STRING

Strings play a very important role in Java programming. Java provides a String class to manipulate and perform operations on strings.

1)
```
String name = "Priya"
String attribute = "Cute".
System.out.println(name);
System.out.println(attribute);
System.out.println(name +attribute);
System.out.println(name + "is a " + attrib
```

Output →
```
Priya
Cute
Priya is a Cute.
```

① int length()

Syntax : stringName.length( );

```
System.out.println(attribute.length());
```

Output
```
4
```

② char ~~at~~ charAt(int index)

Syntax: stringName.charAt(index);

```
System.out.println(name.charAt (~~4~~2)); // a
System.out.println (name.charAt (5)); //erro
```

name = "Priya"
0 1 2 3 4

③ String concat(String string1)

Syntax : string1.concat(string2);

ⓒ String surname = Singh

```
// name = name.concat(surname);              name += surname;
name = name + surname; /name
System.out.println(name);
```

Output

```
Priya Singh
```

④ String substring(int beginIndex)

Syntax : string.substring(beginIndex)

```
                         0 1 2 3 4 5 6 7 8 9 10 11
String statement = "Welcome to Java Tutorials by
                         Yash Jain";

String substring1 = statement.substring(11);
System.out.println("The substring is:"
```

⑤ String substring(int beginIndex, int endIndex)

syntax : string.substring(beginIndex, endIndex)

```
+ subString1);
String subString2 = statement.substring(11,14);
System.out.println("The New Substring is:"
                         + subString2);
```

Output

```
The Substring is : Java Tutorials by Yash
                                     Jain.

The New Substring is : Jav
```

J	a	v	/0
11	12	13	14

⑤ int compareTo(String string1, String strin

Syntax: string1.compareTo(string2);

When we used the compareTo() method, the method returns an integer which indicates the lexicographic (alphabetical) comparison of two strings.

The result is negative if the relative alphabet value of the particular letter of the first string is smaller than that of the second string's letter on the same location. And it is positive if the first string is lexicographical larger than the second string.

If both strings are identical, then a value zero(0) is returned.

```
(*) String string1 = "Hello Yash sir ke champs";
String string2 = "Hello Yash sir ke champs";
int result = string1.compareTo(string2)
System.out.println(result); //0
String sentence = "Welcome";
result = sentence.compareTo("to knowledge gate
                              tutorial");
System.out.println(result); //-29
String sentence1 = "This is a";
String sentence2 = "Java tutorial";
result = sentence1.compareTo(sentence2);
System.out.println(result); // 10
```

Note
1st & 2nd string ke har character ko compare karne k baad dono me jo jiska ASCII value jada hoga. Agar 1st ka hai To '+ve' & 2nd ka hai To '-ve'.

(7) String toUpperCase()

Syntax: string.toUpperCase()

String temp = " Anjali";

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

Output

ANJALI

(8) String toLowerCase()

Synatax: string.toLowerCase()

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

Output

anjali.

(9) String trim()

Syntax: string.trim()

String temp2 = " Anjali loves Abhijeet ";

System.out.println(temp2);

System.out.println (temp2.trim());

~~System.out.println (temp2.replace('a','z'));~~

Output

Anjali loves Abhijeet

Anjali loves Abijeet.

(trim() का use only ~~to~~ starting and ending space remove ~~करने~~ के लिए किया जाता है।)

(10) String replace (char oldChar, char newChar)

Syntax: string.replace(char1, char2)

Syntax.out.println (temp2.replace('a','J'));

~~to~~ ~~Jan~~ JnJjdi loves Jbhijeet.

```
int p=5, q=10;
System.out.println(Math.max(p,q));
System.out.println(Math.max(5,10));
System.out.println(Math.min(p,q));
System.out.println(Math.sqrt(49));
System.out.println(Math.~~abs~~cos sqrt(49));
System.out.println(Math.abs(-5));
System.out.println(Math.abs(5));
System.out.println(Math.pow(3,2));
System.out.println(Math.log(2));
System.out.println(Math.log10(100));
System.out.println(Math.log10(2));
```

<u>Output</u>

```
10
10
5
7.0
6.855654600401044
5
5
9.0
0.6931471805599453
2.0
0.30102999566398120
```

Note
Maximum, minimum and abstraction are always integer value return. ~~rest of them~~
~~All are~~ ~~Double~~ Rest of them are double.

double a=90;
// converting values to radian
double b = Math.toRadians(a);
System.out.println(b);
toRadian(90°)

$180° — \pi^c$

$90° — \frac{\pi}{2}^c$

$\pi — 3.14$

$\frac{\pi}{2} = \frac{3.14}{2} = 1.5707963267948966$

System.out.println(Math.ceil(2.45)); 2+1=3
System.out.println(Math.ceil(2.73)); 2+1=3
System.out.println(Math.ceil(2.00001)); 2+1=3
System.out.println(Math.ceil(2)); 2
System.out.println(Math.ceil(2.99)); 2+1=3

System.out.println(Math.floor(2.45)); 2
System.out.println(Math.floor(2.73)); 2
System.out.println(Math.floor(2.00001)); 2
System.out.println(Math.floor(3)); 3
System.out.println(Math.floor(2.99)); 2

~~2~~

2.45
Cil / floor
2+1=3 / 2

Math.random() → It returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0

Math.rint() → It returns the double value that is closest to the given argument and equal to mathematical integer.

Math.hypot()
It returns sqrt(x^2+y^2) without intermediate overflow or underflow.

Math.ulp()
It returns the size of an ulp of the argument

Math.getExponent()
It is used to return the unbiased exponent used in the representation of a value.

Math.IEEEremainder()
It is used to calculate the remainder operation on two arguments as prescribed by the IEEE 754 standard and returns value.

Math.addExact()

It is used to return the sum of its arguments, throwing an exception if the result overflows an int or long.

How to generate an OTP via JAVA Programming?

```
System.out.println(Math.random());
System.out.println(3+(9-3)* Math.
                          random());
System.out.println(3+(9-3)* Math.
                          random());
:
```

~~error~~
Output

```
0.2456789862 ~
3.6.7.3.8.5
```