### **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.

**2) By new keyword**

String s=new String("Welcome");

 //creates two objects and one reference variable

In such case, [JVM](https://www.javatpoint.com/jvm-java-virtual-machine) 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 Ex

{

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);

}

}

java

strings

example

**Java String class methods**

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

|  |  |
| --- | --- |
| Method | Description |
| [**char charAt(int index)**](https://www.javatpoint.com/java-string-charat) | returns char value for the particular index |
| [**int length()**](https://www.javatpoint.com/java-string-length) | returns string length |
| [**String substring(int beginIndex, int endIndex)**](https://www.javatpoint.com/java-string-substring) | returns substring for given begin index and end index. |
| [**boolean equals(Object another)**](https://www.javatpoint.com/java-string-equals) | checks the equality of string with the given object. |
| [**String concat(String str)**](https://www.javatpoint.com/java-string-concat) | concatenates the specified string. |
| [**static String equalsIgnoreCase(String another)**](https://www.javatpoint.com/java-string-equalsignorecase) | compares another string. It doesn't check case. |
| [**String toLowerCase()**](https://www.javatpoint.com/java-string-tolowercase) | returns a string in lowercase. |
| [**String toLowerCase(Locale l)**](https://www.javatpoint.com/java-string-tolowercase) | returns a string in lowercase using specified locale. |
| [**String toUpperCase()**](https://www.javatpoint.com/java-string-touppercase) | returns a string in uppercase. |
| [**String toUpperCase(Locale l)**](https://www.javatpoint.com/java-string-touppercase) | returns a string in uppercase using specified locale. |
| [**static String valueOf(int value)**](https://www.javatpoint.com/java-string-valueof) | converts given type into string. It is an overloaded method. |

# **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 equals() method
2. By = = operator
3. By compareTo() method

## **1) equals() method**

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.

* **equals(Object another)** compares this string to the specified object.

class Ex

{

 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

 }

}

Output:

 true

 true

 false

## **equalsIgnoreCase() method**

class Ex

{

 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

 }

}

Output:

false

true

## **2) by == operator**

The = = operator compares references not values.

class Ex

{

 public static void main(String args[])

{

   String s1="Sachin";

   String s2="Sachin";

   String s3=new String("Sachin");

   System.out.println(s1==s2);//true (because both refer to same instance)

   System.out.println(s1==s3);//false(because s3 refers to instance created in nonpool)

 }

}

Output:

 true

 false

## **3) compareTo() method**

The String compareTo() method compares values lexicographically and returns an integer value that describes if first string is less than, equal to or greater than second string.

Suppose s1 and s2 are two string variables. If:

* **s1 == s2** :0
* **s1 > s2**  :positive value
* **s1 < s2**  :negative value

class Ex

{

 public static void main(String args[])

{

   String s1="Sachin";

   String s2="Sachin";

   String s3="Ratan";

   System.out.println(s1.compareTo(s2));//0

   System.out.println(s1.compareTo(s3));//1(because s1>s3)

   System.out.println(s3.compareTo(s1));//-1(because s3 < s1 )

 }

}

Output:

 0

 1

 -1

# **String Concatenation in Java**

In java, string concatenation forms a new string that is the combination of multiple strings. There are two ways to concat string 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:

class Ex

{

 public static void main(String args[])

{

   String s="Sachin"+" Tendulkar";

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

 }

}

Output:Sachin Tendulkar

**Example**:

class Ex

{

 public static void main(String args[])

{

   String s=50+30+"Sachin"+40+40;

   System.out.println(s);//80Sachin4040

 }

}

80Sachin4040

### **2) String Concatenation by concat() method**

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

 concat(String another)

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

class Ex

{

 public static void main(String args[])

{

   String s1="Sachin ";

   String s2="Tendulkar";

   String s3=s1.concat(s2);

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

  }

}

Sachin Tendulkar