Opt17.java

```
void main()
{
    int x = 15;
    System.out.println(++x);  //error Unsupported type
    System.out.println("+x);
}
```

Opt18.java

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

    char c = 'b';
    ++c;
    System.out.println(c);//c
}
```

Opt19.java

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

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

Opt20.java

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

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

Opt21.java

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

int i = 5;
int j = ++i + 4;   //5 + 4
System.out.println("+j);//9 , 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+

Opt22.java

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

}
```

Opt23.java

```
void main()

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

Opt24.java

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

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

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

Opt25.java

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

    System.out.println(b); //7
    System.out.println(a);//11
}
```

Opt26.java

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

    System.out.println(x); //12
    System.out.println(y);//22
```

Opt27.java

```
void main()          //RAM x = 17
{
    int x = 15;
    System.out.println(++x + x++);//32
    System.out.println(x);//17
    System.out.println(x);//17

    int x = 15;        //x = 17
    System.out.println(++x + x++); //32
    System.out.println(x);//17
}
```

Opt28.java

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

Opt29.java

```
void main()        //RAM a = 4
{
    int a = 4;
    int b = ++a * 2 / --a;  //4 * 2 / 4
    System.out.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) instanceof operator **[ClassCastException]**

Opt30.java

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

What is a block in Java ?

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

```
{
    System.out.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

Opt31.java
void main()
{
    {
        System.out.println("Block 1");

    }

    {
        System.out.println("Block 2");

    }
}
```

CONTROL STATEMENTS IN JAVA :-

In JavaJavaJavaJ