

# String

1. String is a series of characters.
2. String is a class and not a primitive data type.
3. In java Strings are implemented using two class and java.lang package contain two string classess
  - a. String
  - b. String Buffer
4. String class is used when string cannot change.
5. String Buffer is used when there is a need of manipulation on the contents of string.

# String Functions

# Length()

Method :- length( )

Return Type :- int

Argument :- Nil

Description :- Returns the number of character of the calling string which includes white spaces.

# toLowerCase( )

Method :- toLowerCase( )

Return Type :- String

Argument :- Nil

Description :- The calling Strings gets converted in to lower case.

# toUpperCase()

**Method :- toUpperCase( )**

**Return Type :- String**

**Argument :- Nil**

**Description :- The calling Strings gets converted in to Upper case.**

# trim()

**Method :- trim ( )**

**Return Type :- String**

**Argument :- Nil**

**Description :- Remove extra white space before and after the string. But does not remove white space within the string.**

# charAt()

**Method :- charAt( )**

**Return Type :- char**

**Argument :- int**

**Description :-** Returns the character at the specified index. Where index number starts from 0 .

# equals( )

**Method :- equals ( )**

**Return Type :- boolean**

**Argument :- String**

**Description :-** when the calling string and the string within argument are same, then the function returns true else it returns false.



# equalsIgnoreCase()

Method :-equalsIgnoreCase()

Return Type :- boolean

Argument :- String

Description :- When the calling string and the string within argument are same ignoring the case whether it is capital or small case, function returns true else returns false.

# compareTo()

**Method :- compareTo( )**

**Return Type :- int    Argument :- String**

**Description :-**

- a) The calling string is less than string in the argument then the function returns any negative value.
- b) The calling string is greater than the string in the argument then the function returns any positive value.
- c) The calling string is equal to the string in the argument then the function returns zero.

# Replace( )

**Method :- replace( )**

**Return Type :- String**

**Argument :- char, char**

**Description :- Returns String after replacing the character in the first parameter by the character in second parameter.**

# startsWith()

**Method :- startsWith ( )**

**Return Type :- boolean**

**Argument :- String**

**Description :-** It checks the given String that begins with a specified string returns either 'true' or 'false'.

# endsWith()

**Method :- endsWith ( )**

**Return Type :- boolean**

**Argument :- String**

**Description :-** It checks the given String that ends with a specified string returns either 'true' or 'false'.

# indexOf ( )

This function returns the index( i.e position number) of string.

Syntax :- int variable = string variable.indexOf(character)

e.g. String s="COMPUTER";

int n = s.indexOf('P');

Here, the value returned by function is 3. Hence , n=3.

String s = "MALAYALAM";

int n= s.indexOf(4,'A');

This function will return the index of 'A' available in the string after 4<sup>th</sup> index.

Hence , n=5.

String s="COMPUTER";

int n= s.indexOf(3,'C');

It returns NULL as character 'C' is not available after 3<sup>rd</sup> index.

# substring()

This function is used to extract a part of string (i.e., simultaneous character from one index to another.)

Syntax:

String variable 1 = String  
variable.substring(index);

e.g. String s= "COMPUTER";

String p=s.substring(3);

The function will return all the characters from the string starting from 3<sup>rd</sup> index.

Hence, p=PUTER

As the return value of the function is string so, you must use a string variable which contains the result.

```
String p = s.substring(3,6);
```

It returns a part of string from 3<sup>rd</sup> position to 6<sup>th</sup> position by excluding the character, which is available at 6<sup>th</sup> position . Hence , p result PUT.



# valueOf( )

**Method :- valueOf( )**

**Return Type :- String**

**Argument :- int or float**

**Description :- Returns String after converting the numeric values which are passed as argument to the String .**

```
import java.io.*;
import java.util.*;
class S10
{
    public static void main(String args[])
    {
        Scanner obj = new Scanner(System.in);
        String str;
        int x=90;
        str=String.valueOf(x);
        System.out.println(str);
    } }
```

# concat()

This function is applied to concatenate (join) two strings together.

**Syntax:-**

String variable=string variable 1.concat(string variable2)

e.g. String x="COMPUTER "

String y=" APPLICATIONS";

String z = x.concat(y);

Here , both the strings x and y will be join together .

Hence , z results in "COMPUTER APPLICATIONS"

# lastIndexOf( )

This function is applied to find the index of last occurrence of a character in a String.

Syntax:

```
int variable =
```

```
String variable.lastIndexOf(character);
```

```
int n=s.lastIndexOf('A');
```

It returns 7 to the integer variable n.

Hence , n=7.

# Character Functions

# Character.isLetter( )

This function is used to check whether a given argument is an alphabet or not. It returns a boolean type value either true or false.

Syntax :

`boolean variable = Character.isLetter(character);`

e.g `boolean p= Character.isLetter('c');`

The function will return true to the variable p.

`boolean p=Character.isLetter('6');`

It will return false.

# Character.isDigit( )

This function returns a boolean type value true if a given argument is a digit otherwise false.

Syntax:

`boolean variable = Character.isDigit(character)`

e.g.,

`boolean p = Character.isDigit('7');`

It returns true to the variable p.

`Boolean p = Character.isDigit('G');`

It returns false to the variable p.

# Character.isLetterOrDigit( )

This function returns true if the given argument is either a letter or a digit, false otherwise.

**Syntax:-**

**boolean variable = Character.isLetterOrDigit(character)**

**e.g., boolean b= Character.isLetterOrDigit('A');**

**boolean b= Character.isLetterOrDigit('r');**

It returns true to the boolean variable b as 'A' or 'r' is a letter.

**boolean b= Character.isLetterOrDigit('9');**

It returns true to the boolean variable b as '9' is a digit.

**boolean b= Character.isLetterOrDigit('\*');**

It returns false to the boolean variable b as '\*' is neither a letter or a digit.



# Character.isWhiteSpace( )

This function can be used to check for existing blank or gap in a String. It will return a boolean type value (i.e true) if the given argument is a white space (blank) and false otherwise.

Syntax :

```
boolean variable =  
Character.isWhiteSpace(character);  
boolean b = Character.isWhiteSpace(' ');
```

It return true to the boolean variable b as the given character is a blank.

```
Boolean b = Character.iwWhiteSpace('*');
```

It returns false to the boolean variable b as the given character is not a blank.

# Character.isUpperCase( )

This function will return true if the given argument is an upper case letter other wise false.

Syntax :

boolean variable =

Character.isUpperCase(character);

boolean p = Character.isUpperCase('A');

It return true to the variable p.

boolean p = Character.isUpperCase('a');

It return false to the variable p.

# Character.isLowerCase()

This function will return true if the given argument is a lower case letter other wise false.

Syntax :

boolean variable =

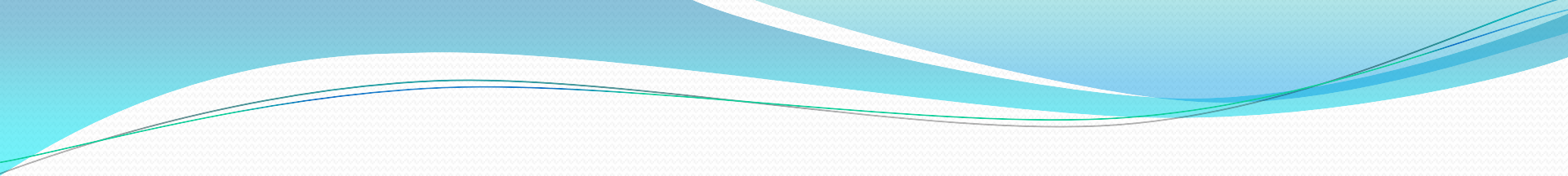
Character.isLowerCase(character);

boolean p = Character.isLowerCase('A');

It return false to the variable p.

boolean p = Character.isLowerCase('a');

It return true to the variable p.



# String Buffer Functions

# String Buffer Functions

You can declare an object of String Buffer type. This create reasonable space in the memory called Buffer to contain String. The following statements are used to create String Buffer objects:

```
StringBuffer m = new StringBuffer( );
```

This allocates an empty space in the memory to store a String.

```
StringBuffer m = new StringBuffer("COMPUTER" );
```

The statement shown above allocates memory for variable m and initializes it with String "COMPUTER".

```
StringBuffer m = new StringBuffer(25 );
```

The statement shown above allocates memory for a string upto 25 characters.

Once an object is created `StringBuffer` type, it can possess the following characteristics:

1. It refers specific length in memory to contain characters.
2. The length can be changed according to the need of the programmers. You can do so by using `SetLength( )` function.
3. If String has bigger length than the size of String Buffer object , then the characters exceeding the size are truncated.
4. If a String is smaller than the size of the String Buffer object then it contains extra characters in the empty space.

# Append()

This function is used to add a string at the end of another string.

Syntax :

```
StringBuffer variable1.append(StringBuffer  
variable2);
```

e.g.,

```
StringBuffer k = new  
StringBuffer("COMPUTER");  
StringBuffer p = new  
StringBuffer("APPLICATIONS");  
k.append(p);
```

It returns as **COMPUTERAPPLICATIONS**

# setCharAt( )

This function can be applied to replace a character with another character at specified index.

Syntax :

```
StringBuffer variable.setCharAt(index,  
character);
```

```
StringBuffer s = new  
StringBuffer("COMMUTER");  
s.setCharAt(3,'P');
```

It will returns as COMPUTER.



# Insert( )

This function allows you to insert a string at specified index into another string.

Syntax :

```
StringBuffer variable1.insert(index,  
StringBuffer variable2);
```

e.g.,

```
StringBuffer s = new  
StringBuffer("COMPUTER FUN");  
StringBuffer p = new StringBuffer("IS");  
s.insert(8,p);
```

# delete()

This function is applied to delete the characters from one index to another index in a given string.

Syntax :

StringBuffer variable.delete(index1 , index2);

e.g., StringBuffer s = new  
StringBuffer("COMPUTER");  
s.delete(3,5);

It will return as COMER.

# setLength()

This function allows you to set length of a string buffer variable upto specified range. If a string to be stored has more than specified length then the characters exceeding the length are truncated otherwise store “\u0000” characters in the empty space.

**Syntax :** StringBuffer  
variable.setLength(Number of characters);  
e.g., p.setLength(20);  
Now, the variable p is set to accommodate a string upto the length 20 characters.

# Reverse()

This function is used to reverse the characters of a given string.

Syntax :

```
StringBuffer variable.reverse( );
```

```
e.g., StringBuffer p = new  
StringBuffer("COMPUTER");  
p.reverse( );
```

It return s as reversed string RETUPMOC.

# Difference

## String

1. String type object has fixed length.
2. New object is created to produce the change.
3. It is a general approach of programming.

## StringBuffer

1. StringBuffer type object has provisions to change the length.
2. Change is maintained in the same object.
3. It is an advanced approach of programming.