operator :- Logical or (||) :-

eg:- int x = 5;
System.out.println(x>3 || x<4);

// returns true because one of the conditions satisfied.

operator :- logical not (!) :-

eg:- int x = 5;
System.out.println(!(x>3 && x<10));

// returns false as it reverse the result.

Java Strings

Strings are used for storing text.

A String variable contains a collection of characters surrounded by double quotes.

eg:- public class Main {
public static void main (String[] args) {
String greeting = "Hello";
System.out.println(greeting);
}
}

eg:-
```
Public class Main {
    Public static void main (String [] args)
    {
        String txt = "ABCDEFGHIJKLMNOPQRST
                      UVWXYZ";
        System.out.println ("The length of the text
        String is: " + txt.length());
    }
}
                    //output is 26
```

String Methods :-

```
Public class Main {
    public static void main (String [] args)
    {
        String txt = "Hello World";
        System.out.println (txt.toUpperCase());
        System.out.println (txt.toLowerCase());
    }
}
                    // HELLO WORLD
                    // hello world
```

eg:

```
String txt = "Please locate where 'locate'
              occurs!";
System.out.println (txt.indexOf ("locate"));
}
```

String Concatenation

The + operator can be used between strings to combine them. This is called concatenation.

NOTE: we have added an empty text (" ") to create a space between firstName and last name on print in below code:-

eg:-
```
String firstName = "Gauri";
String lastName = "Doe";
System.out.println (firstName + " " +
                    lastName);
```

Concat() method:-

```
String firstName = "John";
String LastName = "Harry";
System.out.println (firstName.concat (LastName));
```

NOTE: Java uses the + operator for both addition and concatenation.

→ Numbers are added.
→ Strings are concatenated.

eg:- int x = 10;
int y = 50;
int z = x+y;

output:- 30

whereas String x = "10";
String y = "20";
String z = x+y;

output: z => 1020

Point to remember:-
If you add a number and a string, the result will be a string concatenation

eg:- String x = "10";
int y = 20;
String z = x+y;

output:- 1020 (a String)

Strings Special Characters

Strings must be written within quotes, Jana will misunderstand this string, and generate an error:

```
String txt = "We are the so-called "Vikings" from the north.";
```

The solution to avoid this problem, is to use the backslash escape character.

The backslash (\) escape character turns special characters into string characters.

Escape characters	Result	Description
\'	'	Single quote
\"	"	double quote
\\	\	Backslash

eg:- String txt = "We are the so-called \"Vikings\" from the North";

eg: String txt = "It\'s alright";

eg: String txt = "The character \\ is called backslash.";

Other common escape sequences

	Code	Result
1)	\n	New line
2)	\r	Carriage Return
3)	\t	Tab
4)	\b	Backspace
5)	\f	form feed

Java MATH

The Java Math class has many methods that allows you to perform mathematical tasks on numbers.

Math.max(x,y)

eg:-
```
Public class Main {
    Public static void main (String[] args)
    {
      System.out.println (Math.max(5,10));
    }
}
```

output:- 10

Math.min(x,y)

eg:- System.out.println (Math.min(5,10));

output:- 5

Math.sqrt(x)

eg:- System.out.println (Math.sqrt(64));

output:- 8.

Math.abs(x)

The Math.abs(x) method returns the absolute (positive) value of x:

Eg:- Math.abs(-4.7);

output: -4.7 ⟶ 4.7 (true)

Random Numbers

Math.random() returns a random number between 0.0 (inclusive), and 1.0 (exclusive):

eg: Math.random();

eg: int randomNum = (int)(Math.random() * 101);

Java Booleans

Very often, in programming, you will need a data type that can only have one of two values, like:

- ↳ Yes / NO
- ↳ ON / Off
- ↳ true / false

For this, Java has a boolean data type, which can store true or false.

eg:- boolean isJavaFun = true;
System.out.println(isJavaFun);

Boolean Expression is a Java expression that returns a Boolean value: true or false.

This is useful when we want to compare values to find answers.

for eg:-

```
int x = 10;
int y = 9;
System.out.println(x>y);
```

output:- returns true

Note:- The Boolean value of an expression is the basis for all Java comparisons and conditions.