Lecture 12

Operators in Java

- Arithmetic Operators
- Relational operators
- Logical Operators
- Bitwise Operator
- Assignment operators
- Bit shift operators
- instance of operator

Assignment Operators

These operators used to assign some values to variables.

→ Variable operator value (expression);

expression must be evaluated to a single value

The value on right hand side of the operator is assigned to the variable to its left side.

e.g:- a = 5 ✓ b = 2 ✓

or a = a+1; ✓ c = b ✓

2 = b ✗ incorrect

↓

Left side of = operator must be a variable. We can not take any constant (fixed) value at left side

These operators have **right to left** associativity.

e.g:- int c = 1; int a, b;

a = b = c;

↓

first c value would be assigned to b then b value would be assigned to a

NOTE:- Right hand side value must be a constant (literal) or must be declared/initialized before using it.

eg:- int a = 1;
int b = a; → Here Right side is a variable not a fixed value but this variable has already been initialized. So it will not give any error.

Compound Assignment Operators:-

+=, -=, *=, /=, %=

```
int a=1, b=3, c=10;
a = b+c*10;   ?
a += c*10;    ?
```

Relational Operators:-

(Comparison operators)

< > <= >= == !=

- These are binary operators.
- used to check the relation between two operands like less than, equal to, greater than etc.
- or we can say used to compare values of 2 operands.

```
expression 1  relational operator  expression 2
```

- Relational operators always return true or false (boolean values)
- These operators can be used with primitive numeric data types. (int, char, float, double, byte, short, long)

IMP- Equality operators (==, !=) can be applied to boolean types as well.

eg:-

```
boolean isProgrammingEasy = false;
System.out.println(isProgrammingEasy == true);   → false
System.out.println(isProgrammingEasy != false);  → false
```

output

IMP • Equality Operators can also be used to compare objects (non-primitive data types) references

eg:-

```
class Employee {
    String name;
    int empId;
}
```

IMP Associativity is left-to-right. Lower Precedence than Arithmetic Operators but higher than assignment operators.

```
public static void main(String[] args) {
    Employee empl = new Employee();
    Employee emp2 = new Employee();
    empl.name = "Jenny";
    empl.empId = 1001;
    emp2.name = "Jenny";
    emp2.empId = 1001;
    System.out.println("empl == emp2 :" + (empl == emp2));
    System.out.println("empl != emp2 :" + (empl != emp2));
```

Output
false
true

[Here addresses are compared rather than the values]

- This is also known as identity comparison.

We can also check for null reference.

eg:- In above example, we can write.

```
→ System.out.println(empl == null);
→ System.out.println(empl != null);
```

output
false
true

Basically Relational operators are used in control-flow statements like if, else statements.

eg:-

```
float cgpa = 5.9f;
if (cgpa >= 6.0) {
    System.out.println("You can sit in Placement Drive.");
}
else {
    System.out.println("Improve your CGPA & try next time.");
}
System.out.println("Out of if");
```

Practice Time:-

```
int a = 10, b = 7;
int result = a<b;   // error as relational operator returns boolean values So we can't assign it to an integer variable.
char ch1 = 'A', ch2 = 'B';
Boolean result = ch1 < ch2;   → true
String str1 = "Jenny", str2 = "khatri";
System.out.println (str1 == str2);   → false
System.out.println (str1 != str2);   → true
System.out.println (str1 < str2);   → invalid error
```

NOTE:- [Only Equality operators can be applied to strings]

Challenge Time / Coding Exercises:-

① WAP to find maximum among 3 numbers using ternary operator.

```
max = (a>b) ? (a>c) ? a : c : (b>c) ? b : c ;
```

② WAP to find area & circumference of circle.

③ WAP for BMI Calculator.

$$BMI = \frac{\text{weight (kg)}}{\text{height (m)}^2}$$

Logical Operators

Used to perform logical operations on boolean values

3 Logical operators
- logical AND (&&)
- logical OR (||)
- logical NOT (!)

- In Real life also we face such situations where we apply logical operators.

Buy 2 get 1 free

Tshirt AND (&&) Tshirt = free Tshirt.

laptop OR (||) phone = get 20% off.

NOT (!) good CIBIL Score = Can't apply for home loan

The function of these operators are similar to AND, OR & NOT gate in digital electronics.

Logical && and logical OR (||) are used to combine two or more conditions

e.g:- age > 24 && salary > 50000 then yes for marriage } Case I

But without logical operators things would have been tought. Same above conditions we can check with relational operators as well.

→ Case II

```
if (age > 24) {
    if (salary > 50000) {
        approve
    }
}
```

→ this is more complex looking code than Case I

logical AND (&&) :-

Syntax:- operand 1 logical operator operand 2

→ Operands must be boolean values (true or false). Generally operands are conditions.

eg:- if (age >= 18 && salary > 50000)
condition Condition

→ logical operators returns boolean value (true or false). logical AND returns true if both the conditions are true. If any one of the two condition is false, the operator returns false

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

NOTE:- While using AND operator, the second condition is not evaluated if the first one is false and this is known as short-circuiting

→
```
int a = 10, b = 20, c = 30;
if ( a > b && a > c)
    then a is maximum
```

logical OR (||) :-

operand 1 || operand 2

This operator returns true when one of the two conditions under consideration is satisfied or is true. And if both conditions are true then also it gives true.

e.g:- age = 24
salary = 50000

(age > 24 || salary > 40000)
false ↓ true
returns true

a	b	a || b
true	true	true
true	false	false
false	true	false
false	false	false

In this operator, if the first condition is true, then second will not be evaluated. This is short-circuiting effect.

e.g:- a = 10, b = 20, c = 30, d = 40

```
if ((a<b) || (++c<d))
    { True
        System.out.println("Inside if");
    }
System.out.println(c);
```
will print → 30

Logical NOT (!):- !(Condition) or ! expression

It will invert the value of the expression means if expression evaluates to true then ! operator return false and if the expression evaluates to false then ! operator returns true.

e.g:- a = 10
!a → error because logical operators can not be applied to integer

!(a > 10) → true
!(a <= 10) → false

NOTE:- Logical operators have lower precedence than Arithmetic & Relational Operators but higher precedence than assignment operators. And associativity is from left-to right.

```
int age = 24;
double salary = 60000;
int CIBILScore = 100;
if (salary > 50000 && CIBILScore > 200 || age > 24)
    then loan approved
```

⇒ ((a && b) && c) && d

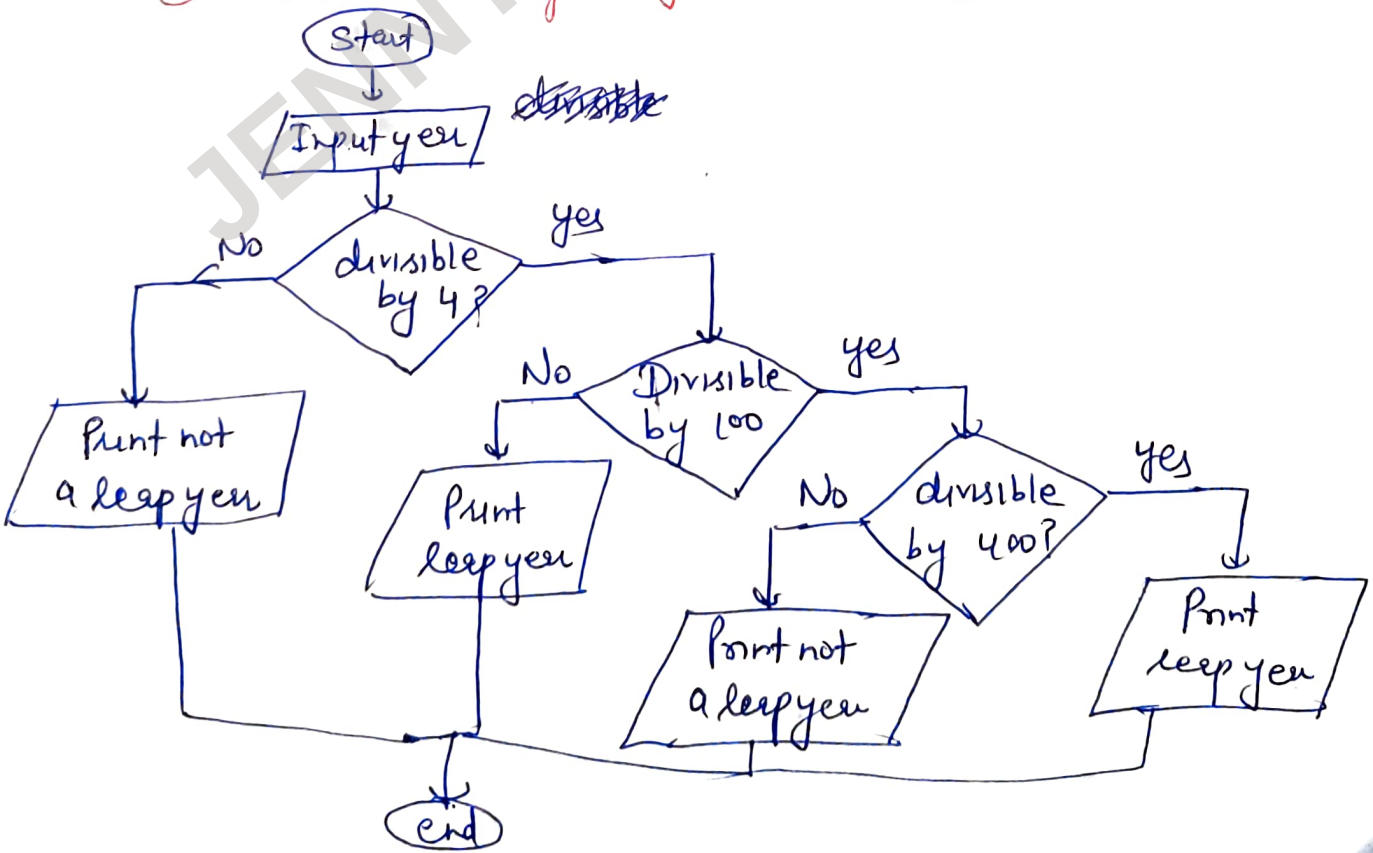
⇒ (a &&b) || (c && d)

⇒ ((a &&b) && c) || d

→ Grouping is done based on precedence rules but evaluation would be from left-to-right always.

Coding Exercise:-

① WAP to check given year is leap year or not



(2) WAP to check if a person is eligible for a discount on a movie ticket. The discount is available if the person is under 18 or over 60 years old.

(3) WAP to check if a person is eligible to vote. A person is eligible to vote if they are 18 years or older and are a citizen of country.

(4) Write a program to check if a person is allowed to enter a concert. A person is allowed if they are 18 years or older and either have a ticket or are invited by a VIP.