2.4.6 Java String Methods

The **java string charAt()** method returns *a char value at the given index number*. The index number starts from 0. It returns StringIndexOutOfBoundsException if given index number is greater than this string or negative index number.

Signature

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

1. public char charAt(int index)

Parameter

index : index number, starts with 0

Returns

char value

Specified by

CharSequence interface

Throws

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

Java String charAt() method example

```
1. public class CharAtExample{
2. public static void main(String args[]){
3. String name="javatpoint";
4. char ch=name.charAt(4);//returns the char value at the 4th index
5. System.out.println(ch);
6. }}
```

Test it Now

Output:
t

StringIndexOutOfBoundsException with charAt()

Let's see the example of charAt() method where we are passing greater index value. In such case, it throws StringIndexOutOfBoundsException at run time.

```
1. public class CharAtExample{
2. public static void main(String args[]){
3. String name="javatpoint";
4. char ch=name.charAt(10);//returns the char value at the 10th index
5. System.out.println(ch);
6. }}
```

Output:

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException:
String index out of range: 10
at java.lang.String.charAt(String.java:658)
at CharAtExample.main(CharAtExample.java:4)
```

Java String compareTo()

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

It compares strings on the basis of Unicode value of each character in the strings.

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

1. if s1 > s2, it returns positive number
2. if s1 < s2, it returns negative number
3. if s1 == s2, it returns 0

Signature

1. 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

```
1. public class CompareToExample{
2. public static void main(String args[]){
3. String s1="hello";
4. String s2="hello";
5. String s3="meklo";
6. String s4="hemlo";
7. String s5="flag";
8. System.out.println(s1.compareTo(s2));//0 because both are equal
9. System.out.println(s1.compareTo(s3));//-5 because "h" is 5 times lower than "m"
10. System.out.println(s1.compareTo(s4));//-1 because "l" is 1 times lower than "m"
11. System.out.println(s1.compareTo(s5));//2 because "h" is 2 times greater than "f"
12. }}
```

Test it Now

Output:

```
0
-5
-1
2
```

Java String compareTo(): empty string

If you compare string with blank or empty string, it returns length of the string. If second string is empty, result would be positive. If first string is empty, result would be negative.

```
1. public class CompareToExample2{
2. public static void main(String args[]){
3. String s1="hello";
4. String s2="";
5. String s3="me";
6. System.out.println(s1.compareTo(s2));
7. System.out.println(s2.compareTo(s3));
8. }}
```

Test it Now

Output:
5
-2

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:

```
1. 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

```
1. public class ConcatExample{
2. public static void main(String args[]){
3. String s1="java string";
4. s1.concat("is immutable");
5. System.out.println(s1);
6. s1=s1.concat(" is immutable so assign it explicitly");
7. System.out.println(s1);
8. }}
```

Test it Now

```
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:

```
1. 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

```
1. class ContainsExample{
2. public static void main(String args[]){
3. String name="what do you know about me";
4. System.out.println(name.contains("do you know"));
5. System.out.println(name.contains("about"));
6. System.out.println(name.contains("hello"));
7. }}
```

Test it Now

```
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.

```
1. public boolean endsWith(String suffix)
```

Parameter

suffix : Sequence of character

Returns

true or false

Java String endsWith() method example

```
1. public class EndsWithExample{
2. public static void main(String args[]){
3. String s1="java by javatpoint";
4. System.out.println(s1.endsWith("t"));
5. System.out.println(s1.endsWith("point"));
6. }}
```

Test it Now

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

```
1. 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

```
1. public class EqualsExample{
2. public static void main(String args[]){
3. String s1="javatpoint";
4. String s2="javatpoint";
5. String s3="JAVATPOINT";
6. String s4="python";
7. System.out.println(s1.equals(s2));//true because content and case is same
8. System.out.println(s1.equals(s3));//false because case is not same
9. System.out.println(s1.equals(s4));//false because content is not same
10. }}
```

Test it Now

```
true
false
false
```

Java String equalsIgnoreCase()

The **String equalsIgnoreCase()** method compares the two given strings on the basis of content of the string irrespective of case of the string. It is like equals() method but doesn't check case. If any character is not matched, it returns false otherwise it returns true.

Signature

```
1. public boolean equalsIgnoreCase(String str)
```

Parameter

str : another string i.e. compared with this string.

Returns

It returns **true** if characters of both strings are equal ignoring case otherwise **false**.

Java String equalsIgnoreCase() method example

```
1. public class EqualsIgnoreCaseExample{
2. public static void main(String args[]){
3. String s1="javatpoint";
4. String s2="javatpoint";
5. String s3="JAVATPOINT";
6. String s4="python";
7. System.out.println(s1.equalsIgnoreCase(s2));//true because content and case both are same
8. System.out.println(s1.equalsIgnoreCase(s3));//true because case is ignored
9. System.out.println(s1.equalsIgnoreCase(s4));//false because content is not same
10. }}
```

Test it Now

```
true
true
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:

1. public static String format(String format, Object... args)
2. and,
3. 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

```
1. public class FormatExample{
2. public static void main(String args[]){
3. String name="sonoo";
4. String sf1=String.format("name is %s",name);
5. String sf2=String.format("value is %f",32.33434);
6. String sf3=String.format("value is %32.12f",32.33434);//returns 12 char fractional part filling with 0
7.
8. System.out.println(sf1);
9. System.out.println(sf2);
10. System.out.println(sf3);
11. }}
```

Test it Now

```
name is sonoo
value is 32.334340
value is 32.334340000000
```

ava 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:

1. public byte[] getBytes()
2. public byte[] getBytes(Charset charset)
3. public byte[] getBytes(String charsetName)throws UnsupportedEncodingException

---

Returns

sequence of bytes.

Java String getBytes() method example

```
1. public class StringGetBytesExample{
2. public static void main(String args[]){
3. String s1="ABCDEFG";
4. byte[] barr=s1.getBytes();
5. for(int i=0;i<barr.length;i++){
6. System.out.println(barr[i]);
7. }
8. }}
```

Test it Now

Output:

```
65
66
67
68
69
70
71
```

Java String getChars()

The **java string getChars()** method copies the content of this string into specified char array. There are 4 arguments passed in getChars() method. The signature of getChars() method is given below:

Signature

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

```
1. public void getChars(int srcBeginIndex, int srcEndIndex, char[] destination, int dstBeginIndex)
```

Returns

It doesn't return any value.

Throws

It throws StringIndexOutOfBoundsException if beginIndex is greater than endIndex.

Java String getChars() method example

```
1. public class StringGetCharsExample{
2. public static void main(String args[]){
3.  String str = new String("hello javatpoint how r u");
4.       char[] ch = new char[10];
5.       try{
6.          str.getChars(6, 16, ch, 0);
7.          System.out.println(ch);
8.       }catch(Exception ex){System.out.println(ex);}
9.  }}
```

Test it Now

Output:

```
javatpoint
```

ava 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

```
1. public class IndexOfExample{
2. public static void main(String args[]){
3. String s1="this is index of example";
4. //passing substring
5. int index1=s1.indexOf("is");//returns the index of is substring
6. int index2=s1.indexOf("index");//returns the index of index substring
7. System.out.println(index1+"  "+index2);//2 8
8.
9. //passing substring with from index
10. int index3=s1.indexOf("is",4);//returns the index of is substring after 4th index
11. System.out.println(index3);//5 i.e. the index of another is
12.
13. //passing char value
14. int index4=s1.indexOf('s');//returns the index of s char value
15. System.out.println(index4);//3
16. }}
```

Test it Now

```
2  8
5
3
```

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:

```
1. public boolean isEmpty()
```

Returns

true if length is 0 otherwise false.

Since

1.6

Java String isEmpty() method example

```
1. public class IsEmptyExample{
2. public static void main(String args[]){
3. String s1="";
4. String s2="javatpoint";
5.
6. System.out.println(s1.isEmpty());
7. System.out.println(s2.isEmpty());
8. }}
```

Test it Now

```
true
false
```

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. }}
```

Test it Now

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:

1. public int length()

Specified by

CharSequence interface

Returns

length of characters

Java String length() method example

```
1. public class LengthExample{
2. public static void main(String args[]){
3. String s1="javatpoint";
4. String s2="python";
5. System.out.println("string length is: "+s1.length());//10 is the length of javatpoint string
6. System.out.println("string length is: "+s2.length());//6 is the length of python string
7. }}
```

Test it Now

```
string length is: 10
string length is: 6
```

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.

1. public String replace(char oldChar, char newChar)
2. and
3. 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

```
1. public class ReplaceExample1{
2. public static void main(String args[]){
3. String s1="javatpoint is a very good website";
4. String replaceString=s1.replace('a','e');//replaces all occurrences of 'a' to 'e'
5. System.out.println(replaceString);
6. }}
```

Test it Now

```
jevetpoint is e very good website
```

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

```
1. public class ReplaceExample2{
2. public static void main(String args[]){
3. String s1="my name is khan my name is java";
4. String replaceString=s1.replace("is","was");//replaces all occurrences of "is" to "was"
5. System.out.println(replaceString);
6. }}
```

Test it Now

```
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

```
1. 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**.

```
1. public class ReplaceAllExample1{
2. public static void main(String args[]){
3. String s1="javatpoint is a very good website";
4. String replaceString=s1.replaceAll("a","e");//replaces all occurrences of "a" to "e"
5. System.out.println(replaceString);
6. }}
```

Test it Now

```
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**.

```
1. public class ReplaceAllExample2{
2. public static void main(String args[]){
3. String s1="My name is Khan. My name is Bob. My name is Sonoo.";
4. String replaceString=s1.replaceAll("is","was");//replaces all occurrences of "is" to "was"
5. System.out.println(replaceString);
6. }}
```

Test it Now

```
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**.

```
1. public class ReplaceAllExample3{
2. public static void main(String args[]){
3. String s1="My name is Khan. My name is Bob. My name is Sonoo.";
4. String replaceString=s1.replaceAll("\\s","");
5. System.out.println(replaceString);
6. }}
```

Test it Now

```
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.

```
1. public String split(String regex)
2. and,
3. 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.

```
1. public class SplitExample{
2. public static void main(String args[]){
3. String s1="java string split method by javatpoint";
4. String[] words=s1.split("\\s");//splits the string based on whitespace
5. //using java foreach loop to print elements of string array
6. for(String w:words){
7. System.out.println(w);
8. }
9. }}
```

Test it Now

```
java
string
split
method
by
javatpoint
```

Java String split() method with regex and length example

```
1. public class SplitExample2{
2. public static void main(String args[]){
3. String s1="welcome to split world";
4. System.out.println("returning words:");
5. for(String w:s1.split("\\s",0)){
6. System.out.println(w);
7. }
8. System.out.println("returning words:");
9. for(String w:s1.split("\\s",1)){
10. System.out.println(w);
11. }
12. System.out.println("returning words:");
13. for(String w:s1.split("\\s",2)){
14. System.out.println(w);
15. }
16.
17. }}
```

Test it Now

```
returning words:
welcome
to
split
world
returning words:
welcome to split world
returning words:
welcome
to split world
```

ava 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.

1. public boolean startsWith(String prefix)
2. public boolean startsWith(String prefix, int offset)

Parameter

prefix : Sequence of character

Returns

true or false

Java String startsWith() method example

```
1. public class StartsWithExample{
2. public static void main(String args[]){
3. String s1="java string split method by javatpoint";
4. System.out.println(s1.startsWith("ja"));
5. System.out.println(s1.startsWith("java string"));
6. }}
```

Test it Now

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

```
1. public class SubstringExample{
2. public static void main(String args[]){
3. String s1="javatpoint";
4. System.out.println(s1.substring(2,4));//returns va
5. System.out.println(s1.substring(2));//returns vatpoint
6. }}
```

Test it Now

```
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:

```
1. public char[] toCharArray()
```

Returns

character array

Java String toCharArray() method example

```
1. public class StringToCharArrayExample{
2. public static void main(String args[]){
3. String s1="hello";
4. char[] ch=s1.toCharArray();
5. for(int i=0;i<ch.length;i++){
6. System.out.print(ch[i]);
7. }
8. }}
```

Test it Now

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:

```
1. public String toLowerCase()
2. 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

```
1. public class StringLowerExample{
2. public static void main(String args[]){
3. String s1="JAVATPOINT HELLO stRIng";
4. String s1lower=s1.toLowerCase();
5. System.out.println(s1lower);
6. }}
```

Test it Now

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:

```
1. public String toUpperCase()
2. 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

```
1. public class StringUpperExample{
2. public static void main(String args[]){
3. String s1="hello string";
4. String s1upper=s1.toUpperCase();
5. System.out.println(s1upper);
6. }}
```

Test it Now

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:

1. public String trim()

Returns

string with omitted leading and trailing spaces

Java String trim() method example

```
1. public class StringTrimExample{
2. public static void main(String args[]){
3. String s1="  hello string   ";
4. System.out.println(s1+"javatpoint");//without trim()
5. System.out.println(s1.trim()+"javatpoint");//with trim()
6. }}
```

Test it Now

```
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:

1. public static String valueOf(boolean b)
2. public static String valueOf(char c)
3. public static String valueOf(char[] c)
4. public static String valueOf(int i)
5. public static String valueOf(long l)
6. public static String valueOf(float f)
7. public static String valueOf(double d)
8. public static String valueOf(Object o)

Returns

string representation of given value

Java String valueOf() method example

```
1. public class StringValueOfExample{
2. public static void main(String args[]){
3. int value=30;
4. String s1=String.valueOf(value);
5. System.out.println(s1+10);//concatenating string with 10
6. }}
```

Test it Now

Output:
3010