Some basic program on Operators :

To find out the binary of any number, rule is :
---
* Have right to left
* Divide the number by half
* Ignore the remainder
* put 0 for even and 1 for odd

```
1    2.5     9.9     11.5     23.5     47
1    0       1       1        1        1
```

```
public class Test
{
    public static void main(String[] args)
    {
        int x = 19;
        int y = x++;
        System.out.println(x + " " + y);
    }
}
---
public class Test
{
    public static void main(String[] args)
    {
        int x = 19;
        int y = --x;
        System.out.println(x + " " + y);
    }
}
---
public class Test
{
    public static void main(String[] args)
    {
        int x = 25;
        int y = x+20; //error
        System.out.println(x + " " + y);
    }
}
---
public class Test
{
    public static void main(String[] args)
    {
        int x = 25;
        int y = ++(++x);
        System.out.println(x + " " + y);
    }
}
---
public class Test
{
    public static void main(String[] args)
    {
        char ch = 'k';
        ch++;
        System.out.println(ch);
    }
}
---
public class Test
{
    public static void main(String[] args)
    {
        double d1 = 12.89;
            d1++;
        System.out.println(d1); //13.89

    }
}
Note : Increment and Decrement operator we can apply on any data type
       except boolean data type.
---
public class Test
{
    public static void main(String[] args)
    {
        while(false)
        {
            System.out.println("Hello");
        }
        System.out.println("World");

    }
}
Note : Compilation error , Unreachable Statement
---
public class Test
{
    public static void main(String[] args)
    {
    boolean b = false;

        while(b)
        {
            System.out.println("Hello");
        }
        System.out.println("World");

    }
}
---
public class Test
{
    public static void main(String[] args)
    {
        final int x = 25;
          final int y = 20;

        while(x > y)
        {
            System.out.println("x is Greater than y");
        }
        System.out.println("Hello World");
    }
}
Note : Compilation error, Unreachable Code because final keyword
---
public class Test
{
    public static void main(String[] args)
    {
            do
        {
            int x = 1;
            System.out.println("Hello");
            x++;
        }
        while(x++<10);
    }
}
Note : x is declared inside do block so we cannot access outside of do block.
---
public class Test
{
    public static void main(String[] args)
    {
        int x = 1;
            do
            {
                System.out.println("Hello");
                x++;

                boolean b = true;
                if(b==false)
                {
                }
            }
            while (x++<10);
    }
}
Note : We can use assignment operator while working with if condition, It
       actually demands boolean value.
---
Numbers Promotion :
---
While working with Arithmetic Operator (+, -, *, /, %) and Unary minus operator, after the
evaluation of the expressions, the entire expression is
promoted into a type that means we can say in order to hold the result
minimum 32 bits are required.

This rule is given to prevent from any kind of data loss.

//Program on Unary Minus Operator :

---
public class Test
{
    public static void main(String[] args)
    {
        int x = 10;
        System.out.println(-x);
    }
}
---
public class Test
{
    public static void main(String[] args)
    {
            byte b = 1;
        short s = 2;
        short t = -b * s; //error
        System.out.println(t);
    }
}
---
public class Test
{
    public static void main(String[] args)
    {
        byte b = 1;
        short s = 2;
        System.out.println(-b);
    }
}
---
public class Test
{
    public static void main(String[] args)
    {
        byte b = 128;
        byte c = -b;

        //int d = b + c;
        //System.out.println(d); //149

        byte d = (byte) (b + c);
        System.out.println(d);
    }
}
---
public class Test
{
    public static void main(String[] args)
    {
        /*
            byte b = 1;
            b = b + 2;
            System.out.println(b); */

            byte b = 2;
            b += 5; //It is valid we will get the output
            System.out.println(b);
    }
}
In the above program we are using short hand operator so we will get the result in byte format
also.
---
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.

   Example :

   System.out.println(5 & 6); //Bitwise AND(&)
   System.out.println(true & true); //Bitwise AND(&)
---
public class Test
{
    public static void main(String[] args)
    {
        int x = 5;
        if( ++x > 5 || ++x > 6)   //Logical OR
        {
            z++;
        }
        System.out.println(x); //7

        System.out.println("..................");

          y = 5;
        if( ++y > 5 | ++y > 6)   //Boolean OR
        {
            z++;
        }
        System.out.println(y); //8

    }
}
---
public class Test
{
    public static void main(String[] args)
    {
        int z = 5;
          if(++z > 5 & ++z> 6)     //Boolean AND
          {
              System.out.println("Inside If");
              z++;
          }
          System.out.println(z);
    }
}
---
//Bitwise X-OR Operator
public class Test
{
    public static void main(String[] args)
    {
        System.out.println(true ^ true); //false
        System.out.println(true ^ false); //true
    }
}
Note : we will get true if both the inputs are opposite to each other.
---
//Bitwise X-OR Operator
public class Test
{
    public static void main(String[] args)
    {
            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  (4)
      1  1  1    5 | 6  (7)
      0  1  1    5 ^ 6  (3)

The output will come based on the truth table.
---
Bitwise Complement Operator (~)

It will not work with boolean.

public class Test
{
    public static void main(String [] args)
    {
        System.out.println(~ true); //error

    }

}
---
//Bitwise complement operator ~
public class Test
{
    public static void main(String[] args)
    {
            //System.out.println(~ true); //Invalid
        System.out.println(~4); //-5
            System.out.println(~-3);//2
    }
}
---
```