Todays agenda : 12-04-2024

String Handling

String Manipulation: Methods in the String Class

StringBuilder and StringBuffer: Mutability and Performance

String Manipulation: Methods in the String Class

***Strings: ‘a’….”aaaaa”…literals***

**String** is a sequence of characters, for e.g. “Hello” is a string of 5 characters. In java, string is an immutable object which means it is constant and cannot be changed once it has been created. In this tutorial we will learn about String class and String methods

## Creating a String

There are two ways to create a String in Java

1. String literal : String pool
2. Using new keyword // String str = new String(“java”); ….. heap memory :str ; java will be created in string pool

### String literal

In java, Strings can be created like this: Assigning a String literal to a String instance:

String str1 = "Welcome";

String str2 = "Welcome";

**The problem with this approach**: that String is an object in Java. However we have not created any string object using new keyword above. The compiler does that task for us it creates a string object having the string literal (that we have provided, in this case it is “Welcome” and assigns it to the provided string instances.

**But** if the object already exists in the memory it does not create a new Object rather it assigns the same old object to the new instance, that means even though we have two string instances above(str1 and str2) compiler only created one string object (having the value “Welcome”) and assigned the same to both the instances. For example there are 10 string instances that have same value, it means that in memory there is only one object having the value and all the 10 string instances would be pointing to the same object.

What if we want to have two different objects with the same string? For that we would need to create strings using **new keyword**.

### Using New Keyword

As we saw above that when we tried to assign the same string object to two different literals, compiler only created one object and made both of the literals to point the same object. To overcome that approach we can create strings like this:

String str1 = new String("Welcome");

String str2 = new String("Welcome");

In this case compiler would create two different objects in memory having the same string.

## A Simple Java String Example

public class Example{

public static void main(String args[]){

//creating a string by java string literal

String str = "Example"; //string pool

char arrch[]={'E','x','a','m','p',’l’,’e’};

//converting char array arrch[] to string str2

String str2 = new String(arrch); //heap

//creating another java string str3 by using new keyword

String str3 = new String("Java String Example"); //heap

//Displaying all the three strings

System.out.println(str);

System.out.println(str2);

System.out.println(str3);

}

}

Output:

Example

Example

Java String Example

## Java String Methods

Here is the list of the methods available in the Java String class. These methods are explained in the separate tutorials with the help of examples.

# Below are also the methods provided by string please go throught and solve examples with it:

1. [char charAt(int index)](https://beginnersbook.com/2013/12/java-string-charat-method-example/): It returns the character at the specified index. Specified index value should be between 0 to length() -1 both inclusive. It throws IndexOutOfBoundsException if index<0||>= length of String.
2. [boolean equals(Object obj)](https://beginnersbook.com/2013/12/java-string-equals-and-equalsignorecase-methods-example/): Compares the string with the specified string and returns true if both matches else false.
3. [boolean equalsIgnoreCase(String string)](https://beginnersbook.com/2013/12/java-string-equals-and-equalsignorecase-methods-example/): It works same as equals method but it doesn’t consider the case while comparing strings. It does a case insensitive comparison.
4. [int compareTo(String string)](https://beginnersbook.com/2013/12/java-string-compareto-method-example/): This method compares the two strings based on the Unicode value of each character in the strings.
5. [int compareToIgnoreCase(String string)](https://beginnersbook.com/2013/12/java-string-comparetoignorecase-method-example/): Same as CompareTo method however it ignores the case during comparison.
6. [boolean startsWith(String prefix, int offset)](https://beginnersbook.com/2013/12/java-string-startswith-method-example/): It checks whether the substring (starting from the specified offset index) is having the specified prefix or not.
7. [boolean startsWith(String prefix)](https://beginnersbook.com/2013/12/java-string-startswith-method-example/): It tests whether the string is having specified prefix, if yes then it returns true else false.
8. [boolean endsWith(String suffix)](https://beginnersbook.com/2013/12/java-string-endswith-method-example/): Checks whether the string ends with the specified suffix.
9. [int indexOf(int ch)](https://beginnersbook.com/2013/12/java-string-indexof-method-example/): Returns the index of first occurrence of the specified character ch in the string.
10. [int indexOf(int ch, int fromIndex)](https://beginnersbook.com/2013/12/java-string-indexof-method-example/): Same as indexOf method however it starts searching in the string from the specified fromIndex.
11. [int lastIndexOf(int ch)](https://beginnersbook.com/2013/12/java-string-lastindexof-method-example/): It returns the last occurrence of the character ch in the string.
12. [int lastIndexOf(int ch, int fromIndex)](https://beginnersbook.com/2013/12/java-string-lastindexof-method-example/): Same as lastIndexOf(int ch) method, it starts search from fromIndex.
13. [int indexOf(String str)](https://beginnersbook.com/2013/12/java-string-indexof-method-example/): This method returns the index of first occurrence of specified substring str.
14. [int lastindexOf(String str)](https://beginnersbook.com/2013/12/java-string-lastindexof-method-example/): Returns the index of last occurrence of string str.
15. [String substring(int beginIndex)](https://beginnersbook.com/2013/12/java-string-substring-method-example/): It returns the substring of the string. The substring starts with the character at the specified index.
16. [String substring(int beginIndex, int endIndex)](https://beginnersbook.com/2013/12/java-string-substring-method-example/): Returns the substring. The substring starts with character at beginIndex and ends with the character at endIndex.
17. [String concat(String str)](https://beginnersbook.com/2013/12/java-string-concat-method-example/): Concatenates the specified string “str” at the end of the string.
18. [String replace(char oldChar, char newChar)](https://beginnersbook.com/2013/12/java-string-replace-replacefirst-replaceall-method-examples/): It returns the new updated string after changing all the occurrences of oldChar with the newChar.
19. [boolean contains(CharSequence s)](https://beginnersbook.com/2017/10/java-string-contains-method/): It checks whether the string contains the specified sequence of char values. If yes then it returns true else false. It throws NullPointerException of ‘s’ is null.
20. [String toUpperCase(Locale locale)](https://beginnersbook.com/2013/12/java-string-tolowercase-method-example/): Converts the string to upper case string using the rules defined by specified locale.
21. [String toUpperCase()](https://beginnersbook.com/2013/12/java-string-tolowercase-method-example/): Equivalent to toUpperCase(Locale.getDefault()).
22. [public String intern()](https://beginnersbook.com/2017/10/java-string-intern-method/): This method searches the specified string in the memory pool and if it is found then it returns the reference of it, else it allocates the memory space to the specified string and assign the reference to it.
23. [public boolean isEmpty()](https://beginnersbook.com/2017/10/java-string-isempty-method-with-example/): This method returns true if the given string has 0 length. If the length of the specified Java String is non-zero then it returns false.
24. [public static String join()](https://beginnersbook.com/2017/10/java-string-join-method/): This method joins the given strings using the specified delimiter and returns the concatenated Java String
25. [String replaceFirst(String regex, String replacement)](https://beginnersbook.com/2013/12/java-string-replace-replacefirst-replaceall-method-examples/): It replaces the first occurrence of substring that fits the given regular expression “regex” with the specified replacement string.
26. [String replaceAll(String regex, String replacement)](https://beginnersbook.com/2013/12/java-string-replace-replacefirst-replaceall-method-examples/): It replaces all the occurrences of substrings that fits the [regular expression regex](https://beginnersbook.com/2014/08/java-regex-tutorial/) with the replacement string.
27. [String[] split(String regex, int limit)](https://beginnersbook.com/2013/12/java-string-split-method-example/): It splits the string and returns the array of substrings that matches the given regular expression. limit is a result threshold here.
28. [String[] split(String regex)](https://beginnersbook.com/2013/12/java-string-split-method-example/): Same as split(String regex, int limit) method however it does not have any threshold limit.
29. [String toLowerCase(Locale locale)](https://beginnersbook.com/2013/12/java-string-tolowercase-method-example/): It converts the string to lower case string using the rules defined by given locale.
30. [public static String format()](https://beginnersbook.com/2017/10/java-string-format-method/): This method returns a formatted java String
31. [String toLowerCase()](https://beginnersbook.com/2013/12/java-string-tolowercase-method-example/): Equivalent to toLowerCase(Locale. getDefault()).
32. [String trim()](https://beginnersbook.com/2013/12/java-string-trim-and-hashcode-methods/): Returns the substring after omitting leading and trailing white spaces from the original string.
33. [char[] toCharArray()](https://beginnersbook.com/2013/12/java-string-tochararray-method-example/): Converts the string to a character array.
34. [static String copyValueOf(char[] data)](https://beginnersbook.com/2013/12/java-string-copyvalueof-method-example/): It returns a string that contains the characters of the specified character array.
35. [static String copyValueOf(char[] data, int offset, int count)](https://beginnersbook.com/2013/12/java-string-copyvalueof-method-example/): Same as above method with two extra arguments – initial offset of subarray and length of subarray.
36. [void getChars(int srcBegin, int srcEnd, char[] dest, int destBegin)](https://beginnersbook.com/2013/12/java-string-getchars-method-example/): It copies the characters of **src** array to the **dest** array. Only the specified range is being copied(srcBegin to srcEnd) to the dest subarray(starting fromdestBegin).
37. [static String valueOf()](https://beginnersbook.com/2017/10/java-string-valueof-method/): This method returns a string representation of passed arguments such as int, long, float, double, char and char array.
38. [boolean contentEquals(StringBuffer sb)](https://beginnersbook.com/2013/12/java-string-contentequals-method-example/): It compares the string to the specified string buffer.
39. [int length()](https://beginnersbook.com/2013/12/java-string-length-method-example/): It returns the length of a String.

|  |  |  |
| --- | --- | --- |
| **Modifier and Type** | **Method** | **Description** |
| public synchronized StringBuffer | append(String s) | is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc. |
| public synchronized StringBuffer | insert(int offset, String s) | is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc. |
| public synchronized StringBuffer | replace(int startIndex, int endIndex, String str) | is used to replace the string from specified startIndex and endIndex. |
| public synchronized StringBuffer | delete(int startIndex, int endIndex) | is used to delete the string from specified startIndex and endIndex. |
| public synchronized StringBuffer | reverse() | is used to reverse the string. |
| public int | capacity() | is used to return the current capacity. |
| public char | charAt(int index) | is used to return the character at the specified position. |
| public int | length() | is used to return the length of the string i.e. total number of characters. |
| public String | substring(int beginIndex) | is used to return the substring from the specified beginIndex. |
| public String | substring(int beginIndex, int endIndex) | is used to return the substring from the specified beginIndex and endIndex. |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Java StringBuffer class

Java StringBuffer class is used to create mutable (modifiable) string. The StringBuffer class in java is same as String class except it is mutable i.e. it can be changed.

A string that can be modified or changed is known as mutable string. StringBuffer and StringBuilder classes are used for creating mutable string.

# Here are some important features and methods of the StringBuffer class:

# StringBuffer objects are mutable, meaning that you can change the contents of the buffer without creating a new object.

# The initial capacity of a StringBuffer can be specified when it is created, or it can be set later with the ensureCapacity() method.

# The append() method is used to add characters, strings, or other objects to the end of the buffer.

# The insert() method is used to insert characters, strings, or other objects at a specified position in the buffer.

# The delete() method is used to remove characters from the buffer.

# The reverse() method is used to reverse the order of the characters in the buffer.

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Java StringBuilder class

Java StringBuilder class is used to create mutable (modifiable) string. The Java StringBuilder class is same as StringBuffer class except that it is non-synchronized. It is available since JDK 1.5.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public StringBuilder append(String s) | is used to append the specified string with this string. The append() method is overloaded like append(char), append(boolean), append(int), append(float), append(double) etc. |
| public StringBuilder insert(int offset, String s) | is used to insert the specified string with this string at the specified position. The insert() method is overloaded like insert(int, char), insert(int, boolean), insert(int, int), insert(int, float), insert(int, double) etc. |
| public StringBuilder replace(int startIndex, int endIndex, String str) | is used to replace the string from specified startIndex and endIndex. |
| public StringBuilder delete(int startIndex, int endIndex) | is used to delete the string from specified startIndex and endIndex. |
| public StringBuilder reverse() | is used to reverse the string. |
| public char charAt(int index) | is used to return the character at the specified position. |
| public int length() | is used to return the length of the string i.e. total number of characters. |
| public String substring(int beginIndex) | is used to return the substring from the specified beginIndex. |
| public String substring(int beginIndex, int endIndex) | is used to return the substring from the specified beginIndex and endIndex. |