

3.7 String in Java

- String is a sequence of characters.
- In java, objects of String are immutable which means a constant and cannot be changed once created.

❑ Creating a String

There are two ways to create string in Java:

String literal

```
String s = "GeeksforGeeks";
```

❑ Using *new* keyword

```
String s = new String ("GeeksforGeeks");
```

❑ Constructors

1. **String(byte[] byte_arr)** – Construct a new String by decoding the *byte array*. It uses the platform's default character set for decoding.

Example:

```
byte[] b_arr = {71, 101, 101, 107, 115};  
String s_byte = new String(b_arr); //Geeks
```

2. **String(byte[] byte_arr, Charset char_set)** – Construct a new String by decoding the *byte array*. It uses the *char_set* for decoding.

Example:

```
byte[] b_arr = {71, 101, 101, 107, 115};  
Charset cs = Charset.defaultCharset();  
String s_byte_char = new String(b_arr, cs); //Geeks
```

3. **String(byte[] byte_arr, String char_set_name)** – Construct a new String by decoding the *byte array*. It uses the *char_set_name* for decoding.

It looks similar to the above constructs and they before similar functions but it takes the *String(which contains*

char_set_name) as parameter while the above constructor takes *CharSet*.

Example:

```
byte[] b_arr = {71, 101, 101, 107, 115};  
String s = new String(b_arr, "US-ASCII"); //Geeks
```

4. String(byte[] byte_arr, int start_index, int length) –

Construct a new string from the *bytes array* depending on the *start_index*(*Starting location*) and *length*(*number of characters from starting location*).

Example:

```
byte[] b_arr = {71, 101, 101, 107, 115};  
String s = new String(b_arr, 1, 3); // Geek
```

5. String(byte[] byte_arr, int start_index, int length, Charset char_set) –

Construct a new string from the *bytes array* depending on the *start_index* (*Starting location*) and *length*(*number of characters from starting location*). Uses *char_set* for decoding.

Example:

```
byte[] b_arr = {71, 101, 101, 107, 115};  
Charset cs = Charset.defaultCharset();  
String s = new String(b_arr, 1, 3, cs); // Geek
```

6. String(byte[] byte_arr, int start_index, int length, String char_set_name) –

Construct a new string from the *bytes array* depending on the *start_index*(*Starting location*) and *length*(*number of characters from starting location*).

Uses *char_set_name* for decoding.

Example:

```
byte[] b_arr = {71, 101, 101, 107, 115};  
String s = new String(b_arr, 1, 4, "US-ASCII"); // Geeks
```

7. String(char[] char_arr) – Allocates a new String from the given *Character array*

Example:

```
char char_arr[] = {'G', 'e', 'e', 'k', 's'};  
String s = new String(char_arr); //Geeks
```

8. String(char[] char_array, int start_index, int count) – Allocates a String from a given *character array* but choose *count* characters from the *start_index*.

Example:

```
char char_arr[] = {'G', 'e', 'e', 'k', 's'};  
String s = new String(char_arr, 1, 3); //Geek
```

9. String(int[] uni_code_points, int offset, int count) – Allocates a String from a *uni_code_array* but choose *count* characters from the *start_index*.

Example:

```
int[] uni_code = {71, 101, 101, 107, 115};  
String s = new String(uni_code, 1, 3); //Geek
```

10. String(StringBuffer s_buffer) – Allocates a new string from the string in *s_buffer*

Example:

```
String s_buffer = "Geeks";  
String s = new String(s_buffer); //Geeks
```

11. String(StringBuilder s_builder) – Allocates a new string from the string in *s_builder*

Example:

```
String s_builder = "Geeks";  
String s = new String(s_builder); //Geeks
```

12. String Methods int length(): Returns the number of characters in the String.

```
"GeeksforGeeks".length(); // returns 13
```

13. Char charAt(int i): Returns the character at ith index.

```
"GeeksforGeeks".charAt(3); // returns 'k'
```

14. String substring (int i): Return the substring from the ith index character to end.

```
"GeeksforGeeks".substring(3); // returns "ksforGeeks"
```

15. String substring (int i, int j): Returns the substring from i to j-1 index.

```
"GeeksforGeeks".substring(2, 5); // returns "eks"
```

16. String concat(String str): Concatenates specified string to the end of this string.

```
String s1 = "Geeks";
```

```
String s2 = "forGeeks";
```

```
String output = s1.concat(s2); // returns "GeeksforGeeks"
```

17. int indexOf (String s): Returns the index within the string of the first occurrence of the specified string.

```
String s = "Learn Share Learn";
```

```
int output = s.indexOf("Share"); // returns 6
```

18. int indexOf (String s, int i): Returns the index within the string of the first occurrence of the specified string, starting at the specified index.

```
String s = "Learn Share Learn";
```

```
int output = s.indexOf('a',3); // returns 8
```

19. Int lastIndexOf(int ch): Returns the index within the string of the last occurrence of the specified string.

```
String s = "Learn Share Learn";
```

```
int output = s.lastIndexOf('a'); // returns 14
```

20. boolean equals(Object otherObj): Compares this string to the specified object.

```
Boolean out = "Geeks".equals("Geeks"); // returns true
```

Boolean out = "Geeks".equals("geeks"); // returns false

21. boolean equalsIgnoreCase (String anotherString): Compares string to another string, ignoring case considerations.

Boolean out= "Geeks".equalsIgnoreCase("Geeks"); // returns true

Boolean out = "Geeks".equalsIgnoreCase("geeks"); // returns true

22. int compareTo(String anotherString): Compares two string lexicographically.

int out = s1.compareTo(s2); // where s1 and s2 are

// strings to be compared

This returns difference s1-s2. If :

out < 0 // s1 comes before s2

out = 0 // s1 and s2 are equal.

out >0 // s1 comes after s2.

23. int compareToIgnoreCase(String anotherString): Compares two string lexicographically, ignoring case considerations.

int out = s1.compareToIgnoreCase(s2);

// where s1 and s2 are

// strings to be compared

This returns difference s1-s2. If :

out < 0 // s1 comes before s2

out = 0 // s1 and s2 are equal.

out >0 // s1 comes after s2.

Note- In this case, it will not consider case of a letter (it will ignore whether it is uppercase or lowercase).

24. String toLowerCase(): Converts all the characters in the String to lower case.

String word1 = "HeLLo";

String word3 = word1.toLowerCase(); // returns "hello"

25. String toUpperCase(): Converts all the characters in the String to upper case.

```
String word1 = "HeLLo";
```

```
String word2 = word1.toUpperCase(); // returns "HELLO"
```

26. String trim(): Returns the copy of the String, by removing whitespaces at both ends. It does not affect whitespaces in the middle.

```
String word1 = " Learn Share Learn ";
```

```
String word2 = word1.trim(); // returns "Learn Share Learn"
```

27. String replace (char oldChar, char newChar): Returns new string by replacing all occurrences of *oldChar* with *newChar*.

```
String s1 = "feeksforfeeks";
```

```
String s2 = "feeksforfeeks".replace('f', 'g'); // returns  
"geeksgorgeeks"
```

Note:- s1 is still feeksforfeeks and s2 is geeksgorgeeks

Program to illustrate all string methods:

```
// Java code to illustrate different constructors and methods
```

```
// String class.
```

```
import java.io.*;
```

```
import java.util.*;
```

```
class Test
```

```
{
```

```
    public static void main (String[] args)
```

```
    {
```

```
        String s= "GeeksforGeeks";
```

```
        // or String s= new String ("GeeksforGeeks");
```

```
        // Returns the number of characters in the String.
```

```
        System.out.println("String length = " + s.length());
```

```
        // Returns the character at ith index.
```

```
        System.out.println("Character at 3rd position = "+ s.charAt(3));
```

```
// Return the substring from the ith index character  
// to end of string
```

```
System.out.println("Substring " + s.substring(3));
```

```
// Returns the substring from i to j-1 index.
```

```
System.out.println("Substring = " + s.substring(2,5));
```

```
// Concatenates string2 to the end of string1.
```

```
String s1 = "Geeks";
```

```
String s2 = "forGeeks";
```

```
System.out.println("Concatenated string = " +  
s1.concat(s2));
```

```
// Returns the index within the string
```

```
// of the first occurrence of the specified string.
```

```
String s4 = "Learn Share Learn";
```

```
System.out.println("Index of Share " +  
s4.indexOf("Share"));
```

```
// Returns the index within the string of the
```

```
// first occurrence of the specified string,
```

```
// starting at the specified index.
```

```
System.out.println("Index of a = " +  
s4.indexOf('a',3));
```

```
// Checking equality of Strings
```

```
Boolean out = "Geeks".equals("geeks");
```

```
System.out.println("Checking Equality " + out);
```

```
out = "Geeks".equals("Geeks");
```

```
System.out.println("Checking Equality " + out);
```

```
out = "Geeks".equalsIgnoreCase("gEeks ");
```

```
System.out.println("Checking Equality" + out);
```

```
int out1 = s1.compareTo(s2);
```

```
System.out.println("If s1 = s2" + out);
```

```

// Converting cases
String word1 = "GeeKyMe";
System.out.println("Changing to lower Case " +
                    word1.toLowerCase());

// Converting cases
String word2 = "GeekyME";
System.out.println("Changing to UPPER Case " +
                    word1.toUpperCase());

// Trimming the word
String word4 = " Learn Share Learn ";
System.out.println("Trim the word " + word4.trim());

// Replacing characters
String str1 = "feeksforfeeks";
System.out.println("Original String " + str1);
String str2 = "feeksforfeeks".replace('f','g') ;
System.out.println("Replaced f with g -> " + str2);
}
}

```

Output :

String length = 13
 Character at 3rd position = k
 Substring ksforGeeks
 Substring = eks
 Concatenated string = GeeksforGeeks
 Index of Share 6
 Index of a = 8
 Checking Equality false
 Checking Equality true

 Checking Equalityfalse
 If s1 = s2false

Changing to lower Case geekyme
Changing to UPPER Case GEEKYME
Trim the word Learn Share Learn
Original String feeksforfeeks
Replaced f with g -> geeksgorgeeks