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:

1. String s="welcome";

### 2) By new keyword

1. String s=**new** String("Welcome");

Java String Example

**StringExample.java**

**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);

}}

# **Immutable String in Java**

A String is an unavoidable type of variable while writing any application program. String references are used to store various attributes like username, password, etc. In Java, **String objects are immutable**. Immutable simply means unmodifiable or unchangeable.

Once String object is created its data or state can't be changed but a new String object is created.

Let's try to understand the concept of immutability by the example given below:

**class** Testimmutablestring{

**public** **static** **void** main(String args[]){

   String s="Sachin";

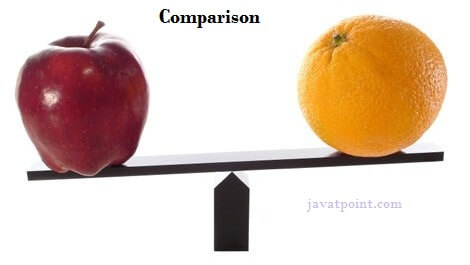
   s.concat(" Tendulkar");

  System.out.println(s);

 }

}

# **Java String compare**



We can compare String in Java on the basis of content and reference.

It is used in **authentication** (by equals() method), **sorting** (by compareTo() method), **reference matching** (by == operator) etc.

There are three ways to compare String in Java:

1. By Using equals() Method
2. By Using == Operator
3. By compareTo() Method

## **1) By Using equals() Method**

The String class equals() method compares the original content of the string. It compares values of string for equality. String class provides the following two methods:

* **public boolean equals(Object another)** compares this string to the specified object.
* **public boolean equalsIgnoreCase(String another)** compares this string to another string, ignoring case.

**Teststringcomparison1.java**

**class** Teststringcomparison1{

**public** **static** **void** main(String args[]){

   String s1="Sachin";

   String s2="Sachin";

   String s3=**new** String("Sachin");

   String s4="Saurav";

   System.out.println(s1.equals(s2));//true

   System.out.println(s1.equals(s3));//true

   System.out.println(s1.equals(s4));//false

 }

}

**Teststringcomparison2.java**

**class** Teststringcomparison2{

**public** **static** **void** main(String args[]){

   String s1="Sachin";

  String s2="SACHIN";

   System.out.println(s1.equals(s2));//false

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

 }

}

**Teststringcomparison3.java**

**class** Teststringcomparison3{

**public** **static** **void** main(String args[]){

  String s1="Sachin";

   String s2="Sachin";

   String s3=**new** String("Sachin");

   System.out.println(s1==s2);

   System.out.println(s1==s3);

 }

}

# **String Concatenation in Java**

In Java, String concatenation forms a new String that is the combination of multiple strings. There are two ways to concatenate strings in Java:

1. By + (String concatenation) operator
2. By concat() method

## **1) String Concatenation by + (String concatenation) operator**

Java String concatenation operator (+) is used to add strings. For Example:

**TestStringConcatenation1.java**

**class** TestStringConcatenation1{

**public** **static** **void** main(String args[]){

   String s="Sachin"+" Tendulkar";

   System.out.println(s);//Sachin Tendulkar

 }

}

2) String Concatenation by concat() method

The String concat() method concatenates the specified string to the end of current string. Syntax:

1. **public** String concat(String another)

Let's see the example of String concat() method.

**TestStringConcatenation3.java**

**class** TestStringConcatenation3{

**public** **static** **void** main(String args[]){

   String s1="Sachin ";

   String s2="Tendulkar";

   String s3=s1.concat(s2);

   System.out.println(s3);//Sachin Tendulkar

  }

}

1. String concatenation using StringBuilder class

StringBuilder is class provides append() method to perform concatenation operation. The append() method accepts arguments of different types like Objects, StringBuilder, int, char, CharSequence, boolean, float, double. StringBuilder is the most popular and fastet way to concatenate strings in Java. It is mutable class which means values stored in StringBuilder objects can be updated or changed.

**StrBuilder.java**

**public** **class** StrBuilder

{

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

    {

        StringBuilder s1 = **new** StringBuilder("Hello");

        StringBuilder s2 = **new** StringBuilder(" World");

        StringBuilder s = s1.append(s2);

            System.out.println(s.toString());

    }

}

2. String concatenation using format() method

String.format() method allows to concatenate multiple strings using format specifier like %s followed by the string values or objects.

**StrFormat.java**

**public** **class** StrFormat

{

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

    {

        String s1 = **new** String("Hello");

        String s2 = **new** String(" World");

        String s = String.format("%s%s",s1,s2);

            System.out.println(s.toString());

    }

}

3. String concatenation using String.join() method (Java Version 8+)

The String.join() method is available in Java version 8 and all the above versions. String.join() method accepts arguments first a separator and an array of String objects.

**StrJoin.java:**

**public** **class** StrJoin

{

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

    {

        String s1 = **new** String("Hello");

        String s2 = **new** String(" World");

        String s = String.join("",s1,s2);

            System.out.println(s.toString());

    }

}

4. String concatenation using StringJoiner class (Java Version 8+)

StringJoiner class has all the functionalities of String.join() method. In advance its constructor can also accept optional arguments, prefix and suffix.

**StrJoiner.java**

**public** **class** StrJoiner

{

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

    {

        StringJoiner s = **new** StringJoiner(", ");

        s.add("Hello");

        s.add("World");

        System.out.println(s.toString());

    }

}

# **Substring in Java**

A part of String is called **substring**. In other words, substring is a subset of another String. Java String class provides the built-in substring() method that extract a substring from the given string by using the index values passed as an argument. In case of substring() method startIndex is inclusive and endIndex is exclusive.

Suppose the string is "**computer**", then the substring will be com, compu, ter, etc.

1. **public String substring(int startIndex):**  
   This method returns new String object containing the substring of the given string from specified startIndex (inclusive). The method throws an IndexOutOfBoundException when the startIndex is larger than the length of String or less than zero.
2. **public String substring(int startIndex, int endIndex):**  
   This method returns new String object containing the substring of the given string from specified startIndex to endIndex. The method throws an IndexOutOfBoundException when the startIndex is less than zero or startIndex is greater than endIndex or endIndex is greater than length of String.

In case of String:

Example of Java substring() method

**TestSubstring.java**

**public** **class** TestSubstring{

**public** **static** **void** main(String args[]){

 String s="SachinTendulkar";

 System.out.println("Original String: " + s);

 System.out.println("Substring starting from index 6: " +s.substring(6));

 System.out.println("Substring starting from index 0 to 6: "+s.substring(0,6));

 }

}

# **Java String Class Methods**

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

Java String is a powerful concept because everything is treated as a String if you submit any form in window based, web based or mobile application.

Let's use some important methods of String class.

### Java String toUpperCase() and toLowerCase() method

The Java String toUpperCase() method converts this String into uppercase letter and String toLowerCase() method into lowercase letter.

**Stringoperation1.java**

**public** **class** Stringoperation1

{

**public** **static** **void** main(String ar[])

{

String s="Sachin";

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

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

System.out.println(s);

}

}

Java String trim() method

The String class trim() method eliminates white spaces before and after the String.

**Stringoperation2.java**

**public** **class** Stringoperation2

{

**public** **static** **void** main(String ar[])

{

String s="  Sachin  ";

System.out.println(s);

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

}

}

Java String charAt() Method

The String class charAt() method returns a character at specified index.

**Stringoperation4.java**

**public** **class** Stringoperation4

{

**public** **static** **void** main(String ar[])

{

String s="Sachin";

System.out.println(s.charAt(0));

System.out.println(s.charAt(3));

}

}

### Java String length() Method

The String class length() method returns length of the specified String.

**Stringoperation5.java**

**public** **class** Stringoperation5

{

**public** **static** **void** main(String ar[])

{

String s="Sachin";

System.out.println(s.length());/

}

}

ava String replace() Method

The String class replace() method replaces all occurrence of first sequence of character with second sequence of character.

**Stringoperation8.java**

**public** **class** Stringoperation8

{

**public** **static** **void** main(String ar[])

{

String s1="Java is a programming language. Java is a platform. Java is an Island.";

String replaceString=s1.replace("Java","Kava");

System.out.println(replaceString);

}

}