Opr17.java

```
void main()
{
    int x = 15;
    int y = ++(++x);  //error Unexpected type
    IO.println("+" + y);

}
```

Opr18.java

```
void main()
{
    char ch = 'W';
    ch++;
    IO.println(ch);  //X

    char c = 'b';
    c--;
    IO.println(c);//a
}
```

Opr19.java

```
void main()
{
    double d1 = 12.12;
    d1++;
    IO.println(d1);
}
```

Note :- Increment and decrement operator we can apply with any data type except boolean.

Opr20.java

```
void main()
{
    int amount = 0;
    IO.println(amount);//0
    IO.println(++amount);//1
    IO.println(amount);//1
    IO.println(amount++);//1
    IO.println(amount);//2
}
```

Increment & decrement operator is having high order of precedence as compared to binary operator.

Opr21.java

```
void main()          //RAM i = 5

{
int i = 5;
int j = ++i + 4;  //6 + 4
IO.println("+" +j);//10 , 9
```

Note :++i always uses **old value from the memory** But i++ will use the **new value from the memory** same rule is applicable for i- **and** i+

Opr22.java

```
void main()
{
    int a = 5;
    int b = ++a + 6;
    IO.println("+" +a);//6  10

}
```

Opr23.java

```
void main()

{
    int i = 5;
    int j = --i - 4;
    IO.println("+" +j);//4   0
```

Opr24.java

```
void main()          //RAM i = 7

{
    int i = 5;
    int result = ++i + ++i;  //5 + 7

    IO.println("+" + i);//7
    IO.println("result = " + result);//12
}
```

Opr25.java

```
void main()      //RAM a = 7
{
    int a = 5;
    int b = a++ + a++;  //5 + 6

    IO.println(b);//7
    IO.println(a);//11

}
```

Opr26.java

```
void main()          //RAM x = 12
{
    int x = 10;
    int y = ++x + x++;  //11 + 11

    IO.println(x);//12
    IO.println(y);//22
```

Opr27.java

```
void main()          //RAM x = 17
{
    int x = 15;
    IO.println(++x + x++);//32
    IO.println(x);//17
    IO.println(x);//18
    int x = 15;          //x = 17
    IO.println(++x + ++x);//32
    IO.println(x);//17
}
```

Opr28.java

```
void main()      //RAM x = 3
{
    int x = 3;
    int y = ++x * 5/x--;  //4 * 5/ 4 [RAM]
    IO.println("+" +y);//5  5
}
```

Opr29.java

```
void main()      //RAM a = 4
{
    int a = 4;
    int b = a++ * 2/--a;  //4 * 2 / 4
    IO.println("+" +b);//4  2
```

Relational Operator :-

These operations are used to compare the values. The return type is **boolean.**

1) < (Greater than)

2) > (Less than)

3) >= (Greater than or equal to)

4) <= (Less than or equal to)

5) == (double equal to)] -> **equality Operator**

6) != (not equal to)]

7) **relational operator [instanceOf(caption)**

Opr30.java

```
void main()
{
    int a = 20;
    int b = 20;
    IO.println("a == b " + (a == b) );//true
    IO.println("a != b " + (a != b) );//false
    IO.println("a > b " + (a > b) );//false
    IO.println("a < b " + (a < b) );//true
    IO.println("a >= b " + (a >= b) );//true
    IO.println("a <= b " + (a <= b) );//true
}
```

What is a block in Java ?

A Block is a self defined block and it is executed automatically.
Example :-

```
{
    IO.println("Block 1");

Note : If we declare any variable inside the block then that variable we cannot use outside of the block.
        Example :-

        {
            int x = 12; //x variable we can use inside the block only

Opr31.java
void main()
{
    {
        IO.println("Block 1");

    }

    IO.println("Block 2");
}
```

CONTROL STATEMENTS IN JAVA :-

In JavaJavaJavaJ