Java Expressions

Definition:
A **Java expression** is a combination of **variables, constants, operators, and method calls** that evaluate to a single value. Expressions are fundamental building blocks of Java programs.

Types of Java Expressions

1. **Arithmetic Expressions**
2. **Relational Expressions**
3. **Logical Expressions**
4. **Bitwise Expressions**
5. **Assignment Expressions**
6. **Conditional (Ternary) Expressions**
7. **Method Call Expressions**
8. **Instanceof Expressions**
9. **Lambda Expressions**

---

1. Arithmetic Expressions

An **arithmetic expression** involves arithmetic operators (`+`, `-`, `*`, `/`, `%`) and evaluates to a numerical value.

Example

```
class ArithmeticExample {
    public static void main(String[] args) {
        int a = 10, b = 5;
        int sum = a + b;  // Addition expression
        int product = a * b;  // Multiplication expression
        double division = (double) a / b; // Division expression

        System.out.println("Sum: " + sum);
        System.out.println("Product: " + product);
        System.out.println("Division: " + division);
    }
}
```

Output:

```
Sum: 15
Product: 50
Division: 2.0
```

---

2. Relational Expressions

A **relational expression** compares two values using **relational operators** (`==`, `!=`, `>`, `<`, `>=`, `<=`) and returns a **boolean value (true/false)**.

Example

```
class RelationalExample {
    public static void main(String[] args) {
        int x = 10, y = 5;
        boolean result = x > y; // Evaluates to true

        System.out.println("Is x greater than y? " + result);
    }
}
```

Output:

```
Is x greater than y? true
```

---

3. Logical Expressions

A **logical expression** involves **logical operators** (`&&`, `||`, `!`) and evaluates to a boolean value (`true` or `false`).

Example

```
class LogicalExample {
    public static void main(String[] args) {
        boolean a = true, b = false;

        System.out.println("Logical AND (a && b): " + (a && b)); // false
        System.out.println("Logical OR (a || b): " + (a || b));  // true
        System.out.println("Logical NOT (!a): " + (!a));  // false
    }
}
```

Output:

```
Logical AND (a && b): false
Logical OR (a || b): true
Logical NOT (!a): false
```

---

4. Bitwise Expressions

A **bitwise expression** uses **bitwise operators** (`&`, `|`, `^`, `~`, `<<`, `>>`) to perform operations on bits.

Example

```
class BitwiseExample {
    public static void main(String[] args) {
        int a = 5, b = 3; // Binary: 5 = 0101, 3 = 0011

        System.out.println("Bitwise AND (a & b): " + (a & b)); // 1 (0001)
```

```
        System.out.println("Bitwise OR (a | b): " + (a | b));  // 7 (0111)
        System.out.println("Bitwise XOR (a ^ b): " + (a ^ b)); // 6 (0110)
        System.out.println("Bitwise Complement (~a): " + (~a)); // -6 (in
2's complement)
    }
}
```

Output:

```
Bitwise AND (a & b): 1
Bitwise OR (a | b): 7
Bitwise XOR (a ^ b): 6
Bitwise Complement (~a): -6
```

---

5. Assignment Expressions

An **assignment expression** assigns a value to a variable using assignment operators (`=`, `+=`, `-=`, `*=`, `/=`, `%=`).

Example

```
class AssignmentExample {
    public static void main(String[] args) {
        int x = 10;
        x += 5; // Same as x = x + 5;

        System.out.println("Updated value of x: " + x);
    }
}
```

Output:

```
Updated value of x: 15
```

---

6. Conditional (Ternary) Expressions

A **conditional expression** (`condition ? expr1 : expr2`) evaluates a condition and returns one of two values.

Example

```
class TernaryExample {
    public static void main(String[] args) {
        int a = 10, b = 20;
        int min = (a < b) ? a : b;

        System.out.println("Minimum value: " + min);
    }
}
```

Output:

```
Minimum value: 10
```

---

7. Method Call Expressions

A **method call expression** calls a method, which may return a value.

Example

```
class MethodExample {
    static int square(int n) {
        return n * n;
    }

    public static void main(String[] args) {
        int result = square(5); // Method call expression
        System.out.println("Square of 5: " + result);
    }
}
```

Output:

```
Square of 5: 25
```

---

8. `instanceof` Expressions

The **`instanceof`** operator checks if an object is an instance of a class.

Example

```
class Parent {}
class Child extends Parent {}

class InstanceofExample {
    public static void main(String[] args) {
        Parent obj = new Child();

        System.out.println("Is obj an instance of Parent? " + (obj
instanceof Parent));
        System.out.println("Is obj an instance of Child? " + (obj
instanceof Child));
    }
}
```

Output:

```
Is obj an instance of Parent? true
Is obj an instance of Child? true
```

---

9. Lambda Expressions (Java 8+)

A **lambda expression** is a concise way to define an anonymous function.

Example

```
interface MathOperation {
    int operation(int a, int b);
}
```

```
class LambdaExample {
    public static void main(String[] args) {
        // Using lambda expression
        MathOperation add = (a, b) -> a + b;
        System.out.println("Addition: " + add.operation(5, 3));
    }
}
```

Output:

```
Addition: 8
```

- **Java expressions** are combinations of **variables, constants, operators, and method calls** that evaluate to a value.
- They are classified into different types based on their functionality.
- Expressions help in writing **efficient, modular, and readable code**.