* Outputs
* Variables
* Datatypes
* Operators
* Inputs
* Conditionals
* Switch cases
* Loops
* Arrays
* Keywords

NOTE:

* n % 10 🡪 Gives last digit
* n /= -> Deletes last digits
* **Outputs:**

Q.Why to use outputs:

To print message.

Used System.out.println/ print \n (“Messages” / or anything you wont to print).

* Println – will leave one line after output
* Print – will not leave one line agter output
* \n – similar to println.

We don’t use double quotes if we wont the output the same that is stored in variabels or methods i.e int var = 2;

Syso(var) – no need for double quotes.

As doubles quotes print the same message inside it.

* **Variables:**

Use to save data /values

1. GLOBAL VARIABLE: (Data Members)

* Declared anywhere in the class body but not inside any method or block.
* Intialize via default constructor
* Always called by its objects **reference variable using dot**
* Eg = int rollno;
* The global variable default values is assigned by jvm and if we declare int a = 10; 🡪 in this case default constructor will intialize this 10 value.

1. LOCAl VARIABLE:
   * Variable declared inside method or block.
   * We need to intiliaze every variable
   * Acces only to its method or block.
   * Local variable are called in SOP statement.
   * Only final modifier is allowed

Q:Can we write same global var and local var

Yes Because local var scope is within the ending of the brackets.

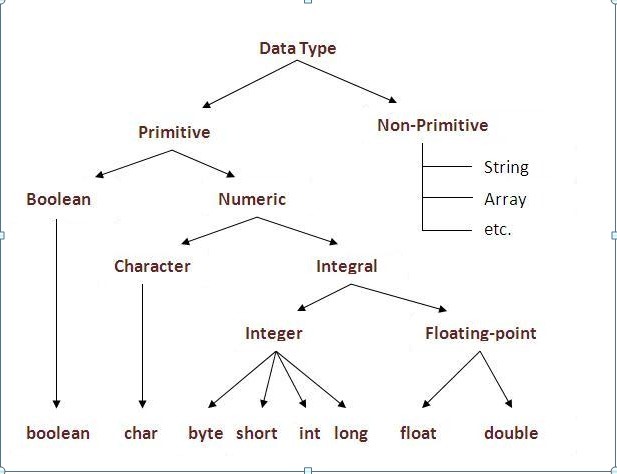
Q:Can we write same var name in local/global

No,Because it would Conflict.

* **Dataypes:**

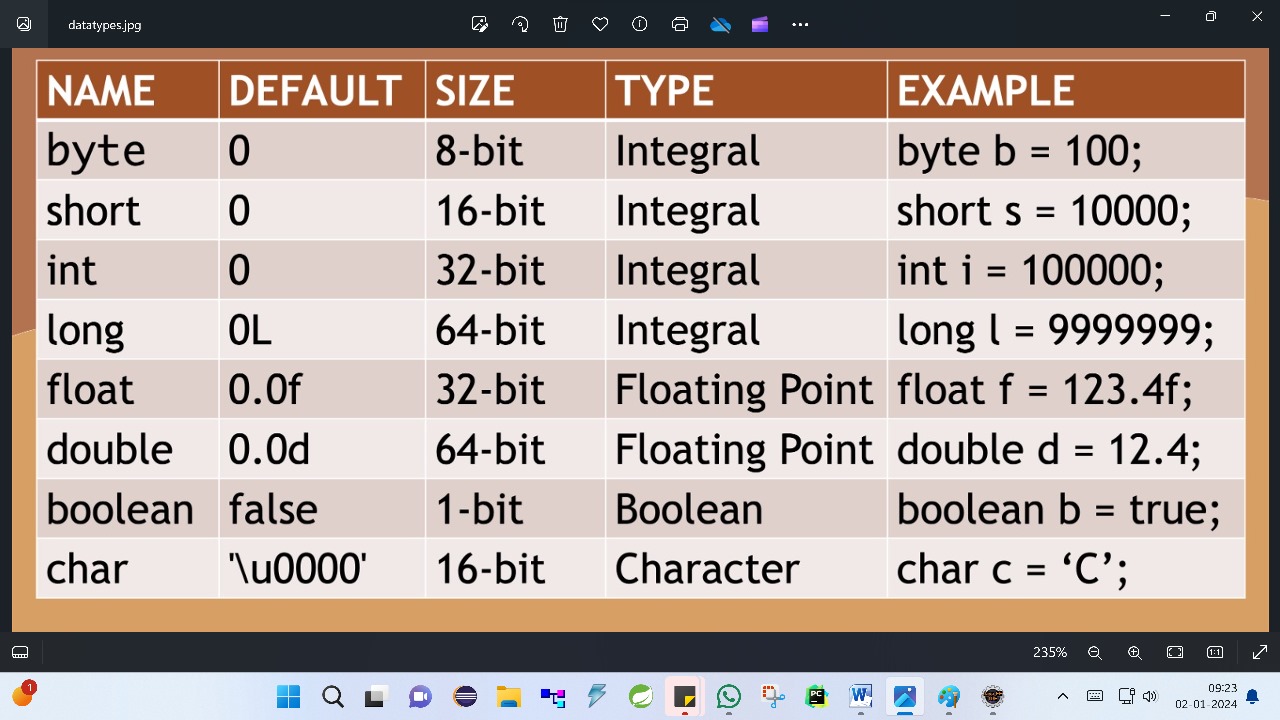
Datatypes show the type of data we store in variables.

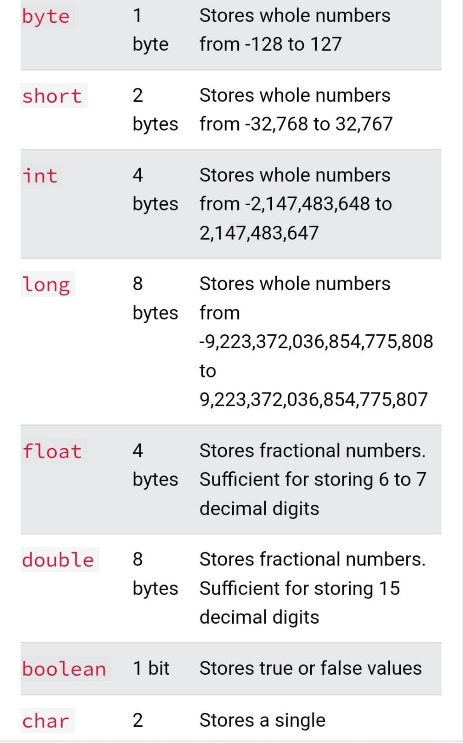
**Classification:**



User Defined

Java Defined



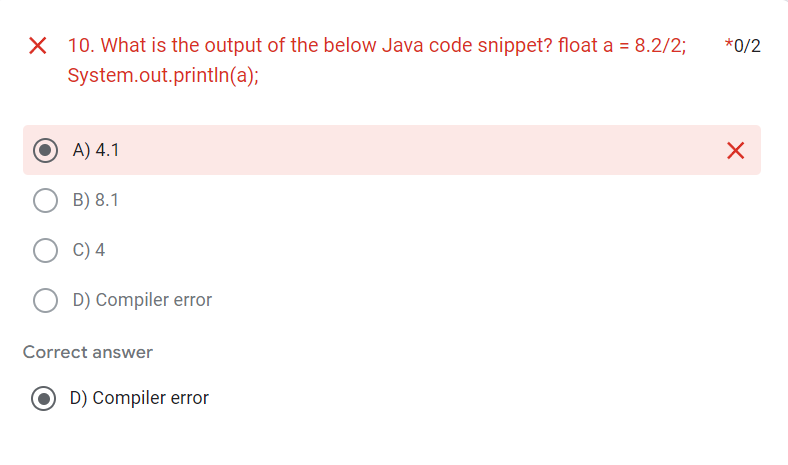


Boolean

0 -> false

1 -> true

Hence one bit is enough



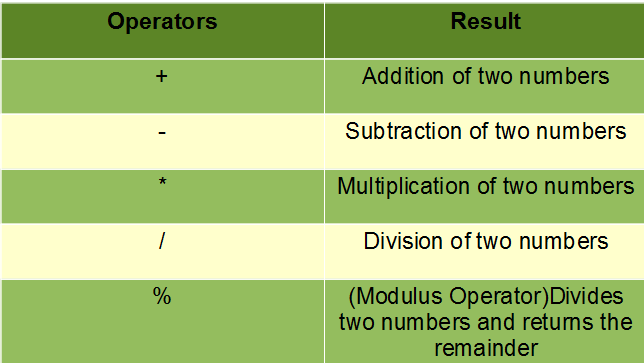
Primitives mai bhi literals or object which is called wrapper class

Eg :

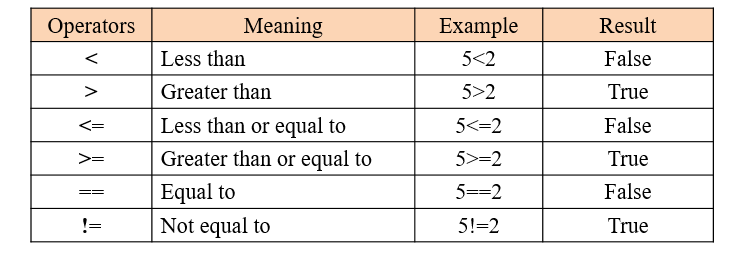
int i = 0 //literal AND Integer i = new Integer(0).

* **Operators:** Operator in [Java](https://www.javatpoint.com/java-tutorial) is a symbol that is used to perform operations. For example: +, -, \*, / etc.
* **Operands :** Operands is the value to which operation is to perform. Eg : a + b : here a,b is operand and + is operator

Arithmetic Operators : Normal mathematical operation



Comparison/Relational Operators : Compare two or more expression



Unary Operators : Done on only one value.

Two Types :

* Increment : Adding by one
* Decrement : Decreasing by one
* Pre-Increment / Pre – Decrement (++a) / (--a)

1: Change value.

2: Use Value

* Eg: int a = 5 ;
* Syso(--a)
* OUTPUT:

4

* Post-Increment / Post – Decrement (a++) / (a--)

1. Use Value
2. Change Value

Eg: int a = 5;

Syso(a++)

OUTPUT:

5

Assignment Operators : Assign Values

|  |  |  |
| --- | --- | --- |
| Operator | Operation | Example |
| = | Assigns value of right operand to left operand | A=B will put value  of B in A |
| += | Adds right operand to the left operand and  assigns the result to left operand. | A+=B means A =  A+B |
| -= | Subtracts right operand from the left operand  and assigns the result to left operand. | A-=B means A=A-B |
| \*= | Multiplies the right operand with the left operand  and assigns the result to the left operand. | A\*=B means A=A\*B |
| /= | Divides left operand with the right operand and  assigns the result to left operand. | A/=B means A=A/B |

Logical Operators : Used to connect multiple expressions or conditions  together.

|  |  |  |
| --- | --- | --- |
| Operator | Operation | Example |
| && | AND operator. Gives true if both operands are non zero . dono condition true hona chaiye | (A && B) is  false |
| || | OR operator. Gives true if atleast one of the two  operands are non-zero.  Dono mai se ek condition true to true | (A || B) is  true |
| ! | NOT operator. Reverse the logical state of operand  true rehga toh false  False hoga toh true | !A is true |

* **Inputs:** We take input Via **Scanner Class** which is present in **util Package .**

Import java.util.Scanner 🡪 To use Scanner class

Scanner sc = new Scanner(System.in). System.out

Taking input from console

Creating Object:

Why: Because every class to access its properties/methodsneeds objects

Showing output in console

Taking inputs can use nextDatatypes() (The datatypes first later Should be captial)

Eg: nextInt(),next()🡪 for string , **int i = sc.nextInt() 🡪 *store output in the same value as input***

next().charAt(0) -> for char . here next gives the string but uska first Alphabet charAt deta hai

eg: Enter 🡪 enter word ke first index i.e 0 mai ‘’E’ 🡪charAt return E

* **Conditionals: (Example in Conditonals packages)**

1. If: Will execute the condition is true (boolean value false huva toh bhi execute nhi hoga).
2. else: Will execute the condition is false

if -> For 1 output

if – else -> For 2 output.

else – if -> For 3 output (else if increase performance )

Q:Why to use else-if rather than nested if.

Nested if will check all the condition will return true if all the condtion is true.

Hence to skip other condition we use else-if ladder.

It will check the condition and will return true and then end the program.

Q:Can we write else first.

No,has there are no condtion includes in else .

Also only false value are executed

Q:When to use {}

Write bracket if there are more than one condition.

No need to write bracket if there is only one statement in the block.

* **switch cases:**
* No complex variables
* More efficient than else-if
* Cant write condition

Eg: Value between 50 to 60 greter than yeb sab nhi hota direct value hi aaiga.

* Only support int,byte,short,char,String(String is supported after version 1.8 / java 8)

switch(key) : key is nothing but variable

break:

* Reserved Keyword
* Use if you only wont 1 single output
* If one condition become true than all other cases would not execute and program will end

Q . When to not use break keyword

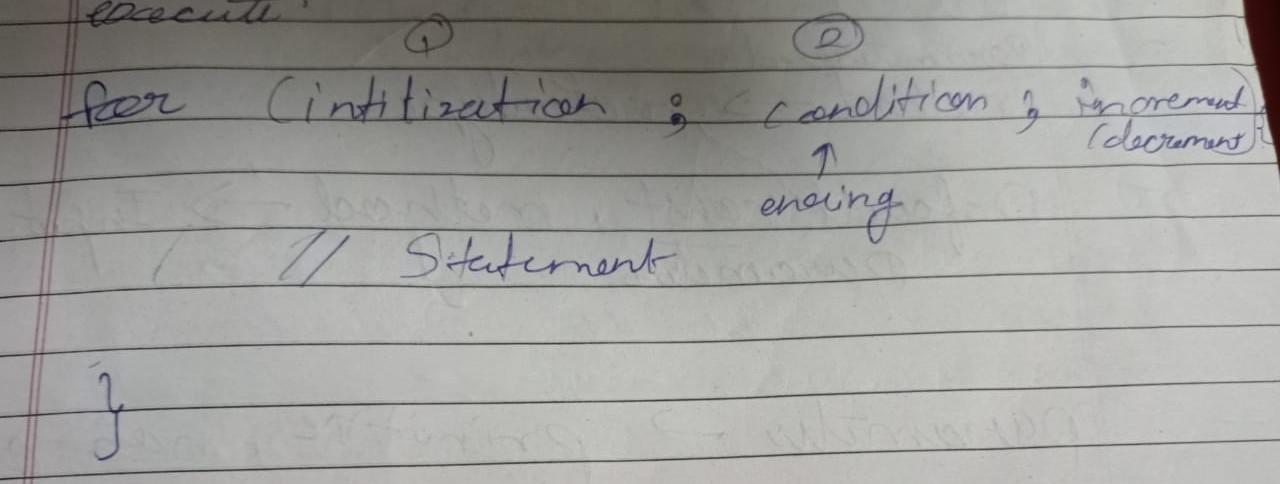
Eg: Print 5 and above days toh 1 to 4 mai break use karo and 5 to 7 ke beech mai mat karna.

Default:

* Same as else
* Do not require break if return at the end
* But agar beech mai hum default likha (Which is not good practice) toh break likhna imp hai.
* **Loops**

To do something repeatedly

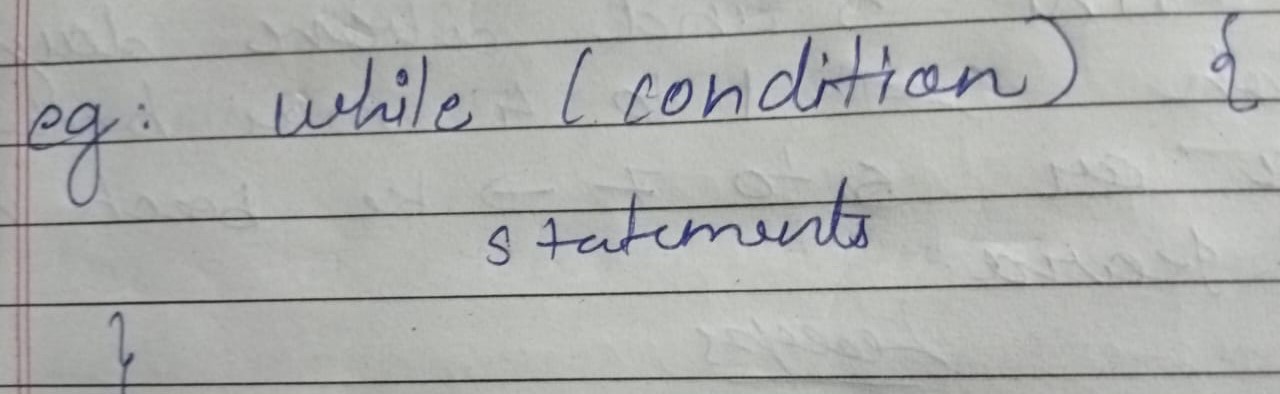
1. for loop 🡪 use when we know how much time the loop will execute



1. while loop 🡪

use when we don’t know how much time the loop will execute

while mai increment /decrement andar hota hai



3 for-each loop

Java provides an enhanced for loop to traverse the data structures like array or collection.

In the for-each loop, we don't need to update the loop variable.

Introduced in jdk 1.7.

The syntax to use the for-each loop in java is given below.

for(data\_type var : array\_name/collection\_name){

//statements

}

Step -1 : for( :arrayname)

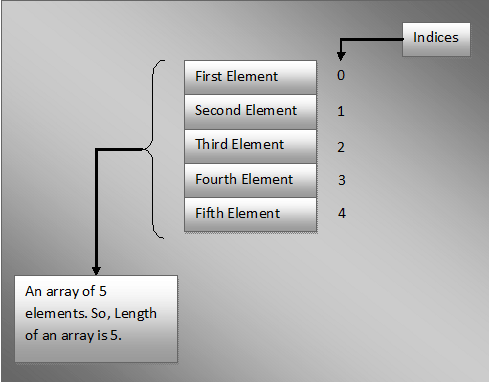
Step -2 for (type of array variable : arrayname) {

//ref var

}

* **Arrays**
* Array is a set of values where each value(Elements) is identified by an index.
* You can make an array of int’s, double’s, boolean’s or any other types but all the values of array must be of same type.
* The index of an array starts from 0.
* The following diagram shows how the array elements are stored in an array object.
* Array follows similar(same) data (homogenous)

System.***out***.println(arrayOfint.length); //this will give the elements present in the array.



5 here is array size/length

Length-> is an variable predefined by java to get the size

Declaring Arrays in Java:

**Data\_Type[] Variable\_Name;**

**AND**

**Data\_Type Variable\_Name[];**

public class ArraysInJava

{

    public static void op(String[] args)

    {

        int[] arrayOfInts;    //Declaring an array of ints

        double arrayOfDoubles[];   //Declaring an array of doubles

        char[] arrayOfChars;     //Declaring an array of characters

        boolean arrayOfBooleans[];   //Declaring an array of booleans

    }

}

Note : As both styles of declaring arrays in java are valid but the style **Data\_Type[] Variable\_Name** is preferred. The style **Data\_Type Variable\_Name[]** comes from C/C++ and it is included in java to accommodate C/C++ programmers.

**Instantiating an Array Object:**

int[] arrayOfInts = new int[10];

arrayOfInts[2]=12;

arrayOfInts[5]=56;

2nd way :

Double[] arrayofDouble = new double[]{12.56,45.84,14.85};

int[] arrayOfints = {12,21,0,5,7};//this is also ok

**Accessing Array Elements:**

int[] arrayOfints = {12,21,0,5,7};

Sop(arrayOfints[0]);

Sop(arrayOfints[3]);

Size/length – starts form 1

Index starts from - 0;

* **Keywords**

1. **assert:**

In Java, the assert keyword is used for debugging purposes to test assumptions in your code.

It enables you to declare a condition (or assertion) that you believe should always be true.

If the condition is false, the program throws an AssertionError. By default, assertions are disabled and need to be explicitly enabled when running the program.

Enabling assertion by :

java -ea classname.

**2.continue:**

continue keyword is used to stop the execution of current iteration and start the execution of next iteration in a loop.