Concatenating String

There are 2 methods to concatenate two or more string.

1. Using **concat()** method
2. Using + operator

1) Using concat() method

```
string s = "Hello";
string str = "Java";
string str2 = s.concat(str);
String str1 = "Hello".concat("Java");    //works with string literals too.
```

2) Using + operator

```
string str = "Rahul";
string str1 = "Dravid";
string str2 = str + str1;
string st = "Rahul"+"Dravid";
```

String Comparison

String comparison can be done in 3 ways.

1. Using **equals()** method
2. Using == operator
3. By **CompareTo()** method

Using equals() method

equals() method compares two strings for equality. Its general syntax is,

boolean **equals** (Object *str*)

It compares the content of the strings. It will return **true** if string matches, else returns **false**.

```
String s = "Hell";

String s1 = "Hello";

String s2 = "Hello";

s1.equals(s2);    //true

s.equals(s1) ;   //false
```

Using == operator

== Operator compares two object references to check whether they refer to same instance. This also, will return **true** on successful match.

```
String s1 = "Java";

String s2 = "Java";

String s3 = new string ("Java");

test(Sl == s2)     //true

test(s1 == s3)      //false
```

By compareTo() method

compareTo() method compares values and returns an int which tells if the string compared is less than, equal to or greater than th other string. Its general syntax is,

int **compareTo**(String *str*)

To use this function you must implement the **Comparable Interface**. compareTo() is the only function in Comparable Interface.

```
String s1 = "Abhi";
String s2 = "Viraaj";
String s3 = "Abhi";
s1.compareTo(S2);    //return -1 because s1 < s2
s1.compareTo(S3);    //return 0 because s1 == s3
s2.compareTo(s1);    //return 1 because s2 > s1
```

String class function

The following methods are some of the most commonly used methods of String class.

charAt()

charAt() function returns the character located at the specified index.

```
String str = "studytonight";

System.out.println(str.charAt(2));

Output : u
```

---

equalsIgnoreCase()

equalsIgnoreCase() determines the equality of two Strings, ignoring thier case (upper or lower case doesn't matters with this fuction).

```
String str = "java";

System.out.println(str.equalsIgnoreCase("JAVA"));

Output : true
```

---

length()

length() function returns the number of characters in a String.

```
String str = "Count me";

System.out.println(str.length());

Output : 8
```

---

replace()

replace() method replaces occurances of character with a specified new character.

```
String str = "Change me";

System.out.println(str.replace('m','M'));

Output : Change Me
```

---

substring()

substring() method returns a part of the string. **substring()** method has two forms,

public String substring(int begin);

public String substring(int begin, int end);

The first argument represents the starting point of the subtring. If the substring() method is called with only one argument, the subtring returned, will contain characters from specified starting point to the end of original string.

But, if the call to substring() method has two arguments, the second argument specify the end point of substring.

```
String str = "0123456789";

System.out.println(str.substring(4));

Output : 456789

System.out.println(str.substring(4,7));

Output : 456
```

---

toLowerCase()

toLowerCase() method returns string with all uppercase characters converted to lowercase.

```
String str = "ABCDEF";

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

Output : abcdef
```

---

toString() with Concatenation

Whenever we concatenate any other primitive data type, or object of other classes with a String object, **toString()** function or **valueOf()** function is called automatically to change the other object or primitive type into string, for successful concatenation.

```
int age = 10;

String str = "He is" + age + "years old.";
```

In above case **10** will be automatically converted into string for concatenation using **valueOf()** function.

---

toUpperCase()

This method returns string with all lowercase character changed to uppercase.

```
String str = "abcdef";

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

Output : ABCDEF
```

---

trim()

This method returns a string from which any leading and trailing whitespaces has been removed.

```
String str = "   hello  ";
System.out.println(str.trim());
Output : hello
```

Java String charAt()

The **Java String class charAt()** method returns *a char value at the given index number*.

The index number starts from 0 and goes to n-1, where n is the length of the string. It returns **StringIndexOutOfBoundsException,** if the given index number is greater than or equal to this string length or a negative number.

Syntax

```
public char charAt(int index)
```

```
public class CharAtExample{
public static void main(String args[]){
String name="javatpoint";
char ch=name.charAt(4);//returns the char value at the 4th index
System.out.println(ch);
}}
```

Output:

```
t
```

Java String compareTo()

The **Java String class compareTo()** method compares the given string with the current string lexicographically. It returns a positive number, negative number, or 0.

It compares strings on the basis of the Unicode value of each character in the strings.

If the first string is lexicographically greater than the second string, it returns a positive number (difference of character value). If the first string is less than the second string lexicographically, it returns a negative number, and if the first string is lexicographically equal to the second string, it returns 0.

```
if s1 > s2, it returns positive number
if s1 < s2, it returns negative number
if s1 == s2, it returns 0
```

```
public class CompareToExample{
public static void main(String args[]){
String s1="hello";
String s2="hello";
String s3="meklo";
String s4="hemlo";
String s5="flag";
System.out.println(s1.compareTo(s2));//0 because both are equal
System.out.println(s1.compareTo(s3));//-5 because "h" is 5 times lower than "m"
System.out.println(s1.compareTo(s4));//-1 because "l" is 1 times lower than "m"
System.out.println(s1.compareTo(s5));//2 because "h" is 2 times greater than "f"
}}
```

Output:

```
0
-5
-1
2
```

Java String contains()

The **Java String class contains()** method searches the sequence of characters in this string. It returns *true* if the sequence of char values is found in this string otherwise returns *false*.

```
class ContainsExample{
public static void main(String args[]){
String name="what do you know about me";
System.out.println(name.contains("do you know"));
System.out.println(name.contains("about"));
System.out.println(name.contains("hello"));
}}
```

Output:

```
1.  true
2.  true
3.  false
```

Java String endsWith()

The **Java String class endsWith()** method checks if this string ends with a given suffix. It returns true if this string ends with the given suffix; else returns false.

```
public class EndsWithExample{
public static void main(String args[]){
String s1="java by javatpoint";
System.out.println(s1.endsWith("t"));
System.out.println(s1.endsWith("point"));
}}
```

Output:

```
true
true
```

Java String equals()

The **Java String class 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.

The String equals() method overrides the equals() method of the Object class.

```
public class EqualsExample{
public static void main(String args[]){
String s1="javatpoint";
String s2="javatpoint";
String s3="JAVATPOINT";
String s4="python";
System.out.println(s1.equals(s2));//true because content and case is same
System.out.println(s1.equals(s3));//false because case is not same
System.out.println(s1.sequals(s4));//false because content is not same
}}
```

Output:

```
true
false
false
```

```
public class EqualsExample2 {
  public static void main(String[] args) {
    String s1 = "javatpoint";
    String s2 = "javatpoint";
    String s3 = "Javatpoint";
    System.out.println(s1.equals(s2)); // True because content is same
    if (s1.equals(s3)) {
      System.out.println("both strings are equal");
    }else System.out.println("both strings are unequal");
  }
}
```

Java String indexOf()

The **Java String class indexOf()** method returns the position of the first occurrence of the specified character or string in a specified string.

```
public class IndexOfExample2 {
   public static void main(String[] args) {
      String s1 = "This is indexOf method";
      // Passing Substring
      int index = s1.indexOf("method"); //Returns the index of this substring
      System.out.println("index of substring "+index);
   }

}
```

Output:

```
index of substring 16
```

Java String length()

The **Java String class length()** method finds the length of a string. The length of the Java string is the same as the Unicode code units of the string.

```
public class LengthExample{
public static void main(String args[]){
String s1="javatpoint";
String s2="python";
System.out.println("string length is: "+s1.length());//10 is the length of javatpoint string
System.out.println("string length is: "+s2.length());//6 is the length of python string
}}
```

Output:

```
string length is: 10
string length is: 6
```

Java String replace()

The **Java String class replace()** method returns a string replacing all the old char or CharSequence to new char or CharSequence.

```
public class ReplaceExample1{
public static void main(String args[]){
String s1="javatpoint is a very good website";
String replaceString=s1.replace('a','e');//replaces all occurrences of 'a' to 'e'
System.out.println(replaceString);
}}
```

Output:

```
jevetpoint is e very good website
```

Java String startsWith()

The **Java String class startsWith()** method checks if this string starts with the given prefix. It returns true if this string starts with the given prefix; else returns false.

```
public class StartsWithExample
{
// main method
public static void main(String args[])
{
// input string
String s1="java string split method by javatpoint";
System.out.println(s1.startsWith("ja"));  // true
System.out.println(s1.startsWith("java string"));   // true
System.out.println(s1.startsWith("Java string"));  // false as 'j' and 'J' are different
}
}
```

Output:

```
true
true
```

false