***NAME : - OM RAJESH DEVAR***

***AF ID : - AF04953324***

***BATCH CODE : - ANP-D1544***

### “*JAVA COUSRE*”

1 : - VARIABLE AND DATA TYPES.

2 : - OPERETORS.

3 : - LOOPS.

4 : - ARRAY.

5 : - JAVA FUNCTION

# **Topic No 1 : - Variable and Data Types**

Java is a **statically typed** language → variables must be declared before use.  
There are 8 **primitive data types** supported by Java:

1. **byte** →
   * Value ranges from **-2⁷ to 2⁷ – 1**
   * Takes **1 byte**
   * Default value is **0**
2. **short** →
   * Value ranges from **–(2¹⁵) to (2¹⁵) – 1**
   * Takes **2 bytes**
   * Default value is **0**
3. **int** →
   * Value ranges from **–(2³¹) to (2³¹) – 1**
   * Takes **4 bytes**
   * Default value is **0**
4. **float** →
   * Value ranges from *(See Docs)*
   * Takes **4 bytes**
   * Default value is **0.0f**
5. **long** →
   * Value ranges from **–(2⁶³) to (2⁶³) – 1**
   * Takes **8 bytes**
   * Default value is **0**
6. **double** →
   * Value ranges from *(See Docs)*
   * Takes **8 bytes**
   * Default value is **0.0d**
7. **char** →
   * Value ranges from **0 to 65535 (2¹⁶ – 1)**
   * Takes **2 bytes** → because it supports Unicode
   * Default value is **'\u0000'**
8. **boolean** →
   * Value can be **true** or **false**
   * Size depends on **JVM**
   * Default value is **false**

Example :-1

public class P1{

    public static void main(String[] args){

        System.out.println("Hello World");

    }

}

Output:- Hello World

Example :-2

public class P2 {

    public static void main(String[] args){

    System.out.println("Enter the sum of two numbers");

    int a = 4;

    int b = 5;

    int sum = a+b;

    System.out.println(sum);

    }

}

Output : - Enter the sum of two numbers

9

## **Topic no 2 :- Opertors**

**1. Arithmetic Operators**

Used for basic mathematical operations.

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| + | Addition | a + b |
| - | Subtraction | a - b |
| \* | Multiplication | a \* b |
| / | Division | a / b |
| % | Modulus (remainder) | a % b |

**2. Relational (Comparison) Operators**

Used to compare two values.

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| == | Equal to | a == b |
| != | Not equal to | a != b |
| > | Greater than | a > b |
| < | Less than | a < b |
| >= | Greater than or equal to | a >= b |
| <= | Less than or equal to | a <= b |

**3. Logical Operators**

Used to combine multiple boolean expressions.

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| && | Logical AND | a > 10 && b < 20 |
| ` |  | ` |
| ! | Logical NOT | !(a > b) |

**4. Assignment Operators**

Used to assign values to variables.

| **Operator** | **Description** | **Example** |
| --- | --- | --- |
| = | Assign | a = 5 |
| += | Add and assign | a += 2 |
| -= | Subtract and assign | a -= 3 |
| \*= | Multiply and assign | a \*= 4 |
| /= | Divide and assign | a /= 2 |
| %= | Modulus and assign | `a %= |

Example :-

public class ShortOperatorExample {

public static void main(String[] args) {

int a = 10, b = 5;

// Arithmetic

System.out.println("Add: " + (a + b));

// Relational

System.out.println("a > b: " + (a > b));

// Logical

System.out.println("true && false: " + (true && false));

// Assignment

a += 2;

System.out.println("a after += 2: " + a);

// Unary

System.out.println("b++: " + (b++)); // prints 5, then b = 6

System.out.println("++b: " + (++b)); // b becomes 7

// Ternary

int max = (a > b) ? a : b;

System.out.println("Max: " + max);

// instanceof

String name = "Java";

System.out.println("Is name a String? " + (name instanceof String));

}

}

Output:-

Add: 15

a > b: true

true && false: false

a after += 2: 12

b++: 5

++b: 7

Max: 12

Is name a String? true

# **Topic 3:-loop**

**Types of Loops**

**Primarily there are three types of loops in Java:**

1. **While loop**
2. **Do-while loop**
3. **For loop**

**🔹 While Loop**

java

CopyEdit

while (boolean condition) {

// Statement

}

➡️ *This keeps executing as long as the condition is true.*

If the condition never becomes false, the while loop keeps getting executed.  
**Such a loop is known as an infinite loop.**

**🔹 Do-while Loop**

This loop is similar to a while loop **except the fact that it is guaranteed to execute at least once.**

java

CopyEdit

do {

// code

} while (condition); // ← Note this semicolon

* while → checks the condition & executes the code
* do → executes the code & then checks the condition

**🔹 For Loop**

The syntax of a **for loop** looks like this:

java

CopyEdit

for (initialization; check/condition; update) {

// code

}

Example :1

**1. While Loop Example**

public class WhileLoopExample {

public static void main(String[] args) {

int i = 100;

while (i <= 200) {

System.out.println(i);

i++;

}

}

}

**Output:**

100

101

102

...

200

**2. Do-While Loop Example**

public class DoWhileLoopExample {

public static void main(String[] args) {

int i = 1;

int n = 5;

do {

System.out.println(i);

i++;

} while (i <= n);

}

}

**Output:**

1

2

3

4

5

**3. For Loop Example**

public class ForLoopExample {

public static void main(String[] args) {

for (int i = 1; i <= 10; i++) {

System.out.println(i);

}

}

}

**Output:**

1

2

3

4

5

6

7

8

9

10

# **Topic :- Array**

**Array is a collection of similar types of data**

**Use Case: Storing marks of 5 students**

java

int[] marks = new int[5]; // → dataType[] ArrName;

* marks → reference
* new int[5] → object
* 5 x 4 = 20 bytes (since each int takes 4 bytes)

css

marks → [0][1][2][3][4]

↓ ↓ ↓ ↓ ↓

object of 5 ints (0 to 4 index)

**Accessing Array Elements**

**Array elements can be accessed as follows**:

java

CopyEdit

marks[0] = 100;

marks[1] = 70;

// ...

marks[4] = 98;

*Note that index starts from 0*

**So in a nutshell, this is how array works:**

java

int[] marks; // Declaration!

marks = new int[5]; // Memory Allocation!

OR

java

int[] marks = new int[5]; // Declaration + Memory Allocation!

OR

java

int[] marks = {100, 70, 80, 71, 98}; // Declare + Initialize!

🔹 **Array indices start from 0 and go till (n - 1)**,  
where n is the size of the array.

**Example:-**

**Java Program: Store and Display Marks of 5 Students**

public class ArrayExample {

public static void main(String[] args) {

int[] marks = {85, 90, 78, 92, 88}

System.out.println("Marks of 5 students:");

for (int i = 0; i < marks.length; i++) {

System.out.println("Student " + (i + 1) + ": " + marks[i]);

}

}

}

**Output:**

Marks of 5 students:

Student 1: 85

Student 2: 90

Student 3: 78

Student 4: 92

Student 5: 88

### TOPIC 5:- JAVA FUNCTION

A **function** in Java (technically called a **method**) is a block of code that performs a specific task and can be **reused**.

returnType functionName(parameters) {

// code block

return value; // if returnType is not void

}

**Example: Java Function to Add Two Numbers**

public class FunctionExample {

// Function to add two numbers

static int add(int a, int b) {

return a + b;

}

public static void main(String[] args) {

int result = add(10, 20); // calling the function

System.out.println("Sum is: " + result);

}

}

**Output:**

Sum is: 30

EXAMPLE :-

**Java Program: Factorial Using Function**

public class FactorialExample {

static int factorial(int n) {

int fact = 1;

for (int i = 1; i <= n; i++) {

fact = fact \* i;

}

return fact;

}

public static void main(String[] args) {

int number = 5;

int result = factorial(number);

System.out.println("Factorial of " + number + " is: " + result);

}

}

**Output:**

Factorial of 5 is: 120