JAVA print Variables:

If we have print the variable along with text + operator is used.

Eg:

```
Public class Main {
    public static void main (string[] args) {
    System out println ("Hello" +
    String name = "Jaya";
    System.out.println ("Hello" + name);
    }
}
```

o/p:- Hello Jaya

In case of number + will perform as mathematical operator.

```
int x = 5;
Int y = 6;
System.out.println (x + y); // print the values of x+y
```

o/p:- 11

Eg-3

```
string fst name = "Jaya";
String sec name = "Sri";
string full name = fst name + sec name;
System.out.println (full name);
```

o/p:- Jaya sri

Declare multiple variables

* By using comma separated list different variables can be assigned for a same data type.

Eg:-

```
int x = 10 , y = 3 , z = 50 ;
System.out.println (x+y+z) ;
```

o/p: 63

* more than 1 variable can take same value

Eg:

```
int x, y, z ;
x = y = z = 50;
System.out.println (x+y+z) ;
```

o/p:- 150

Identifiers:

Variables are identified by unique name called identifiers

Set of rules:

* name should begin with lowercase letter and should not contain whitespace.

* names are case sensitive (myVar or myvar are different)

* Reserved words (JAVA keywords like int boolean) should not be used as name.

* name can contain digits, underscore, dollar symbol.

* It can also begin with dollar symbol.

* It must begin with letters

eg:

```
int m = 60 ;
```

(or)

```
int minutesperhour = 60;
```

Data types:

Data types / Primitive
\ Non-primitive

Primitive Data types

* Byte - 1 byte
* short - 2 byte
* int - 4 byte
* long - 8 byte
* float - 4 byte
* double - 8 byte
* boolean - 1 byte
* char - 2 byte

These data types specifies the size and type of variable values

JAVA numbers

Primitive Data types / Integer
\ floating

Integer → whole numbers, +ve (or) -ve numbers

Data type used: int, byte, long, short

byte:

stores whole numbers from -128 to 127

```
byte myNum = 100 ;
System.out.println (myNum);
```

Short:

Stores value from -32768 to 32767

Long:

Stores value from -9223372036854775808 to 9223307203685477580 7.

Int:

Int stores value from -2147483648 to 2147483647.

Floating Point types:

↳ For storing decimal values and fractional parts.

↳ Data types used: float & double.

float → precision 6 digits after decimal point.

double → precision 15 digits after decimal point.

(X) If we use float then the value should end with 'f' ~~!!!~~ for double, it ends with 'd'

Eg:

(1) float myNum = 9.99f;

System.out.println(myNum);

(2) ~~float~~ double myNum = 99.999d;

System.out.println(myNum);

Scientific number:-

floating point no can also be scientific number with e to indicate the power of 10.

Eg:

```
float f1 = 35e3f;
double d1 = 12E4d;
System.out.println(f1);
System.out.println(d1);
```

JAVA Boolean Data Types:

For true / false

Yes / No

ON / OFF

Conditional statements

```
boolean isKISHORE_PRESENT = true;
System.out.println("KISHORE_PRESENT");
```

O/P: true

Characters:

Store single characters but in single quotes

eg:

```
char myGrade = 'B';
System.out.println(myGrade);
```

```
char myVar1 = 65, myVar2 = 66, myVar3 = 67;
System.out.println(myVar1);
System.out.println(myVar2);
System.out.println(myVar3);
```

Strings:

Store sequence of characters in double quotes

eg:

```
String greeting = "Hello world";
System.out.println(greeting);
```

Non-Primitive Data types:

These reference type because they refer to the Objects

Difference between primitive & Non-primitive

Primitive data type	Non-primitive data
Predefined by JAVA	Created by user
Cannot call the operations to perform	Can call to perform certain operation
Has a value	Value is null
Lowercase start	Starts with uppercase eg: Strings, Arrays, Class, Interface

JAVA Type CASTING:

Assign a value of primitive data type to another

Two types:

* Widening casting

converts smaller type to larger type of data

byte > short > char > int > long > float > double

```
eg. public class Main {
        public static void main (String[] args) {
            int MyInt = 9;
            double myDouble = myInt;
            System.out.println(myInt);
            System.out.println(myDouble);
        }
    }
```

* Narrowing

converting larger type to smaller type

double > float > long > int > char > short > byte

It is done by placing the type of data in parenthesis before the value

```
public class Main {
    public static void main (String[] args) {
        int myint = 9;
        double mydouble = (int) myint;
        double myDouble = 9.78d;
        int myint = (int) myDouble;
        System.out.println (myDouble);
        System.out.println (myInt);
    }
}
```

Java operators:

It is used to perform operations on variables and numbers.

Types:

* Arithmetic operator
* logical operator
* Assignment operators
* Comparison operators
* Bitwise operators

Arithmetic operation perform addition, subtraction, increment, decrement, multiplication, division.

```
Public class Main {
    public static void main (String[] args) {
        int x = 5;
        int y = 4;
        System.out.println (x+y);
    }
}
```

Decrement --x ; Increment ++x addition (+)
Subtraction (-) multiplication (x) division /

Assignment Operator :-

Assign Values for the Variables

```
public class Main {
    public static void main (String[] args);
        int x = 5;
        x += 5;
        System.out.println (x);
    }
}
```

100
2|5
2|2-1
1-0

Comparison operator :-

compare the two values.

101
011
001

Value true or False.

```
int x = 5;
int y = 3;
System.out.println (x>y);
```

O/P :- TRUE

operators : >, <, !=, ==, <=, >=

Logical operators :-

&& - Logical and
|| - Logical OR
! - Not

eg:

```
int x = 5
System.out.println (x > 10 && x < 3);
```

o/p:

False

Assignment operators:

symbols: `&=` → and (bitwise)

`|=` → or (bitwise)

`^x` → exclusive or.

```
int x = 5;
int y = 3;
System.out.println (x &= y);
```

o/p:- 1 [manually - 5 ⇒ 101
3 ⇒ 011 &
001]

```
System.out.println (x |= y);
```

o/p: 6

```
5 ⇒ 101
3 ⇒ 011 |
    110 → 6
```

JAVA strings:

For storing the text.

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

String length :-

It is object that performs certain operations on strings.

It can be found with length() method.

```
String txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
System.out.println ("The length of string is:" + txt.length);
```

Here the length of text is found. o/p → 26

o/p: The length of String is : 26

other string Methods:

toUpperCase()
toLowerCase()

eg:

```
String name = "Jayasri";
System.out.println("Uppercase letter = " + name.toUpperCase());
```

o/p:

Uppercase letter = JAYASRI

Finding a character in a string:

indexOf() → returns the index of the first occurence of a specified text in a string

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

o/p: 7

JAVA String Concatenation:

+ → strings concatenate

similarly the concat() → join two text

```
String firstName = "John";
String Secondname = "Rio";
System.out.println(firstname.concat(Secondname));
```

o/p: John Rio

Java numbers and strings:

Both numbers

① int x = 20;
int y = 15;
System.out.println(x+y);
o/p: 35

String (both)

② String x = "20";
String y = "15";
System.out.println(x+y);
o/p: 2015 (String)

③ 1 number, 1 string o/p → string concatenation

String x = "10";
int y = 20;
String z = x+y;
System.out.println(z);
o/p: 1020 (string)

JAVA Special characters:

↳ within quotes present - JAVA misunderstand
So we are using \", \', \\

String name = " This is "blue" colour";

↳ error

eg
String name = "This is \"blue\" colour";
System.out.println(name);
o/p :- "This is "blue" colour.

Others:

\n → new line Hello\n world → o/p: Hello
world

\r → carriage return

\t → Tab → space Hello\t world → o/p: Hello world

\b → Backspace → "Hel\blo world → o/p: Helo world

\f - Form feed

JAVA math:

1) perform mathematical operation.

① Math.max(x,y)

eg:-

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

② Math.min(x,y); ⇒ Minimum value;

③ Math.abs(x,y); ⇒ Positive value

④ Math.sqrt(x); ⇒ Square root of no.

⑤ Math.random~~number~~(); ⇒ random number bet 0.0 to 1.0 (excludes) (includes)

⑥ ⇒ For ~~void~~ specific value of random number ⇒ (0 to 100)

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

JAVA Boolean:

It is used to return "true" (or) "false"

eg: boolean JAVAisfun = "true";
boolean isfishtasty = false;
System.out.println(JAVAisfun);
System.out.println(fishtasty);

o/p: true
false

⇒ It uses boolean data type

⇒ Mostly it returns "true" (or) false"
Yes or No
ON or OFF

comparison operation is also used to return true or f

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

o/p True :

Real life example using if else:

```
Print
public class Main {
    public static void main(~~sout~~ String[] args) {
        int myage = 20;
        int voting age = 18;
        if (myage >= voting age) {
            {
            System.out.println("can vote");
            }
        else
            {
            system.out.println("cannot vote");
            }
    }
}
```

o/p :- can vote.

JAVA if else:

① if → True condition

② else → if (True) → Print. if it is false. it moves to else condt.

③ if else if → To check the other block of condt.

④ Switch → many alternative blocks of code to be executed.

Example for if else, (syntax).

```
if (condition) {
    code to be executed;
}
else {
    code to be executed;
}
```

eg.

```
if (18 < 20) {
    System.out.println("print true");
}
else {
    System.out.println("False");
}
```

o/p:
print true

else if - example.

```
if (10 < 20) {
    System.out.println("True");
}
else if (10 < 18) {
    System.out.println("False");
}
else
{
    System.out.println("true false");
}
```

Java Short Hand if else (Ternary operator):

↳ Three operands

↳ multiple lines of code with a single line.

↳ Replacement of if else statement.

Syntax:

```
Variable = (condition) ? expressionTrue : expressionFalse ;
```

eg:-

```
Int time = 20;
String time = (time > 40) ? "Good day" : "Bad day";
System.out.println (time);
```

JAVA Switch:

It is used instead of if else

↳ Selects one code block out of many.

Syntax:

```
switch (condition) {
  case 1 :
      block of code ;
      break ;
  case 2 :
      block of code ;
      break ;
  default :
      block of code ;
}
```

⇒ break & default are keywords

↳ break → It stops from the execution of further block of codes.

↳ default → if every block is not satisfied they default block is displayed.

eg:

```
int day = 4 ;
Switch (day =4) [
    Case 1 :
        System.out.println ("Monday");
        break ;
    Case 2 :
        System.out.println (" Tuesday");
        break ;
    Case 3 :
        System.out println (" Wed " );
    break ;
    Case 4 ;
        System.out.println (" Thursday");
    break ;
    default :
        System.out.println (" No one") ;
    }
```

o/p :- Thursday.

JAVA while loop:

Loop → execute a block of code as long as a specified condition is reached.

while loop :→ check the code block as long as the condn is true.

Syntax:

```
while (condition) {
        code block ;
        (Increment or decrement) ;
    }
```

eg:

```
int i = 0 ;
while (i<5){
    System.out.println(i) ;
}
```

o/p: 0
1
2
3
4

do/while loop:-

↳ check the block once and repeat till the condn. is attained

eg:

```
int i=0;
do {
    System.out.println(i);
    i++;
}
while (i>5);
```

o/p: 0
1
2
3
4

JAVA For Loop:-

If the Number of times the loop has to executed is known then for loop is used instead of while

Syntax

```
for (statement1 ; Statement 2 ; Statement 3;) {
    code block;
```

eg:

```
for (int i=0 ; i>5 ; i++) {
    System.out.println(i);
}
```

o/p: 0
1
2
3
4

Statement-1 ⇒ assignment of values

Statement-2 ⇒ condition

Statement-3 ⇒ incrementation

Nested loop :

The loop placed inside one loop is nested loop.

↳ For each outer loop inner loop is executed

```
for (int i=0 ; i<3 ; i++) {
    System.out.println ("outer" + i);
    for (int j=0 ; j<2 ; j++) {
     System.out.println ("inner" + j);
  }
}
```

o/p: outer 0
inner 0
inner 1
~~inner~~ .
outer 1
inner 0
inner 1
outer 2
inner 0
inner 1

JAVA For Each Loop:

For - Each Loop:

↳ Loop through an array.

Syntax:

```
for (type VariableName : arrayName) {
    code block
}
```

eg:

```
String [] cars = {"Volvo", "BMW", "Audi"}
for (String i : cars) {
      System.out.println (i);
}
```

o/p: Volvo
BMW
Audi

JAVA Break and continue

Break ⇒ It breaks the code after condt. is satisfied

Continue ⇒ It ~~continue~~ only break the particular condt. again it continue until the condn. is satisfied

eg:- ⇒ Break

```
for (int i = 0 ; i < 10; i++) {
    if (i == 4) {
        break ;
    }
    System.out.println (i);
}
```

o/p : 0
1
2
3

Continue eg:-

```
for (int i=0 ; i<10; i++) {
    if (i == 4) {
        continue ;
    System.out.println (i) ;
}
```

o/p: 0
1
2
3
5
6
7
8
9

Break and continue in while loop.

```
int i=0;
while (i<10)
{
    ~~if (i == 4){~~ System.out.println (i);
    ~~i++~~ i++ ;
    if (i==4) {
        break;
}
```

Continue:

```
int i = 0;
while (i < 10) {
    if (i == 4) {
        i++;
        continue;
    }
    System.out.println(i);
    i++;
}
```

JAVA array:

↳ used to stores various values in one variable name

```
String [] cars = {"volvo", "BMW"}
```

[] → It is used to determine the array.

```
String [] cars = {"volvo", "BMW", "xxx"};
System.out.println (cars);
```

o/p: volvo BMW xxx

↳ It can be accessed by indexed number:

```
String [] cars = {"volvo", "BMW", "xxx"};
System.out.println (cars [1]);
```

o/p: BMW

↳ we can change an array element

```
String [] cars = {"volvo", "BMW", "xxx"};
cars [0] = "YYY";
System.out.println (cars[0]);
```

o/p: YYY