

Java String charAt

The **java string charAt()** method returns a char value at the given index number. The index number starts from 0.

Signature

The signature of string charAt() method is given below:

**public** **char** charAt(**int** index)

Parameter

**index** : index number, starts with 0

Returns

**char value**

Specified by

**CharSequence** interface

Throws

**IndexOutOfBoundsException** : if index is negative value or greater than this string length.

Java String charAt() method example

**public** **class** CharAtExample{

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

String name="javatpoint";

**char** ch=name.charAt(4);//returns the char value at the 4th index

System.out.println(ch);

}

}

Output:

t

Java String compareTo

The **java string compareTo()** method compares the given string with current string lexicographically. It returns positive number, negative number or 0.

If first string is greater than second string, it returns positive number (difference of character value). If first string is less than second string, it returns negative number and if first string is equal to second string, it returns 0.

s1 > s2 => positive number

s1 < s2 => negative number

s1 == s2 => 0

Signature

**public** **int** compareTo(String anotherString)

Parameters

**anotherString**: represents string that is to be compared with current string

Returns

an integer value

Java String compareTo() method example

**public** **class** LastIndexOfExample{

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

String s1="hello";

String s2="hello";

String s3="meklo";

String s4="hemlo";

System.out.println(s1.compareTo(s2));

System.out.println(s1.compareTo(s3));

System.out.println(s1.compareTo(s4));

}

}

Output:

0

-5

-1

Java String concat

The **java string concat()** method *combines specified string at the end of this string*. It returns combined string. It is like appending another string.

Signature

The signature of string concat() method is given below:

**public** String concat(String anotherString)

Parameter

**anotherString** : another string i.e. to be combined at the end of this string.

Returns

combined string

Java String concat() method example

**public** **class** ConcatExample{

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

String s1="java string";

s1.concat("is immutable");

System.out.println(s1);

s1=s1.concat(" is immutable so assign it explicitly");

System.out.println(s1);

}

}

java string

java string is immutable so assign it explicitly

Java String contains

The **java string contains()** method searches the sequence of characters in this string. It returns *true* if sequence of char values are found in this string otherwise returns *false*.

Signature

The signature of string contains() method is given below:

**public** **boolean** contains(CharSequence sequence)

Parameter

**sequence** : specifies the sequence of characters to be searched.

Returns

**true** if sequence of char value exists, otherwise **false**.

Throws

**NullPointerException** : if sequence is null.

Java String contains() method example

**class** ContainsExample{

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

String name="what do you know about me";

System.out.println(name.contains("do you know"));

System.out.println(name.contains("about"));

System.out.println(name.contains("hello"));

}

}

true

true

false

Java String endsWith

The **java string endsWith()** method checks if this string ends with given suffix. It returns true if this string ends with given suffix else returns false.

Signature

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

**public** **boolean** endsWith(String suffix)

Parameter

**suffix** : Sequence of character

Returns

true or false

Java String endsWith() method example

**public** **class** EndsWithExample{

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

String s1="java by javatpoint";

System.out.println(s1.endsWith("t"));

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

}

}

Output:

true

true

Java String equals

The **java string equals()** method compares the two given strings based on the content of the string. If any character is not matched, it returns false. If all characters are matched, it returns true.

The String equals() method overrides the equals() method of Object class.

Signature

**public** **boolean** equals(Object anotherObject)

Parameter

**anotherObject** : another object i.e. compared with this string.

Returns

**true** if characters of both strings are equal otherwise **false**.

Overrides

equals() method of java Object class.

Java String equals() method example

**public** **class** EqualsExample{

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

String s1="javatpoint";

String s2="javatpoint";

String s3="JAVATPOINT";

String s4="python";

System.out.println(s1.equals(s2));//true because content and case is same

System.out.println(s1.equals(s3));//false because case is not same

System.out.println(s1.equals(s4));//false because content is not same

}

}

true

false

false

Java String format

The **java string format()** method returns the formatted string by given locale, format and arguments.

If you don't specify the locale in String.format() method, it uses default locale by calling *Locale.getDefault()* method.

The format() method of java language is like *sprintf()* function in c language and *printf()* method of java language.

Signature

There are two type of string format() method:

**public** **static** String format(String format, Object... args)

and,

**public** **static** String format(Locale locale, String format, Object... args)

Parameters

**locale** : specifies the locale to be applied on the format() method.

**format** : format of the string.

**args** : arguments for the format string. It may be zero or more.

Returns

formatted string

Throws

**NullPointerException** : if format is null.

**IllegalFormatException** : if format is illegal or incompatible.

Java String format() method example

**public** **class** FormatExample{

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

String name="sonoo";

String sf1=String.format("name is %s",name);

String sf2=String.format("value is %f",32.33434);

String sf3=String.format("value is %32.12f",32.33434);//returns 12 char fractional part filling with 0

System.out.println(sf1);

System.out.println(sf2);

System.out.println(sf3);

}

}

name is sonoo

value is 32.334340

value is 32.334340000000

Java String getBytes()

The **java string getBytes()** method returns the byte array of the string. In other words, it returns sequence of bytes.

Signature

There are 3 variant of getBytes() method. The signature or syntax of string getBytes() method is given below:

**public** **byte**[] getBytes()

**public** **byte**[] getBytes(Charset charset)

**public** **byte**[] getBytes(String charsetName)**throws** UnsupportedEncodingException

Returns

sequence of bytes.

Java String getBytes() method example

**public** **class** StringGetBytesExample{

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

String s1="ABCDEFG";

**byte**[] barr=s1.getBytes();

**for**(**int** i=0;i<barr.length;i++){

System.out.println(barr[i]);

}

}

}

Output:

65

66

67

68

69

70

71

Java String indexOf

The **java string indexOf()** method returns index of given character value or substring. If it is not found, it returns -1. The index counter starts from zero.

Signature

There are 4 types of indexOf method in java. The signature of indexOf methods are given below:

|  |  |  |
| --- | --- | --- |
| No. | Method | Description |
| 1 | int indexOf(int ch) | returns index position for the given char value |
| 2 | int indexOf(int ch, int fromIndex) | returns index position for the given char value and from index |
| 3 | int indexOf(String substring) | returns index position for the given substring |
| 4 | int indexOf(String substring, int fromIndex) | returns index position for the given substring and from index |

Parameters

**ch**: char value i.e. a single character e.g. 'a'

**fromIndex**: index position from where index of the char value or substring is retured

**substring**: substring to be searched in this string

Returns

index of the string

Java String indexOf() method example

**public** **class** IndexOfExample{

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

String s1="this is index of example";

//passing substring

**int** index1=s1.indexOf("is");//returns the index of is substring

**int** index2=s1.indexOf("index");//returns the index of index substring

System.out.println(index1+"  "+index2);//2 8

//passing substring with from index

**int** index3=s1.indexOf("is",4);//returns the index of is substring after 4th index

System.out.println(index3);//5 i.e. the index of another is

//passing char value

**int** index4=s1.indexOf('s');//returns the index of s char value

System.out.println(index4);//3

}

}

2 8

5

3

Java String intern

The **java string intern()** method returns the interned string. It returns the canonical representation of string.

It can be used to return string from pool memory, if it is created by new keyword.

Signature

The signature of intern method is given below:

**public** String intern()

Returns

interned string

Java String intern() method example

**public** **class** InternExample{

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

String s1=**new** String("hello");

String s2="hello";

String s3=s1.intern();//returns string from pool, now it will be same as s2

System.out.println(s1==s2);//false because reference is different

System.out.println(s2==s3);//true because reference is same

}

}

false

true

Java String isEmpty

The **java string isEmpty()** method checks if this string is empty. It returns *true*, if length of string is 0 otherwise *false*.

The isEmpty() method of String class is included in java string since JDK 1.6.

Signature

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

**public** **boolean** isEmpty()

Returns

true if length is 0 otherwise false.

Since

**1.6**

Java String isEmpty() method example

**public** **class** IsEmptyExample{

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

String s1="";

String s2="javatpoint";

System.out.println(s1.isEmpty());

System.out.println(s2.isEmpty());

}

}

true

false

Java String join

The **java string join()** method returns a string joined with given delimiter. In string join method, delimiter is copied for each elements.

In case of null element, "null" is added. The join() method is included in java string since JDK 1.8.

There are two types of join() methods in java string.

Signature

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

**public** **static** String join(CharSequence delimiter, CharSequence... elements)

and

**public** **static** String join(CharSequence delimiter, Iterable<? **extends** CharSequence> elements)

Parameters

**delimiter** : char value to be added with each element

**elements** : char value to be attached with delimiter

Returns

joined string with delimiter

Throws

**NullPointerException** if element or delimiter is null.

Since

**1.8**

Java String join() method example

**public** **class** StringJoinExample{

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

String joinString1=String.join("-","welcome","to","javatpoint");

System.out.println(joinString1);

}

}

welcome-to-javatpoint

Java String lastIndexOf

The **java string lastIndexOf()** method returns last index of the given character value or substring. If it is not found, it returns -1. The index counter starts from zero.

Signature

There are 4 types of lastIndexOf method in java. The signature of lastIndexOf methods are given below:

|  |  |  |
| --- | --- | --- |
| No. | Method | Description |
| 1 | int lastIndexOf(int ch) | returns last index position for the given char value |
| 2 | int lastIndexOf(int ch, int fromIndex) | returns last index position for the given char value and from index |
| 3 | int lastIndexOf(String substring) | returns last index position for the given substring |
| 4 | int lastIndexOf(String substring, int fromIndex) | returns last index position for the given substring and from index |

Parameters

**ch**: char value i.e. a single character e.g. 'a'

**fromIndex**: index position from where index of the char value or substring is retured

**substring**: substring to be searched in this string

Returns

last index of the string

Java String lastIndexOf() method example

1. **public** **class** LastIndexOfExample{
2. **public** **static** **void** main(String args[]){
3. String s1="this is index of example";//there are 2 's' characters in this sentence
4. **int** index1=s1.lastIndexOf('s');//returns last index of 's' char value
5. System.out.println(index1);//6
6. }}

Output:

6

Java String length

The **java string length()** method length of the string. It returns count of total number of characters. The length of java string is same as the unicode code units of the string.

Signature

The signature of the string length() method is given below:

**public** **int** length()

Specified by

CharSequence interface

Returns

length of characters

Java String length() method example

**public** **class** LengthExample{

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

String s1="javatpoint";

String s2="python";

System.out.println("string length is: "+s1.length());//10 is the length of javatpoint string

System.out.println("string length is: "+s2.length());//6 is the length of python string

}

}

string length is: 10

string length is: 6