**Dwija Java Programming**

**Q 1. Types of Data Types in Java ?**

In Java, values are the actual data stored in variables, while data types define the type of data a variable can hold.

* **Primitive Data Types (store simple values)**
  + **byte (1 byte)** → Stores whole numbers from -128 to 127
  + **short (2 bytes)** → Stores whole numbers from -32,768 to 32,767
  + **int (4 bytes)** → Stores whole numbers from -2 billion to +2 billion
  + **long (8 bytes)** → Stores large whole numbers
  + **float (4 bytes)** → Stores decimal numbers (e.g., 3.14f)
  + **double (8 bytes)** → Stores more precise decimal numbers (e.g., 3.1415926535)
  + **char (2 bytes)** → Stores single characters (e.g., 'A', '9')
  + **boolean (1 bit)** → Stores true or false values
* **Non-Primitive Data Types (objects with methods)**
  + **String** → Stores sequences of characters (e.g., "Hello").
  + **Arrays** → Stores multiple values of the same type.

**Example Program:**

public class **DataTypesExample** {

public static void main(String[] args) {

int age = 14; // Integer value

double pi = 3.14159; // Decimal value

char grade = 'A'; // Single character

boolean isPassed = true; // True or false value

String name = "Alice"; // String value

System.out.println("Age: " + age);

System.out.println("Pi: " + pi);

System.out.println("Grade: " + grade);

System.out.println("Passed: " + isPassed);

System.out.println("Name: " + name);

}

}

**Q 2. Operators in Java**

Operators in Java are special symbols used to perform operations on variables and values. Java has different types of operators:

**1. Arithmetic Operators (used for mathematical calculations)**

| **Operator** | **Description** | **Example (a = 10, b = 5)** | **Result** |
| --- | --- | --- | --- |
| + | Addition | a + b | 15 |
| - | Subtraction | a - b | 5 |
| \* | Multiplication | a \* b | 50 |
| / | Division | a / b | 2 |
| % | Modulus (Remainder) | a % b | 0 |

**2. Relational (Comparison) Operators (used to compare values)**

| **Operator** | **Description** | **Example (a = 10, b = 5)** | **Result** |
| --- | --- | --- | --- |
| == | Equal to | a == b | false |
| != | Not equal to | a != b | true |
| > | Greater than | a > b | true |
| < | Less than | a < b | false |
| >= | Greater than or equal to | a >= b | true |
| <= | Less than or equal to | a <= b | false |

**3. Logical Operators (used for boolean values)**

| **Operator** | **Description** | **Example (x = true, y = false)** | **Result** |
| --- | --- | --- | --- |
| && | Logical AND | x && y | false |
| || | Logical OR | x || y | Logical OR |
| ! | Logical NOT | !x | false |

**4. Assignment Operators (used to assign values)**

| **Operator** | **Description** | **Example (a = 10)** | **Equivalent to** |
| --- | --- | --- | --- |
| = | Assign value | a = 5 | a = 5 |
| += | Add and assign | a += 5 | a = a + 5 |
| -= | Subtract and assign | a -= 5 | a = a - 5 |
| \*= | Multiply and assign | a \*= 5 | a = a \* 5 |
| /= | Divide and assign | a /= 5 | a = a / 5 |
| %= | Modulus and assign | a %= 5 | a = a % 5 |

**5. Increment & Decrement Operators (used to increase/decrease values)**

| **Operator** | **Description** | **Example (a = 10)** | **Result** |
| --- | --- | --- | --- |
| ++ | Increment | a++ or ++a | 11 |
| -- | Decrement | a-- or --a | 9 |

**Example Java Program**

public class **JavaOperators** {

public static void main(String[] args) {

int a = 10, b = 5;

boolean x = true, y = false;

// Arithmetic Operators

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

System.out.println("Multiplication: " + (a \* b));

// Relational Operators

System.out.println("Is a greater than b? " + (a > b));

// Logical Operators

System.out.println("x AND y: " + (x && y));

System.out.println("x OR y: " + (x || y));

// Assignment Operators

a += 5;

System.out.println("New value of a after +=5: " + a);

// Increment & Decrement

a++;

System.out.println("Value after increment: " + a);

}

}

**Q 3. Input in Java**

In Java, we use the **Scanner class** from the java.util package to take user input from the keyboard.

**1. Importing Scanner Class**

Before using the Scanner, we must import it at the beginning of the program:

**Syntax: import java.util.Scanner;**

**2. Creating a Scanner Object**

To read input, we create a Scanner object:

**Syntax: Scanner sc = new Scanner(System.in);**

**3. Reading Different Types of Input**

| **Method** | **Data Type** | **Example** |
| --- | --- | --- |
| nextInt() | Integer | int age = sc.nextInt(); |
| nextDouble() | Decimal Number | double pi = sc.nextDouble(); |
| next() | Single word | String name = sc.next(); |
| nextLine() | Full sentence | String sentence = sc.nextLine(); |
| nextBoolean() | Boolean (true/false) | boolean isPassed = sc.nextBoolean(); |

**Example Java Program**

**import java.util.Scanner; // Import Scanner class**

public class **UserInputExample** {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in); // Create Scanner object

// Taking different types of input

System.out.print("Enter your name: ");

String name = sc.nextLine(); // Reads full name

System.out.print("Enter your age: ");

int age = sc.nextInt(); // Reads integer

System.out.print("Enter your marks: ");

double marks = sc.nextDouble(); // Reads decimal number

System.out.print("Did you pass? (true/false): ");

boolean isPassed = sc.nextBoolean(); // Reads boolean value

// Displaying the inputs

System.out.println("\nStudent Details:");

System.out.println("Name: " + name);

System.out.println("Age: " + age);

System.out.println("Marks: " + marks);

System.out.println("Passed: " + isPassed);

sc.close(); // Closing scanner

}

}

**Explanation:**

1. sc.nextLine() → Reads full name with spaces.
2. sc.nextInt() → Reads an integer value.
3. sc.nextDouble() → Reads a decimal number.
4. sc.nextBoolean() → Reads a boolean value (true or false).
5. sc.close() → Closes the scanner to free system resources.

**Q 4. Mathematical Library Methods in Java**

Java provides the Math class in the java.lang package, which contains various mathematical functions. These methods can be used directly without creating an object.

**1. Common Math Methods**

| **Method** | **Description** | **Example** | **Output** |
| --- | --- | --- | --- |
| Math.abs(x) | Absolute value | Math.abs(-5) | 5 |
| Math.max(a, b) | Maximum of two numbers | Math.max(10, 20) | 20 |
| Math.min(a, b) | Minimum of two numbers | Math.min(10, 20) | 10 |
| Math.pow(a, b) | Power (a^b) | Math.pow(2, 3) | 8.0 |
| Math.sqrt(x) | Square root | Math.sqrt(25) | 5.0 |
| Math.cbrt(x) | Cube root | Math.cbrt(27) | 3.0 |
| Math.round(x) | Rounds to nearest integer | Math.round(4.7) | 5 |
| Math.floor(x) | Rounds down | Math.floor(4.9) | 4.0 |
| Math.ceil(x) | Rounds up | Math.ceil(4.1) | 5.0 |
| Math.random() | Generates a random number (0.0 to 1.0) | Math.random() | 0.6574 (Example) |

**2. Example Java Program**

public class **MathMethodsExample** {

public static void main(String[] args) {

double num1 = -25.5, num2 = 4.7, num3 = 16;

System.out.println("Absolute Value: " + Math.abs(num1));

System.out.println("Maximum of 10 and 20: " + Math.max(10, 20));

System.out.println("Minimum of 10 and 20: " + Math.min(10, 20));

System.out.println("Power of 2^3: " + Math.pow(2, 3));

System.out.println("Square root of 16: " + Math.sqrt(num3));

System.out.println("Rounded value of 4.7: " + Math.round(num2));

System.out.println("Floor of 4.7: " + Math.floor(num2));

System.out.println("Ceiling of 4.7: " + Math.ceil(num2));

System.out.println("Random number: " + Math.random());

}

}

**Q 5. Conditional Statements in Java - if Construct**

In Java, conditional statements allow the program to make decisions based on conditions. The **if construct** is one of the most commonly used conditional statements.

**1. Types of if Constructs in Java**

**1.1 Simple if Statement**

* Executes a block of code only if the condition is true.

public class **IfExample** {

public static void main(String[] args) {

int age = 16;

if (age >= 18) {

System.out.println("You are eligible to vote.");

}

System.out.println("This statement is outside the if block.");

}

}

**Output:**  
This statement is outside the if block. (Since age is less than 18, the if block does not execute.)

**1.2 if-else Statement**

* Adds an alternative block that executes if the condition is false.

public class **IfElseExample** {

public static void main(String[] args) {

int age = 16;

if (age >= 18) {

System.out.println("You are eligible to vote.");

} else {

System.out.println("You are not eligible to vote.");

}

}

}

**Output:**  
You are not eligible to vote.

**1.3 if-else if-else Statement**

* Used when there are multiple conditions to check.

public class **IfElseIfExample** {

public static void main(String[] args) {

int marks = 85;

if (marks >= 90) {

System.out.println("Grade: A+");

} else if (marks >= 80) {

System.out.println("Grade: A");

} else if (marks >= 70) {

System.out.println("Grade: B");

} else {

System.out.println("Grade: C");

}

}

}

**Output:**  
Grade: A (because marks = 85)

**1.4 Nested if Statement**

* An if statement inside another if statement.

public class **NestedIfExample** {

public static void main(String[] args) {

int num = 10;

if (num > 0) {

if (num % 2 == 0) {

System.out.println("Positive Even Number");

}

}

}

}

**Output:**  
Positive Even Number

**Key Points:**

✅ **if Statement** → Executes if the condition is true.  
✅ **if-else Statement** → Chooses between two options.  
✅ **if-else if-else Statement** → Handles multiple conditions.  
✅ **Nested if** → Checks conditions inside another if.

**Q 6. Iterative Constructs in Java – for Loop**

A **loop** is used to execute a block of code multiple times. The **for loop** is one of the most commonly used iterative constructs in Java.

**1. Syntax of for Loop**

for (initialization; condition; update) {

// Code to be executed in each iteration

}

**Initialization:** Declares and initializes the loop variable.

**Condition:** Checks whether the loop should continue running.

**Update:** Modifies the loop variable after each iteration.

**2. Example: Printing Numbers from 1 to 5**

public class **ForLoopExample** {

public static void main(String[] args) {

for (int i = 1; i <= 5; i++) {

System.out.println("Number: " + i);

}

}

}

**Output:**

Number: 1

Number: 2

Number: 3

Number: 4

Number: 5

**3. Variations of for Loop**

**3.1 Reverse Loop (Counting Down)**

public class ReverseLoop {

public static void main(String[] args) {

for (int i = 5; i >= 1; i--) {

System.out.println(i);

}

}

}

**Output:**

5

4

3

2

1

**3.2 Loop with Step Incrementation**

public class **StepLoop** {

public static void main(String[] args) {

for (int i = 2; i <= 10; i += 2) {

System.out.println(i);

}

}

}

**Output:**

2

4

6

8

10

**3.3 Nested for Loop**

A for loop inside another for loop is called a **nested loop**.

public class **NestedLoop** {

public static void main(String[] args) {

for (int i = 1; i <= 3; i++) { // Outer loop

for (int j = 1; j <= 2; j++) { // Inner loop

System.out.println("i = " + i + ", j = " + j);

}

}

}

}

Output:

i = 1, j = 1

i = 1, j = 2

i = 2, j = 1

i = 2, j = 2

i = 3, j = 1

i = 3, j = 2

**Key Points:**

✅ **for Loop** → Best for when the number of iterations is known.  
✅ **Reverse for Loop** → Runs in decreasing order.  
✅ **Step for Loop** → Increments by custom values.  
✅ **Nested for Loop** → Used for patterns and grids.