Working with Bitwise AND (&), Bitwise OR (|) and Bitwise X-OR (^) :
---------------------------------------------------------------
a) Logical Operator works with **boolean** value only.
   Example : System.out.println(5 && 6); //**error**

b) Bitwise AND(&), Bitwise OR(|) and Bitwise X-OR (^) can work
   with number and boolean both types of values.

   i) When it works with **boolean** type then It is known as Boolean operator.
   ii) When it works with **number** type then It is called Bitwise operator.

   Example :
   System.out.println(5 && 6); //Bitwise AND(&);
   System.out.println(true & true); //Boolean Operator(&);

Truth table of AND gate, OR Gate and X-OR gate :

| AND Gate (A.B) | | | OR Gate (A+B) | | | Exclusive OR Gate (A⊕B) | | |
|---|---|---|---|---|---|---|---|---|
| A | B | Output | A | B | Output | A | B | Output |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| | | | | | | [Same input, Output is 0] | | |

```java
public class Test
{
    public static void main(String[] args)
    {
        System.out.println(true ^ false);
    }
}
```

Here we will get the output based on the input i.e Same input output will be false, In the above
program we will get the output as true.

```
System.out.println(5 & 6);
System.out.println(5 | 6);
System.out.println(5 ^ 6);
```

```
Binary of 5  :   1   0   1
Binary of 6  :   1   1   0
                 ------------
                 1   0   0      (5 & 6)
```

```
Binary of 5  :   1   0   1
Binary of 6  :   1   1   0
                 ------------
                 1   1   1      (5 | 6)
```

```
Binary of 5  :   1   0   1
Binary of 6  :   1   1   0
                 ------------
                 0   1   1      (5 ^ 6)
```

```java
public class Test
{
    public static void main(String[] args)
    {
        System.out.println(5 & 6); //4
        System.out.println(5 | 6); //7
        System.out.println(5 ^ 6); //3
    }
}
```

Bitwise Complement Operator :
-------------------------------
* It will not work with boolean type

```java
public class Test
{
    public static void main(String[] args)
    {
        System.out.println(~5); //-6
        System.out.println(~-6); //5
    }
}
```

```
                    + 2
                    - 1
                   -----
                    8   4   2   1
                    0   1   1   0
```

Binary of 2 is : 0  0  1  0
Let say  A = 0   1   1   0
1's complement of A = 1   1   0   1
2's complement of A = _____+  1
                      1   1   1   0   (-2)

```
Binary of +2 =  0   0   1   0
Binary of -2 =  1   1   1   0   (-2)
                -----------------
                0   0   0   0
```

Member access Operator (.)
----------------------------
* The member access operator is used to access the member of the class by using object reference
OR class name.
Example :
Integer.parseInt("100");

Here parseInt() is a static method of Integer class so to access this parseInt() method we should
use . (dot) operator. [We will see in more details in Object Oriented Programming]

new Operator :
----------------
* By using new operator we can create Object of a class.
* In order to access the non static member of the class, First of all we need to create object.
* new keyword IS RESPONSIBLE TO PROVIDE THE DEFAULT VALUES FOR NON STATIC
VARIABLES.

```java
class Welcome
{
    int x = 100; //non static variable

    public void greet() //non static method
    {
        System.out.println("Hello Everyone!");
    }
}
public class Test
{
    public static void main(String[] args)
    {
        Welcome w = new Welcome();
        System.out.println(w.x);
        w.greet();
    }
}
```

instanceof operator [It is relational Operator]
------------------------------------------------
[We will cover this topic in dynamic Polymorphism]

Limitation of if-else :
------------------------
The major drawback with if condition is, It checks the condition again and again so It increases the
burden over CPU so we introduced switch case statement to reduce the overhead of the CPU.

Switch Statement in java [Till JDK 17V] :
------------------------------------------
It is a selective statement in java i.e, we can select one statement among the available statements.

While writing the cases, break is optional but if we use break then the control will move from out of
the switch body.

We can write default or if any statement is not matching then default will be executed but [It is
optional. [default is compulsory, If we take switch statement as an expression]

Valid types of switch expression :
-----------------------------------
byte
short
int
char
data
Short
Integer
Character
enum [JDK 1.5]
String [JDK 1.7]
Pattern Matching [JDK 17] [enhanced]

Invalid types are :
--------------------
long
float
double
boolean

Some important points :
------------------------
a) We cannot accept duplicate case value.

b) The label of case must be constant OR literals.

c) The break statement is used to exit from the switch block. It is optional but recommended [JDK
17 (->) break is not required]

d) The default case is optional and executes if no case matches in the switch expression.

//Programs :
-------------
```java
package com.ravi.switch_statement;

import java.util.Scanner;

public class SwitchDemo1
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a character : ");
        char colour = sc.next().toLowerCase().charAt(0);

        switch(colour)
        {
        case 'r' :
            System.out.println("Red");
            break;
        case 'g' :
            System.out.println("Green");
            break;
        case 'b' :
            System.out.println("Blue");
            break;
        case 'w' :
            System.out.println("White");
            break;
        default:
            System.out.println("No colour");
            break;
        }
        System.out.println("Completed");
        sc.close();
    }
}
```

```java
package com.ravi.switch_statement;

import java.util.Scanner;

public class SwitchDemo2
{
    public static void main(String[] args)
    {
        System.out.println("111**Main Menu**");
        System.out.println("111**100 Police**");
        System.out.println("111**101 Fire**");
        System.out.println("111**102 Ambulance**");
        System.out.println("111**108 Railway**");
        System.out.println("111**101 Women's Helpline**");

        Scanner sc = new Scanner(System.in);

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

        int choice = Integer.parseInt(sc.nextLine());

        switch(choice)
        {
        case 100 -> System.out.println("Police Services...");
        case 101 -> System.out.println("Fire Services...");
        case 102 -> System.out.println("Ambulance Services...");
        case 119 -> System.out.println("Railway Services...");
        case 101 -> System.out.println("Women's Helpline");

        default -> System.out.println("No Valid choice");
        }
        sc.close();
    }
}
```

```java
package com.ravi.switch_statement;

import java.util.Scanner;

public class SwitchDemo3
{
    public static void main(String[] args)
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the Language you want to learn : ");
        String language = sc.nextLine().toUpperCase();

        switch (language)
        {
        case 'C' -> System.out.println("You selected C language");
        case "JAVA" -> System.out.println("You selected Java language");
        case "HTML" -> System.out.println("You selected HTML language");
        default -> System.out.println("Language not supported.");
        }
        sc.close();
    }
}
```