DAY 3

OPERATORS AND CONTROL FLOW

OPERATORS:

Operators are special symbols (characters) that carry out operations on operands (variables and values). For example, + is an operator that performs addition

- **Assignment Operator**

Assignment operators are used in Java to assign values to variables. For example,

```
int age;

age = 5;
```

The assignment operator assigns the value on its right to the variable on its left. Here, 5 is assigned to the variable age using = operator.
There are other assignment operators too. However, to keep things simple, we will learn other assignment operators later in this article.

EXAMPLE:-

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

        int number1, number2;

        // Assigning 5 to number1
        number1 = 5;
        System.out.println(number1);

        // Assigning value of variable number2 to number1
        number2 = number1;
        System.out.println(number2);
    }
}
```

OUTPUT:-

```
5
5
```

- ## Arithmetic Operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc.

Operator	Meaning
+	Addition (also used for string concatenation)
-	Subtraction Operator
*	Multiplication Operator
/	Division Operator
%	Remainder Operator

Example :- Arithmetic Operator

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

        double number1 = 12.5, number2 = 3.5, result;

        // Using addition operator
        result = number1 + number2;
        System.out.println("number1 + number2 = " + result);

        // Using subtraction operator
        result = number1 - number2;
        System.out.println("number1 - number2 = " + result);

        // Using multiplication operator
        result = number1 * number2;
        System.out.println("number1 * number2 = " + result);

        // Using division operator
        result = number1 / number2;
        System.out.println("number1 / number2 = " + result);

        // Using remainder operator
        result = number1 % number2;
        System.out.println("number1 % number2 = " + result);
    }
}
```

Output:

```
number1 + number2 = 16.0
number1 - number2 = 9.0
number1 * number2 = 43.75
number1 / number2 = 3.5714285714285716
number1 % number2 = 2.0
```

- ## Unary Operators

The unary operator performs operations on only one operand.

Operator	Meaning
+	Unary plus (not necessary to use since numbers are positive without using it)
-	Unary minus: inverts the sign of an expression
++	Increment operator: increments value by 1
--	decrement operator: decrements value by 1
!	Logical complement operator: inverts the value of a boolean

Example : Unary Operator

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

        double number = 5.2, resultNumber;
        boolean flag = false;

        System.out.println("+number = " + +number);
        // number is equal to 5.2 here.

        System.out.println("-number = " + -number);
        // number is equal to 5.2 here.

        // ++number is equivalent to number = number + 1
        System.out.println("number = " + ++number);
        // number is equal to 6.2 here.

        // -- number is equivalent to number = number - 1
```

```
        System.out.println("number = " + --number);
        // number is equal to 5.2 here.

        System.out.println("!flag = " + !flag);
        // flag is still false.
    }
}
```

Output:

```
+number = 5.2
-number = -5.2
number  = 6.2
number  = 5.2
!flag  = true
```

Example : Unary Operator

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

        double number = 5.2;

        System.out.println(number++);
        System.out.println(number);

        System.out.println(++number);
        System.out.println(number);
    }
}
```

Output:

```
5.2
6.2
7.2
7.2
```

- ## Increment and Decrement Operator

You can also use `++` and -- operator as both prefix and postfix in Java.

The `++` operator increases value by 1 and `--` operator decreases the value by 1.

```
int myInt = 5;

++myInt    // myInt becomes 6

myInt++    // myInt becomes 7

--myInt    // myInt becomes 6

myInt--    // myInt becomes 5
```

Simple enough until now. However, there is a crucial difference while using increment and decrement operators as prefix and postfix.

- ## Equality and Relational Operators

  The equality and relational operators determine the relationship between the two operands. It checks if an operand is greater than, less than, equal to, not equal to and so on. Depending on the relationship, it is evaluated to either `true` or `false`.

==	5 == 3 is evaluated to false
!=	5 != 3 is evaluated to true
>	5 > 3 is evaluated to true
<	5 < 3 is evaluated to false

>=	greater than or equal to	5 >= 5 is evaluated to true
<=	less than or equal to	5 <= 5 is evaluated to true

Equality and relational operators are used in decision making and loops

Example : Equality and Relational Operators

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

        int number1 = 5, number2 = 6;

        if (number1 > number2) {
            System.out.println("number1 is greater than number2.");
        }
        else {
            System.out.println("number2 is greater than number1.");
        }
    }
}
```

Output:

```
number2 is greater than number1.
```

Here, we have used `>` operator to check if `number1` is greater than `number2` or not.

Since `number2` is greater than `number1`, the expression `number1 > number2` is evaluated to `false`.

Hence, the block of code inside `else` is executed and the block of code inside `if` is skipped.

If you didn't understand the above code, don't worry. You will learn it in detail in Java if...else .

For now, just remember that the equality and relational operators compare two operands and is evaluated to either `true` or `false`.

- instanceof Operator

In addition to relational operators, there is also a type comparison operator `instanceof` which compares an object to a specified type. For example,

Example : instanceof Operator

Here's an example of instanceof operator.

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

        String test = "asdf";
        boolean result;

        result = test instanceof String;
        System.out.println("Is test an object of String? " + result);
    }
}
```

Output:

```
Is test an object of String? true
```

Here, since the variable `test` is of `String` type. Hence, the `instanceof` operator returns `true`. To learn more, visit Java instanceof.

- **Logical Operators**

The logical operators `||` (conditional-OR) and `&&` (conditional-AND) operate on boolean expressions. Here's how they work.

Operator	Description	Example
`||`	conditional-OR: `true` if either of the boolean expression is `true`	`false || true` is evaluated to `true`
`&&`	conditional-AND: true if all boolean expressions are `true`	`false && true` is evaluated to `false`

Example : Logical Operators

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

        int number1 = 1, number2 = 2, number3 = 9;
        boolean result;

        // At least one expression needs to be true for the result to be true
        result = (number1 > number2) || (number3 > number1);

        // result will be true because (number3 > number1) is true
        System.out.println(result);

        // All expression must be true from result to be true
        result = (number1 > number2) && (number3 > number1);

        // result will be false because (number1 > number2) is false
        System.out.println(result);
    }
}
```

Output:

```
true
false
```

- ## Ternary Operator

The conditional operator or ternary operator `?:` is shorthand for the `if-then-else` statement. The syntax of the conditional operator is:

```
variable = Expression ? expression1 : expression2
```

Here's how it works.

- If the `Expression` is `true`, `expression1` is assigned to the `variable`.
- If the `Expression` is `false`, `expression2` is assigned to the `variable`.

Example : Ternary Operator

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

        int februaryDays = 29;
        String result;

         result =  (februaryDays == 28) ? "Not a leap year" : "Leap year";
         System.out.println(result);
    }
}
```

Output:

```
Leap year
```

- ## Bitwise and Bit Shift Operators

To perform bitwise and bit shift operators in Java, these operators are used.

Operator	Description
~	Bitwise Complement
<<	Left Shift
>>	Right Shift
>>>	Unsigned Right Shift
&	Bitwise AND
^	Bitwise exclusive OR
|	Bitwise inclusive OR

CONTROL FLOW

In Java there are three types of control flow statements:

1. Decision making statements: if, if else, nested if else
2. Looping statements: while, for, do-while
3. Branching statements: return, break, continue

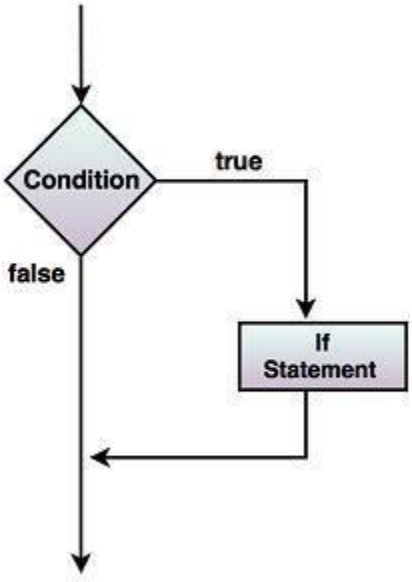
- if statement

- It instructs the program to execute only certain part of the code if the condition is true.
- If boolean value is true then statement1 will be executed.

Syntax:

```
if (boolean expr)
{
    statement 1:      //expr true
}
statement 2;
```

Flow Diagram



- if else statement

In this case, when if clause is false then else part will be executed.

Syntax:

```
if (boolean expr)
{
```

```
    statement 1;      //expr true
}
else
{
    statement 2;      //expr false
}
```

Flow Diagram

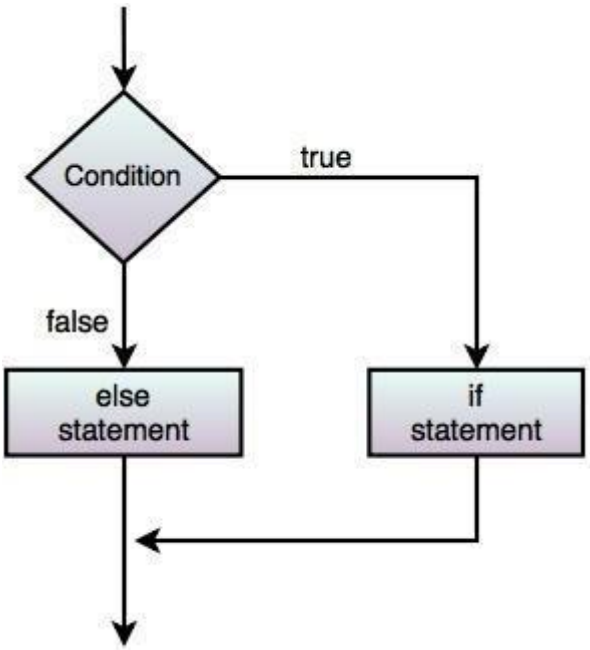


Fig: Flow diagram of if else statement

Nested if else

Syntax:

```
if (boolean expr1)
{
    statement 1;      // expr1 true
}
else if (boolean expr2)
{
    Statement 2;      // expr 1 false and expr 2 true
}
else
{
```

```
    statement 3;      // expr 2 false
}
```

- Switch Case Statement

It is type of selection mechanism that provides multi-way selection.

Flow Diagram

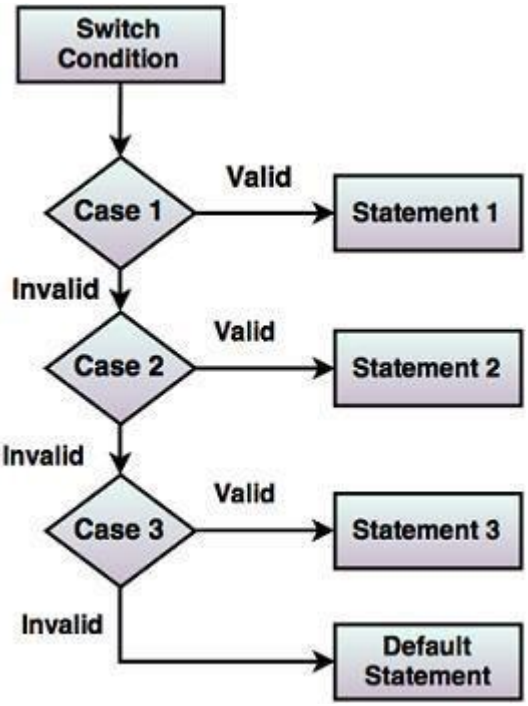


Fig: Flow diagram of switch statement