its contents are initialized with the characters of this string.

Signature

The signature or syntax of string toCharArray() method is given below:

**public** **char**[] toCharArray()

Returns

character array

Java String toCharArray() method example

**public** **class** StringToCharArrayExample{

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

String s1="hello";

**char**[] ch=s1.toCharArray();

**for**(**int** i=0;i<ch.length;i++){

System.out.print(ch[i]);

}

}

}

Output:

hello

Java String toLowerCase()

The **java string toLowerCase()** method returns the string in lowercase letter. In other words, it converts all characters of the string into lower case letter.

The toLowerCase() method works same as toLowerCase(Locale.getDefault()) method. It internally uses the default locale.

Signature

There are two variant of toLowerCase() method. The signature or syntax of string toLowerCase() method is given below:

**public** String toLowerCase()

**public** String toLowerCase(Locale locale)

The second method variant of toLowerCase(), converts all the characters into lowercase using the rules of given Locale.

Returns

string in lowercase letter.

Java String toLowerCase() method example

**public** **class** StringLowerExample{

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

String s1="JAVATPOINT HELLO stRIng";

String s1lower=s1.toLowerCase();

System.out.println(s1lower);

}

}

Output:

javatpoint hello string

Java String toUpperCase

The **java string toUpperCase()** method returns the string in uppercase letter. In other words, it converts all characters of the string into upper case letter.

The toUpperCase() method works same as toUpperCase(Locale.getDefault()) method. It internally uses the default locale.

Signature

There are two variant of toUpperCase() method. The signature or syntax of string toUpperCase() method is given below:

**public** String toUpperCase()

**public** String toUpperCase(Locale locale)

The second method variant of toUpperCase(), converts all the characters into uppercase using the rules of given Locale.

Returns

string in uppercase letter.

Java String toUpperCase() method example

**public** **class** StringUpperExample{

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

String s1="hello string";

String s1upper=s1.toUpperCase();

System.out.println(s1upper);

}

}

Output:

HELLO STRING

# Java String trim

The **java string trim()** method eliminates leading and trailing spaces. The unicode value of space character is '\u0020'. The trim() method in java string checks this unicode value before and after the string, if it exists then removes the spaces and returns the omitted string.

#### The string trim() method doesn't omits middle spaces.

### Signature

The signature or syntax of string trim method is given below:

**public** String trim()

### Returns

string with omitted leading and trailing spaces

## Java String trim() method example

**public** **class** StringTrimExample{

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

String s1="  hello string   ";

System.out.println(s1+"javatpoint");//without trim()

System.out.println(s1.trim()+"javatpoint");//with trim()

}

}

hello string javatpoint

hello stringjavatpoint

Java String valueOf

The **java string valueOf()** method converts different types of values into string. By the help of string valueOf() method, you can convert int to string, long to string, boolean to string, character to string, float to string, double to string, object to string and char array to string.

Signature

The signature or syntax of string valueOf() method is given below:

**public** **static** String valueOf(**boolean** b)

**public** **static** String valueOf(**char** c)

**public** **static** String valueOf(**char**[] c)

**public** **static** String valueOf(**int** i)

**public** **static** String valueOf(**long** l)

**public** **static** String valueOf(**float** f)

**public** **static** String valueOf(**double** d)

**public** **static** String valueOf(Object o)

Returns

string representation of given value

Java String valueOf() method example

**public** **class** StringValueOfExample{

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

**int** value=30;

String s1=String.valueOf(value);

System.out.println(s1+10);//concatenating string with 10

}

}

Output:

3010