

Java String

In Java, string is basically an object that represents sequence of char values. An array of characters works same as Java string. For example:

```
char[] ch={'j','a','v','a','t','p','o','i','n','t'};  
String s=new String(ch);
```

is same as:

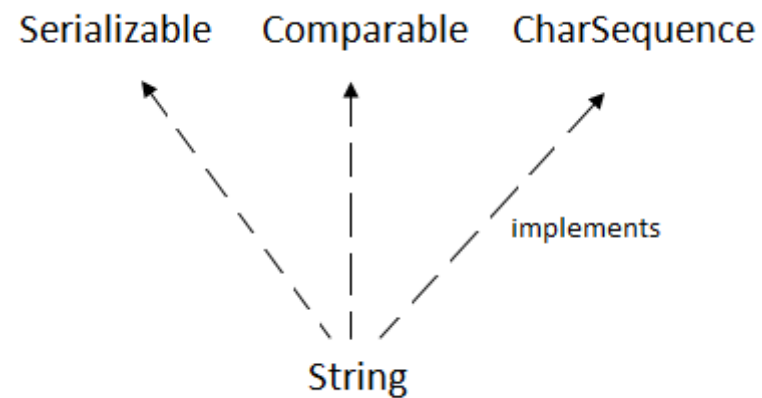
```
String s="javatpoint";
```

Java String class provides a lot of methods to perform operations on strings such as `compare()`, `concat()`, `equals()`, `split()`, `length()`, `replace()`, `compareTo()`, `intern()`, `substring()` etc.





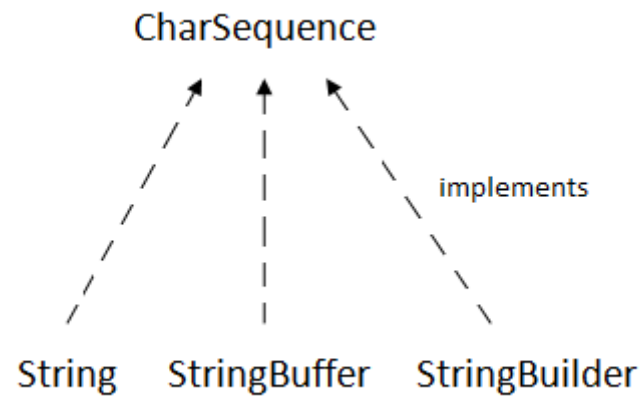
The `java.lang.String` class implements *Serializable*, *Comparable* and *CharSequence* interfaces.



CharSequence Interface

The `CharSequence` interface is used to represent the sequence of characters. `String`, `StringBuffer` and `StringBuilder` classes implement it. It means, we can create strings in java by using these three classes.





The Java String is immutable which means it cannot be changed. Whenever we change any string, a new instance is created. For mutable strings, you can use `StringBuffer` and `StringBuilder` classes.

We will discuss immutable string later. Let's first understand what is String in Java and how to create the String object.



What is String in java

Generally, String is a sequence of characters. But in Java, string is an object that represents a sequence of characters. The `java.lang.String` class is used to create a string object.

How to create a string object?

There are two ways to create String object:

1. By string literal
2. By new keyword

1) String Literal

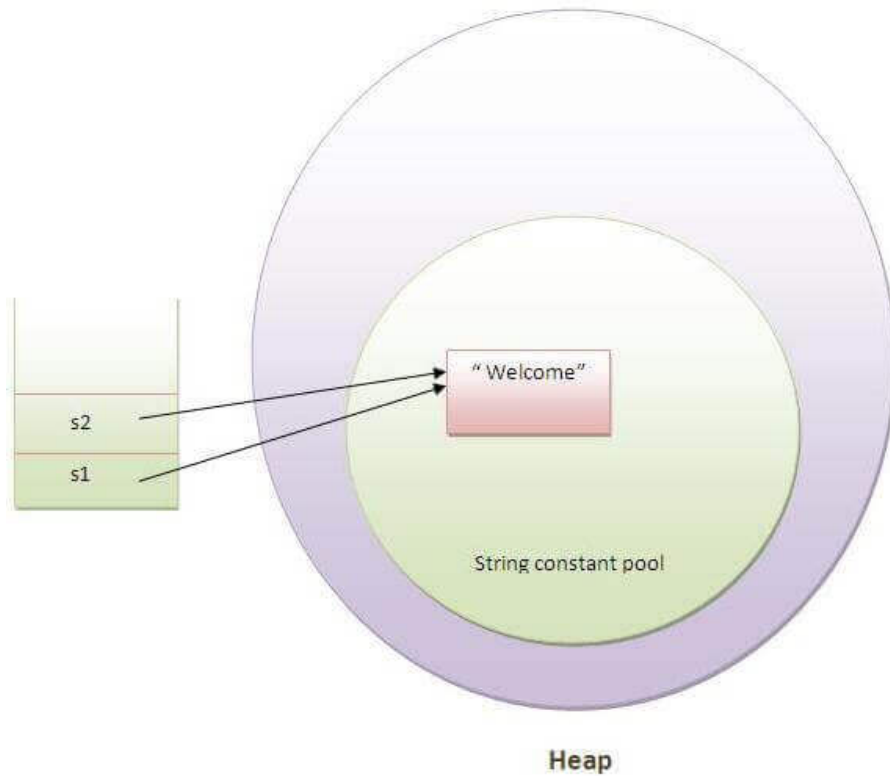
Java String literal is created by using double quotes. For Example:

```
String s="welcome";
```

Each time you create a string literal, the JVM checks the "string constant pool" first. If the string already exists in the pool, a reference to the pooled instance is returned. If the string doesn't exist in the pool, a new string instance is created and placed in the pool. For example:

```
String s1="Welcome";  
String s2="Welcome";//It doesn't create a new instance
```





In the above example, only one object will be created. Firstly, JVM will not find any string object with the value "Welcome" in string constant pool, that is why it will create a new object. After that it will find the string with the value "Welcome" in the pool, it will not create a new object but will return the reference to the same instance.

Note: String objects are stored in a special memory area known as the "string constant pool".

Why Java uses the concept of String literal?



To make Java more memory efficient (because no new objects are created if it exists already in the string constant pool).

2) By new keyword

```
String s=new String("Welcome");//creates two objects and one reference variable
```

In such case, JVM will create a new string object in normal (non-pool) heap memory, and the literal "Welcome" will be placed in the string constant pool. The variable s will refer to the object in a heap (non-pool).

Java String Example

```
public class StringExample{  
    public static void main(String args[]){  
        String s1="java";//creating string by java string literal  
        char ch={'s','t','r','i','n','g','s'};  
        String s2=new String(ch);//converting char array to string  
        String s3=new String("example");//creating java string by new keyword  
        System.out.println(s1);  
        System.out.println(s2);  
        System.out.println(s3);  
    }  
}
```

Test it Now

```
java  
strings
```



```
example
```

Java String class methods

The java.lang.String class provides many useful methods to perform operations on sequence of char values.

No.	Method	Description
1	char charAt(int index)	returns char value for the particular index
2	int length()	returns string length
3	static String format(String format, Object... args)	returns a formatted string.
4	static String format(Locale l, String format, Object... args)	returns formatted string with given locale.
5	String substring(int beginIndex)	returns substring for given begin index.
6	String substring(int beginIndex, int endIndex)	returns substring for given begin index and end index.
7	boolean contains(CharSequence s)	returns true or false after matching the sequence of char value.
8	static String join(CharSequence delimiter, CharSequence... elements)	returns a joined string.
9	static String join(CharSequence delimiter, Iterable<? extends CharSequence> elements)	returns a joined string.



10	<code>boolean equals(Object another)</code>	checks the equality of string with the given object.
11	<code>boolean isEmpty()</code>	checks if string is empty.
12	<code>String concat(String str)</code>	concatenates the specified string.
13	<code>String replace(char old, char new)</code>	replaces all occurrences of the specified char value.
14	<code>String replace(CharSequence old, CharSequence new)</code>	replaces all occurrences of the specified CharSequence.
15	<code>static String equalsIgnoreCase(String another)</code>	compares another string. It doesn't check case.
16	<code>String[] split(String regex)</code>	returns a split string matching regex.
17	<code>String[] split(String regex, int limit)</code>	returns a split string matching regex and limit.
18	<code>String intern()</code>	returns an interned string.
19	<code>int indexOf(int ch)</code>	returns the specified char value index.
20	<code>int indexOf(int ch, int fromIndex)</code>	returns the specified char value index starting with given index.
21	<code>int indexOf(String substring)</code>	returns the specified substring index.
22	<code>int indexOf(String substring, int fromIndex)</code>	returns the specified substring index starting with given index.



23	String toLowerCase()	returns a string in lowercase.
24	String toLowerCase(Locale l)	returns a string in lowercase using specified locale.
25	String toUpperCase()	returns a string in uppercase.
26	String toUpperCase(Locale l)	returns a string in uppercase using specified locale.
27	String trim()	removes beginning and ending spaces of this string.
28	static String valueOf(int value)	converts given type into string. It is an overloaded method.

Do You Know?

- Why are String objects immutable?
- How to create an immutable class?
- What is string constant pool?
- What code is written by the compiler if you concatenate any string by + (string concatenation operator)?
- What is the difference between StringBuffer and StringBuilder class?

What will we learn in String Handling?

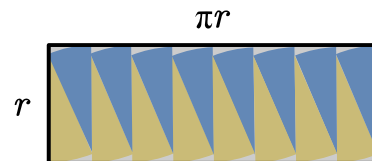
- Concept of String
- Immutable String



- String Comparison
- String Concatenation
- Concept of Substring
- String class methods and its usage
- StringBuffer class
- StringBuilder class
- Creating Immutable class
- toString() method
- StringTokenizer class

next →

Help Others, Please Share



Area = πr^2

BRILLIANT


Learn Latest Tutorials



Proc*C


 social media
marketing
SMM

 GDB
GDB


 Fuzzy Logic
Fuzzy Logic


 DHTML
DHTML

 Google
Classroom
Classroom


 autocad
Tutorial
AutoCad

 kubernetes
Tutorial
Kubernetes


 Openpyxl
Tutorial
Openpyxl


 Tally
Tutorial
Tally


 Godot
Tutorial
Godot


 Spring Boot
Tutorial
Spring Boot


Preparation

 Aptitude
Aptitude

 Logical
Reasoning
Reasoning













 Verbal
Ability
Verbal A.

 Interview
Questions
Interview











 Company
Interview
Questions
Company

Trending Technologies



 Artificial Intelligence Tutorial AI	 AWS Tutorial AWS	 Selenium tutorial Selenium	 Cloud tutorial Cloud	 Hadoop tutorial Hadoop
 ReactJS Tutorial ReactJS	 Data Science Tutorial D. Science	 Angular 7 Tutorial Angular 7	 Blockchain Tutorial Blockchain	 Git Tutorial Git
 Machine Learning Tutorial ML	 DevOps Tutorial DevOps			

B.Tech / MCA

 DBMS tutorial DBMS	 Data Structures tutorial DS	 DAA tutorial DAA	 Operating System tutorial OS	 Computer Network tutorial C. Network
 Compiler Design tutorial Compiler D.	 Computer Organization and Architecture COA	 Discrete Mathematics Tutorial D. Math.	 Ethical Hacking Tutorial E. Hacking	 Computer Graphics Tutorial C. Graphics





Software
Engineering
Tutorial

Software E.



html tutorial
Web Tech.



Cyber
Security
tutorial

Cyber Sec.



Automata
Tutorial
Automata



C Language
tutorial
C



C++ tutorial
C++



Java tutorial
Java



.Net
Framework
tutorial
.Net



Python
tutorial
Python



List of
Programs
Programs



Control
Systems
tutorial
Control S.



Data Mining
Tutorial
Data Mining