**JAVA STRING METHODS**

* The ***java.lang.String*** class provides a lot of built-in methods that are used to manipulate **string in Java.**
* These methods, we can perform operations on String objects such as trimming, concatenating, converting, comparing, replacing strings etc.

**LIST OF STRING METHODS**

**int length()**

* Returns the number of characters in the string.

**Example:**

public class StringMethodExample{

public static void main(String[] args){

String s1 = “Program”;

int len = s1.length();

System.out.println(len);

}

}

**Output:**

7

**char charAt(int index)**

* Returns the character at the specified index in the string.

**Example:**

public class StringMethodExample1{

public static void main(String[] args){

String s1 = “Program”;

Char res = s1.charAt(3);

System.out.println(res);

}

}

**Output:**

g

**void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)**

* Copies characters from a source string object into the destination character array.
* The first character to be copied is at index ***srcBegin***, the last character to be copied is at index ***srcEnd-1***.
* The character is copied into ***dst***, starting at index ***dstBegin***.

**Parameters:**

* **int srcBeginIndex:** The index from where copying of characters is started.
* **int srcEndIndex:** The index which is next to the last character that is getting copied.
* **Char[] destination:** The char array where characters from the string that invokes the getChars() method is getting copied.
* **int dstEndIndex:** It shows the position in the destination array from where the characters from the string will be pushed.

**Example:**

public class StringMethodEx2{

public static void main(String[] args) {

String s1 = new String("Welcome to java ");

char[] s2 = new char[6];

s1.getChars(11, 15, s2, 0);

System.out.print("Value copied= ");

System.out.println(s2);

}

}

**Output:**

java

0th index 1st index 2nd index 3rd index 4th index 5th index

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0 | 0 | j | a | v | a |

dstBegin—0

dstBegin—1

dstBegin—2

ArrayIndexOutOfBoundException

**Boolean equals(object obj)**

* Compares the current string object with the other string and returns a boolean value.

**Example:**

public class StringMethodExample3{

public static void main(String[] args){

String s1 = “Program”;

String s2 = “Program”;

Boolean res1 = s1.equals(s2);

System.out.println(res1);

}

}

**Output:**

True

**int compareTo(String str)**

* Compares the current string object with another string.
* It returns positive number, negative number or 0.
* It compares strings on the basis of the Unicode value of each character in the strings.

**Note:**

* If the first string is lexicographically greater than second string it returns positive number.
* If the first string is less than the second string lexicographically, it returns negative number.
* If the first string is lexicographically equal to second string, it returns 0.

**Sample format:**

S1>S2🡪Positive number

S1<S2🡪Negative number

S1==S2🡪0

**Example:**

public class StringMethodExample4{

public static void main(String[] args){

String s1 = “heuits”;

String s2 = “hellos”;

int res2 = s1.compareTo(s2);

System.out.println(res2);

}

}

**Output:**

9

**boolean startsWith(String prefix)**

* Tests whether a string starts with the specified prefix or not.
* If the character sequence represented by the argument is a prefix of the string, the return value is *true*. Otherwise, it is *false*.

**Example:**

public class StringMethodExample5{

public static void main(String[] args){

String s1 = “Welcome to java ”;

System.out.print(“Value returned:”);

System.out.println(s1.startsWith(“Welcome”));

}

}

**Output:**

Value returned: true

**boolean endsWith(String suffix)**

* Tests if the string object ends with the specified suffix.
* Returns *true* if the character sequence represented by the argument is a suffix of the character sequence represented by object. Otherwise, it is *false*.

**Example:**

public class StringMethodExample6{

public static void main(String[] args){

String s1 = “Welcome to java ”;

System.out.print(“Value returned:”);

System.out.println(s1.endsWith(“java”));

}

}

**Output:**

Value returned: true

**int indexOf(int ch)**

* Returns the index of the first occurrence of the specified character within a string.
* If the character is not found, the method returns -1.

**Example:**

public class StringMethodExample7{

public static void main(String[] args){

String s1 = “Welcome to java ”;

System.out.print(“Index Found at:”);

System.out.println(s1.indexOf(‘o’));

}

}

**Output:**

Index Found at:4

**int lastIndexOf(int ch)**

* Returns the index of the specified character occurring last in the string.
* If the character is not found, the method, returns -1.

**Example:**

public class StringMethodExample8{

public static void main(String[] args){

String s1 = “Welcome to java ”;

System.out.print(“Last Index Found at:”);

System.out.println(s1.lastIndexOf(‘o’));

}

}

**Output:**

Last Index Found at:9

**String substring(int beginindex)**

* Returns a substring of a string.
* The substring begins with the character at the specified index and extends to the end of the main string.

**Example:**

public class StringMethodExample9{

public static void main(String[] args){

String s1 = “Welcome to java ”;

System.out.print(“Value returned:”);

System.out.println(s1.subString(10));

}

}

**Output:**

Value returned: java

**String concat(String str)**

* Concatenates the specified string to the end of the string object.

**Example:**

public class StringMethodExample10{

public static void main(String[] args){

String s1 = “Hello ”;

String s2 = “world”;

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

}

}

**Output:**

Hello world

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

* Replaces the occurrence of a specified character by a new specified character.
* Returns the string derived from the string by replacing every occurrence of *oldChar* with *newChar*

**Example:**

public class StringMethodExample11{

public static void main(String[] args){

String s1 = “Welcome to java ”;

System.out.print(“Value returned:”);

System.out.println(s1.replace(‘o’, ‘T’));

}

}

**Output:**

Value returned: WelcTme tT java

**String toUpperCase()**

* Converts the string to uppercase and returns it.

**Example:**

public class StringMethodExample12{

public static void main(String[] args){

String s1 = “mugilan ”;

System.out.print(“Value returned:”);

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

}

}

**Output:**

Value returned: MUGILAN

**String toLowerCase()**

* Converts the string to lowercase and returns it.

**Example:**

public class StringMethodExample13{

public static void main(String[] args){

String s1 = “MUGILAN”;

System.out.print(“Value returned:”);

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

}

}

**Output:**

Value returned: mugilan

**char[] toCharArray()**

* Returns a newly allocated array whose length is the length of the string and whose content are initialized to contain the character sequence represented by this string.

**Example:**

public class StringMethodExample14{

public static void main(String[] args){

String s1 = “WELCOME TO JAVA”;

System.out.print(“Value returned:”);

char[] ch = s1.toCharArray();

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

}

}

**Output:**

Value returned: WELCOME TO JAVA

**String valueOf(Object obj)**

* Returns the string representation of the specified argument.
* Returns null if the argument is null.
* The valueOf() method is a static method

**Example:**

public class StringMethodExample15{

public static void main(String[] args){

int a =10;  
double b = 2.000;  
char[] arr = {'a', 'g', 'e'};  
System.out.println("Return value: "+ String.valueOf(a));  
System.out.println("Return double value: "+ String.valueOf(b));

}

}

**Output:**

Return value: 10

Return double value: 2.0

**boolean equalsIgnoreCase(string anotherString)**

* Compares this string with another string and ignores case considerations.

**Example:**

public class StringMethodExample16{

public static void main(String[] args){

String s1 = “MUGILAN”;

String s2 = “mugilan”;

System.out.print(“Return:”);

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

}

}

**Output:**

Return: true

**String trim()**

* Removes whitespace from both ends of a string object and returns the trimmed string.

**Example:**

public class StringMethodExample17{

public static void main(String[] args){

String s1 = new String(“ Welcome to java”);

System.out.print(“Value returned:”);

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

}

}

**Output:**

Value returned: Welcome to java

**String split()**

* Splits the string around matches of the given regular expression.

**Example:**

public class StringMethodExample18{

public static void main(String[] args){

String s1 = “Object Oriented Programming Language”;

String[] data = s1.split(“\\s”);

for(String d : data){

System.out.println(d);

}

}

}

**Output:**

Object

Oriented

Programming

Language

**String join()**

* Returns a string joined with a given delimiter.
* In the String join() method, the delimiter is copied for each element.
* The join() method is included in the Java string since JDK 1.8.

**Example:**

public class StringMethodExample19{

public static void main(String[] args){

String s1 = String.join(“ ”, “Java”, “contains”, “string”, “concept”);

System.out.println(s1);

}

}

**String isEmpty()**

* Checks if the input string is empty or not.

**Note:**

* Empty means the number of characters contained in a string is zero.

**Example:**

public class StringMethodExample20{

public static void main(String[] args){

String s1 = “”;

String s2 = “Java program”;

System.out.println(“Return value1:”+s1.isEmpty());

System.out.println(“Return value2:”+s2.isEmpty());

}

}

**Output:**

Return value1: true

Return value2: false

**String ReplaceAll()**

* Returns a string replacing all the sequence of characters matching regex and replacement string.

**Example:**

public class StringMethodExample21{

public static void main(){

String s1 = “Java is platform independent”;

String str = s1.replaceAll(“a”, “e”);

System.out.println(str);

}

}

**Output:**

Jeve is platform indepebdent

**String format()**

* Returns a formatted string using the specified format string and arguments.

**Example:**

public class StringMethodExample22{

public static void main(String[] args){

String s1 = “Blockchain”;

String sf1 = String.format(“name is %s”, s1);

String sf2 = String.format(“Value is %f”,25.3145);

String sf3 = String.format(“Value is %32.12f”,25.3145);

System.out.println(sf1);

System.out.println(sf2);

System.out.println(sf3);

}

}

**Output:**

name is Blockchain

Value is 25.314500

Value is 25.314500000000

**String intern()**

* Returns the interned string.
* It returns the canonical representation of string.
* It can be used to return string from memory if it is created by a new keyword.
* It creates an exact copy of the heap string object in the String Constant Pool.

**Example:**

public class StringMethodExample23{

public static void main(String[] args){

String s1 = new String(“JavaDeveloper”);

String s2 = “JavaDeveloper”;

String s3 = s1.intern();

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

System.out.println(s2==s3);

}

}

**Output:**

false

true