String Comparison in Java

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:

- By = = operator
- By equals() method
- By compareTo() method

Let's start with understanding == operator:

The = = operator compares **references not values**.

Example1

```
class Demo
{
    public static void main(String[] args)
    {
        String s1 = new String("JAVA");
        String s2 = new String("JAVA");

        if(s1==s2)
        {
            System.out.println("String references are equal.");
        }
        else
        {
            System.out.println("String references are not equal.");
        }
    }
}
```

Output: **String references are not equal.**

Whenever **new** keyword is used to create string, it gets created in non-constant pool. And when == operator is used to compare two strings its references are compared and not the values. And as s1 has different reference than s2, we get the output as references are not equal.

Now let's see comparison by equals() method:

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

Example2

```
class Demo
{
    public static void main(String[] args)
    {
        String s1 = new String("JAVA");
        String s2 = new String("JAVA");

        if(s1.equals(s2)==true)
        {
            System.out.println("String references are equal.");
        }
        else
        {
            System.out.println("String references are not equal.");
        }
    }
}
```

Output: **String values are equal.**

Example3

```
class Demo
{
    public static void main(String[] args)
    {
        String s1 = "java";
        String s2 = new String("JAVA");

        if(s1.equals(s2)==true)
        {
            System.out.println("String values are equal.");
        }
        else
        {
            System.out.println("String values are not equal.");
        }
    }
}
```

Output: **String values are not equal.**

Example4


```
class Demo
{
    public static void main(String[] args)
    {
        String s1 = "java";
        String s2 = new String("JAVA");

        if(s1.equalsIgnoreCase(s2)==true)
        {
            System.out.println("String values are equal.");
        }
        else
        {
            System.out.println("String values are not equal.");
        }
    }
}
```

Output: **String values are equal.**

Example5

```
class Demo
{
    public static void main(String[] args)
    {
        String s1 = "JAVA";
        String s2;
        s2 = s1;
        if(s1==s2)
        {
            System.out.println("references are equal.");
        }
        else
        {
            System.out.println("references are not equal.");
        }
    }
}
```

Output: **references are equal.**

Here as new keyword is not used s1 gets created in constant pool. And s2 is reference which gets created in constant pool. So s1 reference is assigned to s2, which means both are referring to same value.

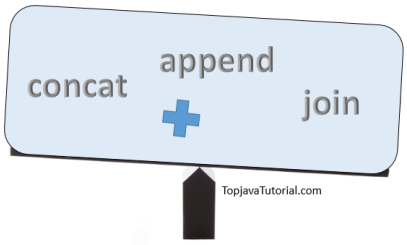
String Concatenation in Java

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

- By + (string concatenation) operator
- By concat() method

String Concatenation by + (string concatenation) operator

Java string concatenation operator (+) is used to add strings.



Example6

```
class Demo
{
    public static void main(String[] args)
    {
        String s1 = "JAVA";
        String s2 = "PYTHON";
        String s3 = "JAVA"+"PYTHON";
        String s4 = "JAVA"+"PYTHON";

        if(s3==s4)
        {
            System.out.println("references are equal.");
        }
        else
        {
            System.out.println("references are not equal.");
        }
    }
}
```

Output: **references are equal.**

As here values are getting added and not the references. And duplication cannot happen in constant pool, therefore references are same.

Example7

```
class Demo
{
    public static void main(String[] args)
    {
        String s1 = "JAVA";
        String s2 = "PYTHON";
        String s3 = s1+s2;
        String s4 = s1+s2;

        if(s3==s4)
        {
            System.out.println("references are equal.");
        }
        else
        {
            System.out.println("references are not equal.");
        }
    }
}
```

Output: **references are not equal.**

As here values are not getting added but references are, because s1 and s2 are not literals. And duplication is possible in non-constant pool, therefore references are not the same.