SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23CSB101- OBJECT ORIENTED PROGRAMMING

I YEAR /II SEMESTER

Unit I – INTRODUCTION TO OOP AND JAVA

Topic : OPERATORS

Operators

Operator is a special symbol that tells the compiler to perform specific mathematical or logical Operation. Java supports following lists of operators.

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators
- Increment and Decrement Operators
- Shift Operators
- Ternary or Conditional Operators
- instanceof operator

Operators

	Operator	Type
unary operator →	+ +, - -	Unary operator
Binary operator	+, -, *, /, %	Arithmetic perator
	<, <=, >, >=, ==, !=	Relational operator
	&&, | |, !	Logical operator
	&, |, <<, >>, ~, ^	Bitwise operator
	=, +=, - =, *=, /=, %=	Assignment operator
Ternary operator →	?:	Ternary or conditional operator

Operator	Example (int A=8, B=3)	Result
+	A+B	11
-	A-B	5
*	A*B	24
/	A/B	2
%	A%4	0

Arithmetic operators

Operators	Example (int A=8, B=3)	Result
<	A<B	False
<=	A<=10	True
>	A>B	True
>=	A<=B	False
==	A== B	False
!=	A!=(-4)	True

Relational operators

Operator	Example (int A=8, B=3, C=-10)	Result
&&	(A<B) && (B>C)	False
||	(B!=-C) || (A==B)	True
!	!(B<=-A)	True

Logical operators

Operator	Example (int A=8, B=3)	Result
+=	A+=B or A=A+B	11
-=	A-=3 or A=A+3	5
*=	A*=7 or A=A*7	56
/=	A/=B or A=A/B	2
%=	A%=5 or A=A%5	3
=a=b	Value of b will be assigned to a	

Assignment operators

Increment and Decrement Operator

++ , Increments by 1.

Post-Increment: Uses value first, then increments.

Pre-Increment: Increments first, then uses value.

-- , Decrements by 1.

Post-Decrement: Uses value first, then decrements.

Pre-Decrement: Decrements first, then uses value.

Bitwise Operators

Used to perform the manipulation of individual bits of a number and with any of the integer types.

Used when performing update and query operations of the Binary indexed trees.

& (Bitwise AND) – returns bit-by-bit AND of input values.

| (Bitwise OR) – returns bit-by-bit OR of input values.

^ (Bitwise XOR) – returns bit-by-bit XOR of input values.

~ (Bitwise Complement) – inverts all bits (one’s complement).

Shift Operators

Used to shift the bits of a number left or right, thereby multiplying or dividing the number by two, respectively.

<< (Left shift) – Shifts bits left, filling 0s (multiplies by a power of two).

>> (Signed right shift) – Shifts bits right, filling 0s (divides by a power of two), with the leftmost bit depending on the sign.

>>> (Unsigned right shift) – Shifts bits right, filling 0s, with the leftmost bit always 0.

Ternary Operator

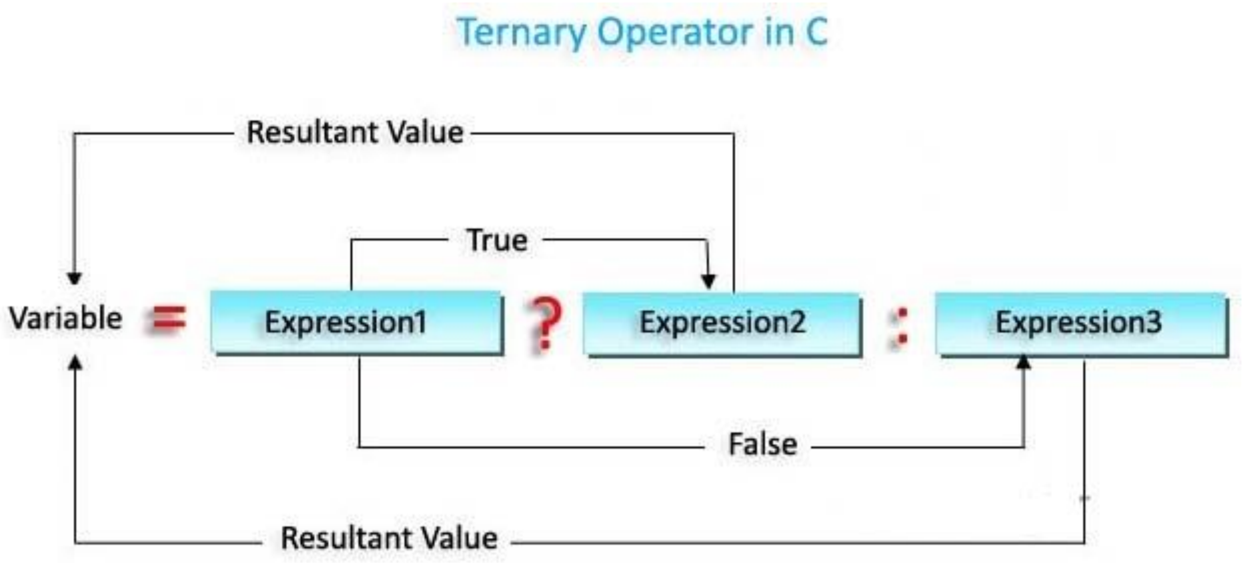
The ternary operator is an operator that takes three arguments. The first argument is a comparison argument, the second is the result upon a true comparison, and the third is the result upon a false comparison. Ternary operator is shortened way of writing an if-else statement.

Syntax: expression-1 ? expression-2 : expression-3

Example:

```
a<b ? printf("a is less") : printf("a is greater");
```

Ternary Operator



instanceof operator

Used for type checking. It can be used to **test** if an **object is an instance of a class, a subclass, or an interface**. The general format ,

object **instanceof** class/subclass/interface

instanceof operator

```
public class Geeks
{
    public static void main(String[] args)
    {

        Person obj1 = new Person();
        Person obj2 = new Boy();

        // As obj is of type person, it is not an
        // instance of Boy or interface
        System.out.println("obj1 instanceof Person: "
                           + (obj1 instanceof Person));
        System.out.println("obj1 instanceof Boy: "
                           + (obj1 instanceof Boy));
        System.out.println("obj1 instanceof MyInterface: "
                           + (obj1 instanceof MyInterface));

        // Since obj2 is of type boy,
        // whose parent class is person
        // and it implements the interface Myinterface
        // it is instance of all of these classes
```

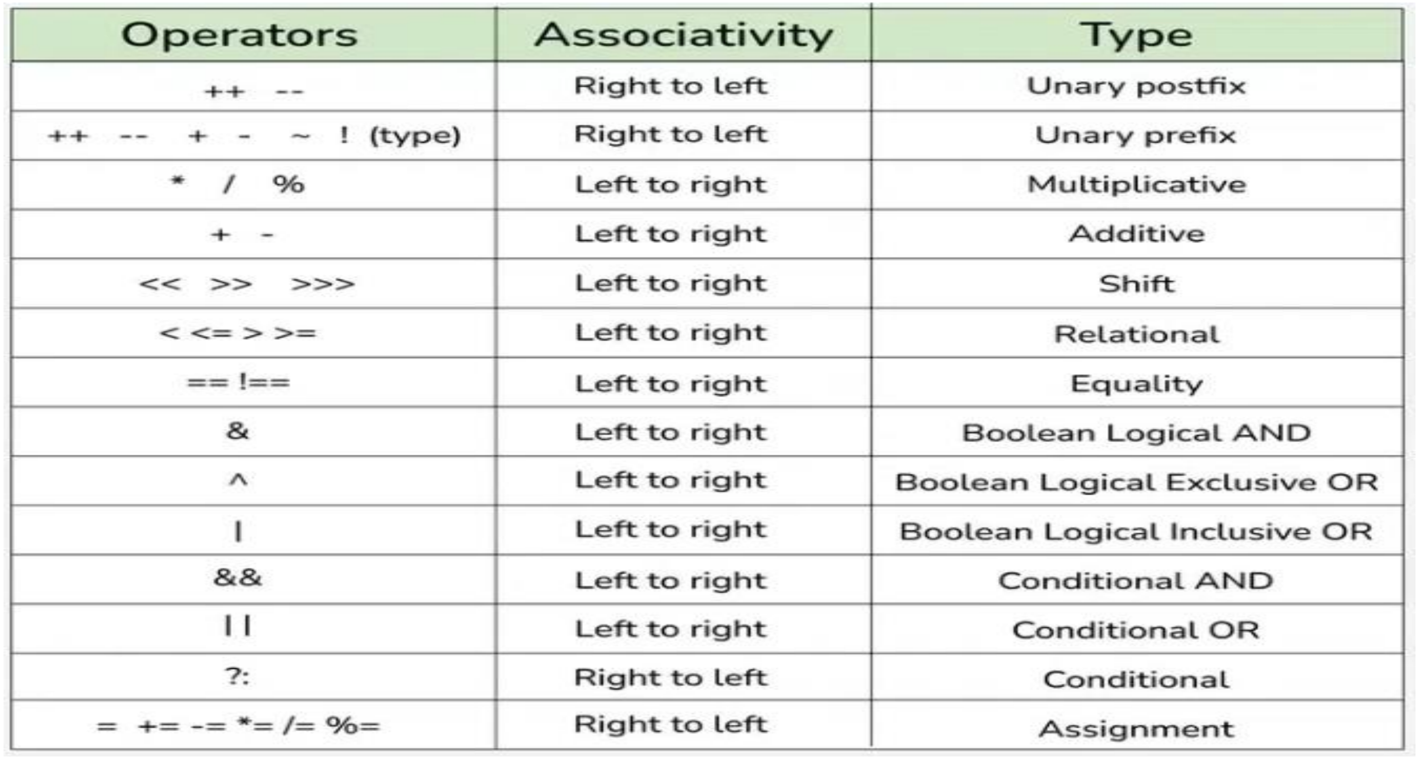
```
 System.out.println("obj2 instanceof Person: "
                           + (obj2 instanceof Person));
        System.out.println("obj2 instanceof Boy: "
                           + (obj2 instanceof Boy));
        System.out.println("obj2 instanceof MyInterface: "
                           + (obj2 instanceof MyInterface));
    }
}

// Classes and Interfaces used
// are declared here
class Person {
}

class Boy extends Person implements MyInterface {
}

interface MyInterface {
}
```

instanceof operator

Output:

```
obj1 instanceof Person: true
obj1 instanceof Boy: false
obj1 instanceof MyInterface: false
obj2 instanceof Person: true
obj2 instanceof Boy: true
obj2 instanceof MyInterface: true
```

Precedence and Associativity of Java Operators



Operators	Associativity	Type
++ --	Right to left	Unary postfix
++ -- + - ~ ! (type)	Right to left	Unary prefix
* / %	Left to right	Multiplicative
+ -	Left to right	Additive
<< >> >>>	Left to right	Shift
< <= > >=	Left to right	Relational
== !==	Left to right	Equality
&	Left to right	Boolean Logical AND
^	Left to right	Boolean Logical Exclusive OR
|	Left to right	Boolean Logical Inclusive OR
&&	Left to right	Conditional AND
||	Left to right	Conditional OR
?:	Right to left	Conditional
= += -= *= /= %=	Right to left	Assignment

Program

Write a Java program to print the results of the following operations.

Test Data:

a. -5 + 8 * 6

b. (55+9) % 9

c. 20 + -3*5 / 8

d. 5 + 15 / 3 * 2 - 8 % 3

Program

```
import java.util.Scanner;

public class Exercise33 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Prompt the user to input an integer
        System.out.print("Input an integer: ");

        // Read the integer from the user
        long n = input.nextLong();

        // Calculate and display the sum of the digits
        System.out.println("The sum of the digits is: " + sumDigits(n));
    }
```

```
    public static int sumDigits(long n) {
        int sum = 0;

        // Calculate the sum of the digits
        while (n != 0) {
            sum += n % 10;
            n /= 10;
        }

        return sum;
    }
}
```

Program

```
public class Exercise4 {
    public static void main(String[] args) {
        // Calculate and print the result of the expression: -5 + 8 * 6
        System.out.println(-5 + 8 * 6);

        // Calculate and print the result of the expression: (55 + 9) % 9
        System.out.println((55 + 9) % 9);

        // Calculate and print the result of the expression: 20 + -3 * 5 / 8
        System.out.println(20 + -3 * 5 / 8);

        // Calculate and print the result of the expression: 5 + 15 / 3 * 2 - 8 % 3
        System.out.println(5 + 15 / 3 * 2 - 8 % 3);
    }
}
```

Array Program

```
public class ArrayExample {
    public static void main(String[] args) {

        // Declare and initialize an array of integers
        int[] numbers = {10, 20, 30, 40, 50};

        // Print the elements of the array using a for loop
        System.out.println("Elements of the array:");
        for (int i = 0; i < numbers.length; i++) {
            System.out.println("numbers[" + i + "] = " + numbers[i]);
        }

        // Modify an element in the array
        numbers[2] = 100;  // Change the 3rd element (index 2) to 100
```

```
        // Print the modified array
        System.out.println("\nModified array:");
        for (int i = 0; i < numbers.length; i++) {
            System.out.println("numbers[" + i + "] = " + numbers[i]);
        }

        // Using enhanced for loop (for-each) to iterate over the
array
        System.out.println("\nUsing enhanced for loop (for-
each):");
        for (int number : numbers) {
            System.out.println(number);
        }
    }
}
```

Array Program

Output:

```
Elements of the array:
numbers[0] = 10
numbers[1] = 20
numbers[2] = 30
numbers[3] = 40
numbers[4] = 50

Modified array:
numbers[0] = 10
numbers[1] = 20
numbers[2] = 100
numbers[3] = 40
numbers[4] = 50

Using enhanced for loop (for-each):
10
20
100
40
50
```

Thank you