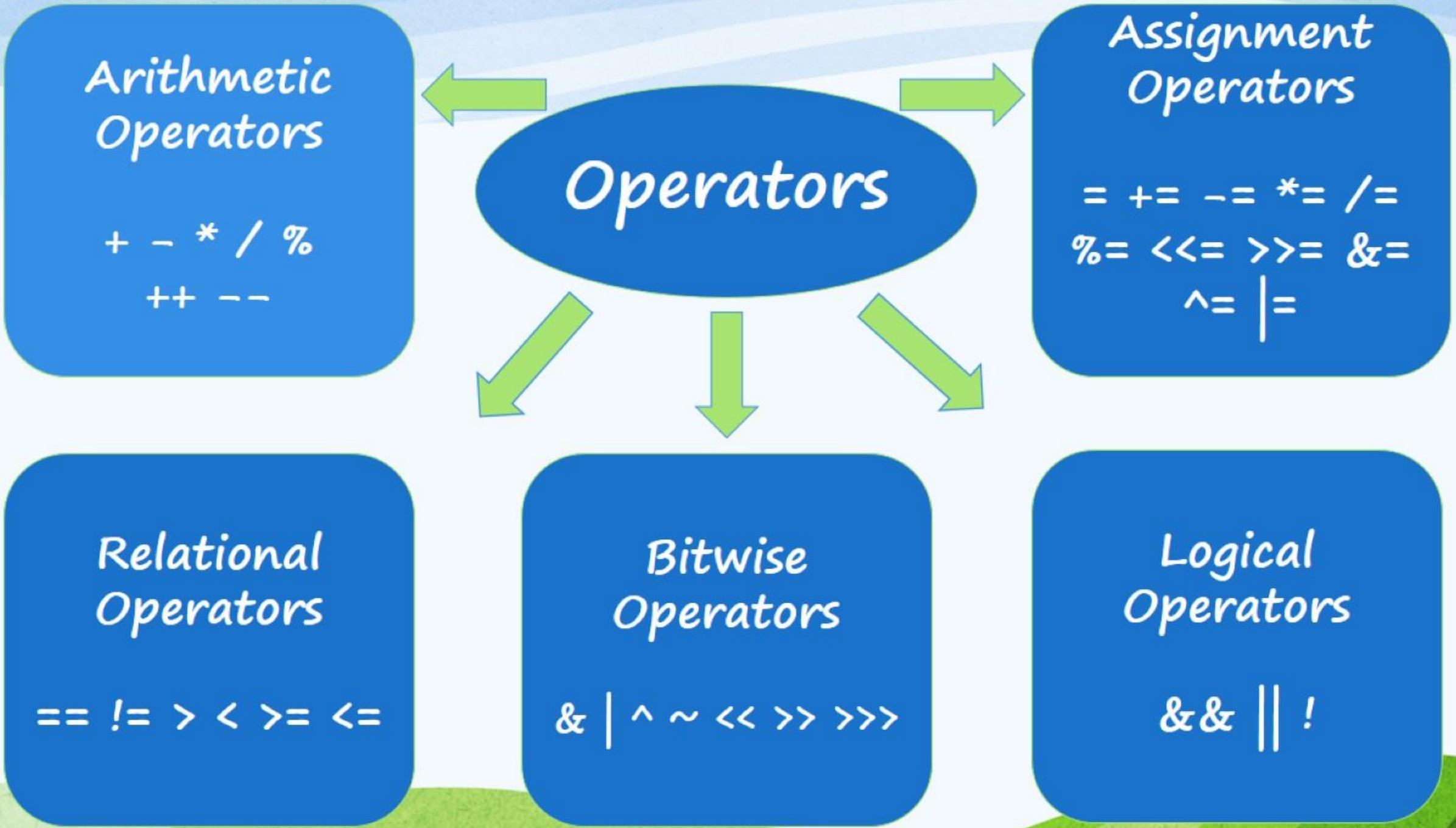
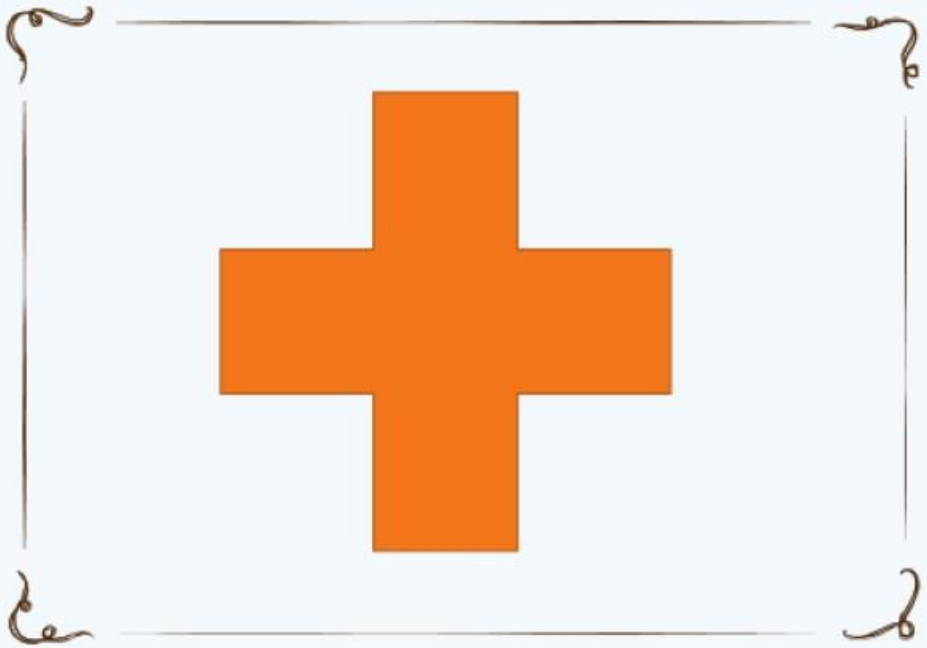

Operators in java

Operators

- Operator in java is a symbol that is used to perform operations.
- For example: +, -, *, / etc.

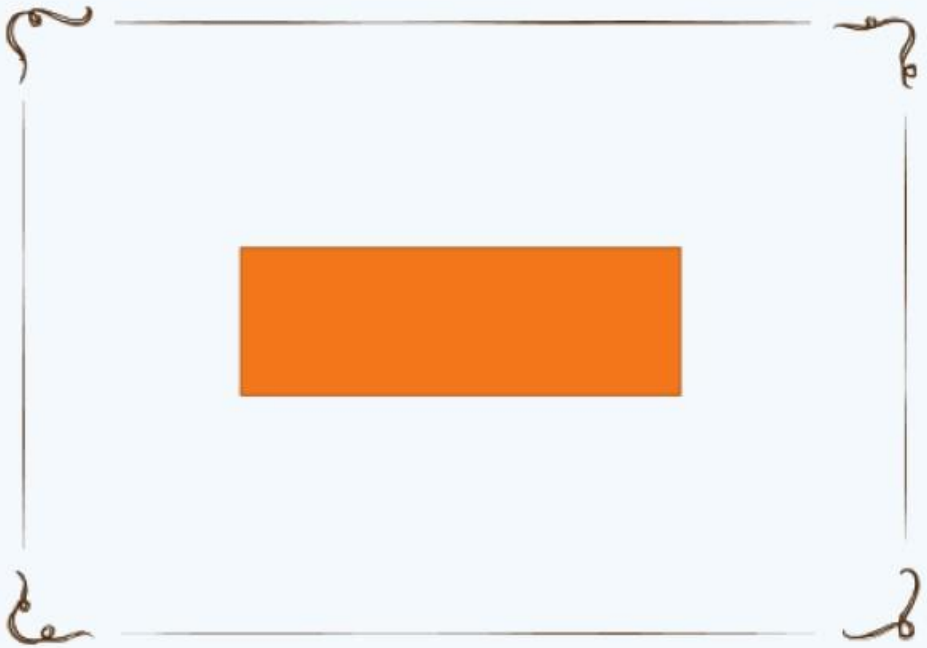

Operators
Arithmetic Operators
+ - * / %
++ --
Assignment Operators
= += -= *= /=
%= <<= >>= &=
^= |=
Relational Operators
== != > < >= <=
Bitwise Operators
& | ^ ~ << >> >>>
Logical Operators
&& || !

Arithmetic Operators



Addition

a + b



Subtraction

a - b

Example:

```
int a=9;
int b=2;
int c=a + b;
System.out.println(c);
```

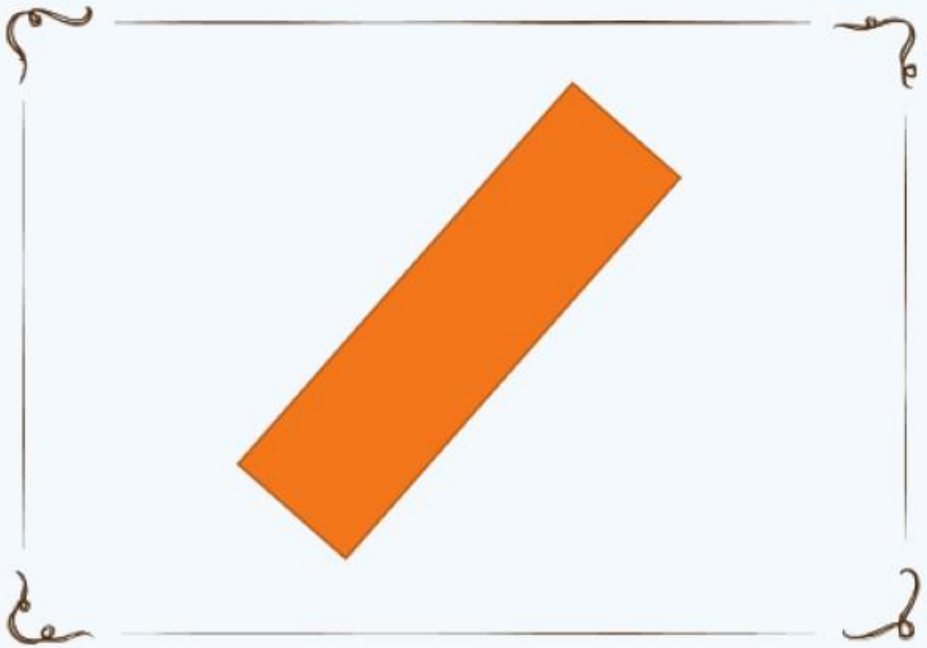
Output: 11

```
int a=9;
int b=2;
int c=a - b;
System.out.println(c);
```

Output: 7

Multiplication

a * b



Division

a / b

Example:

```
int a=9;
int b=2;
int c=a * b;
System.out.println(c);
```

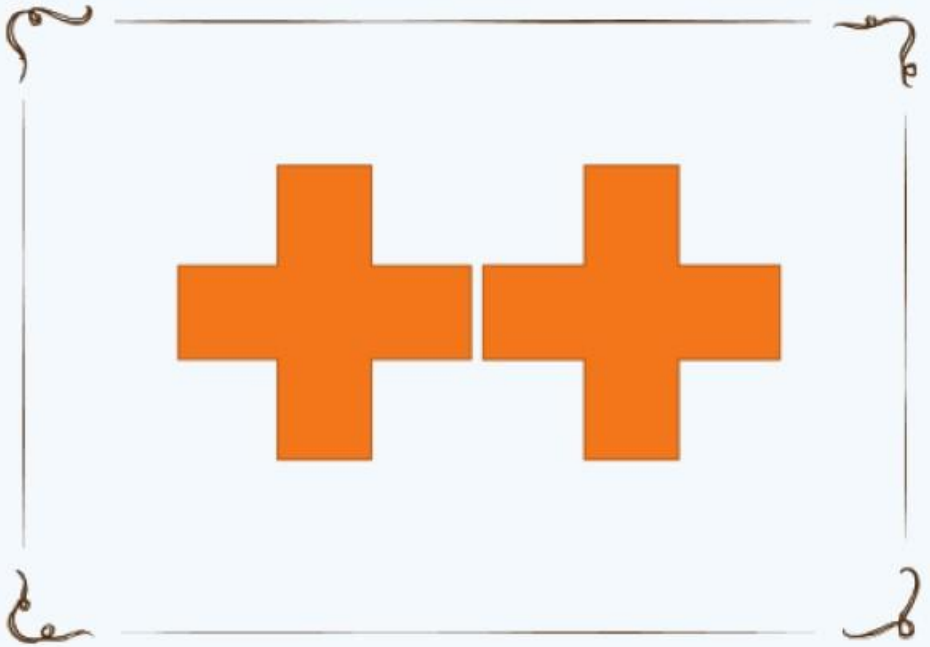
Output: 18

```
int a=9;
int b=2;
int c=a / b;
System.out.println(c);
```

Output: 4

Modulus

a % b



a++

Example:

```
int a=9;
int b=2;
int c=a % b;
System.out.println(c);
```

Output: 1

```
int a=9;
int b=2;

System.out.println(++a);
System.out.println(b++);
```

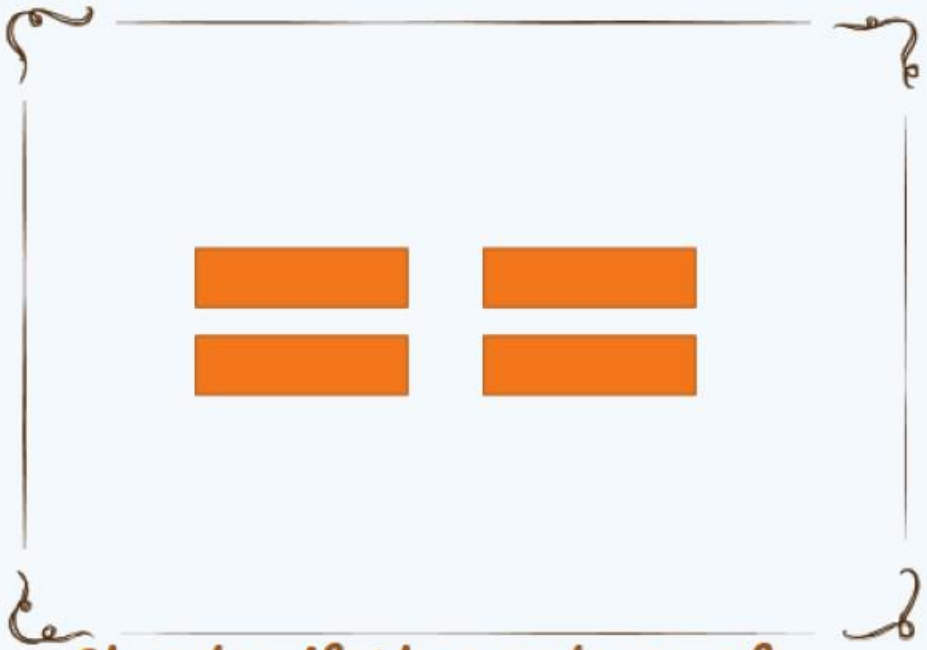
Output: 10
2

```
int a=9;
int b=2;

System.out.println(--a);
System.out.println(b--);
```

Output: 8

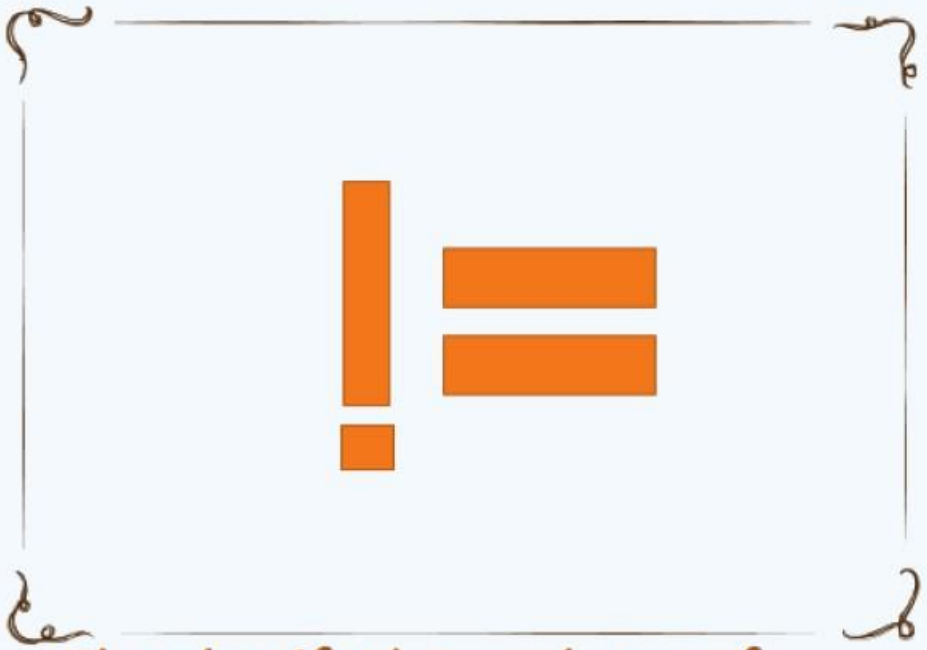
2

Relational Operators



Checks if the values of two operands are equal or not

a == b



Checks if the values of two operands are equal or not

a != b

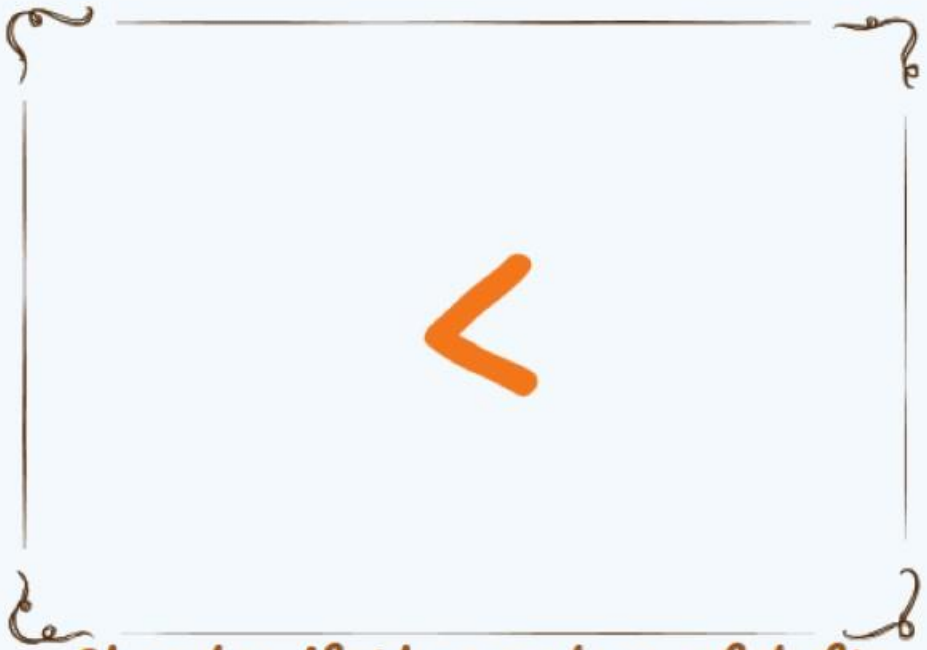
Example:

```
int a = 2;
int b = 4;
System.out.println(a==b);
```

Output: false

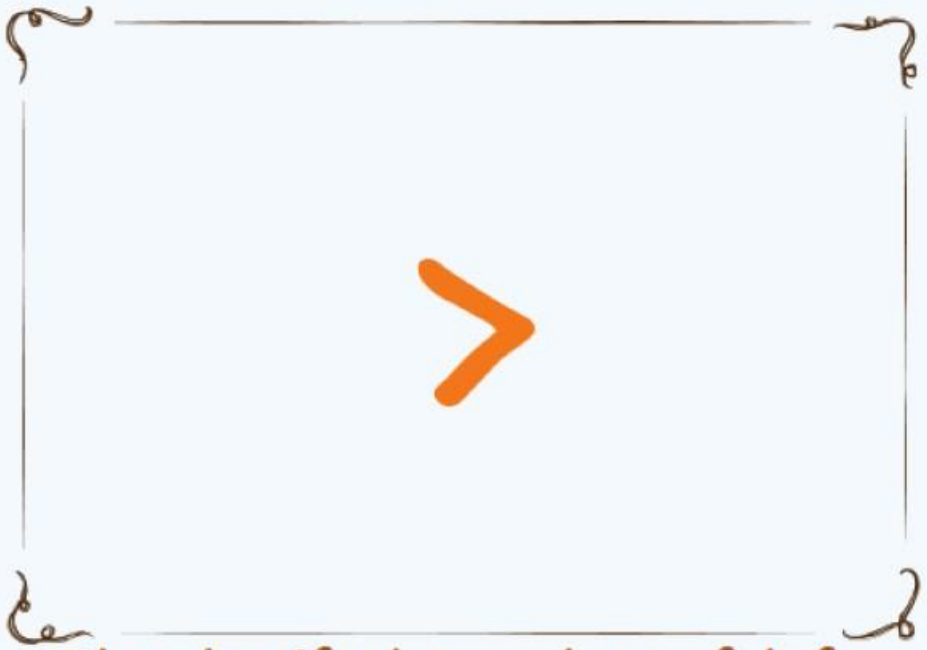
```
int a = 2;
int b = 4;
System.out.println(a!=b);
```

Output: true



Checks if the value of left operand is greater than the value of right operand

a > b



Checks if the value of left operand is less than the value of right operand

a < b

Example:

```
int a = 2;

int b = 4;
System.out.println(a>b);
```

Output: false

```
int a = 2;
int b = 4;
System.out.println(a<b);
```

Output: true

>=

Checks if the value of left operand is greater than or equal to the value of right operand

a >= b

<=

Checks if the value of left operand is less than or equal to the value of right operand

a <= b

Example:

```
int a = 2;

int b = 4;
System.out.println(a>=b);
```

Output: false

```
int a = 2;
int b = 4;
System.out.println(a<=b);
```

Output: true

Bitwise Operators

Binary AND Operator copies a bit to the result if it exists in both operands.

a & b

Binary OR Operator copies a bit if it exists in either operand.

a | b

Example:

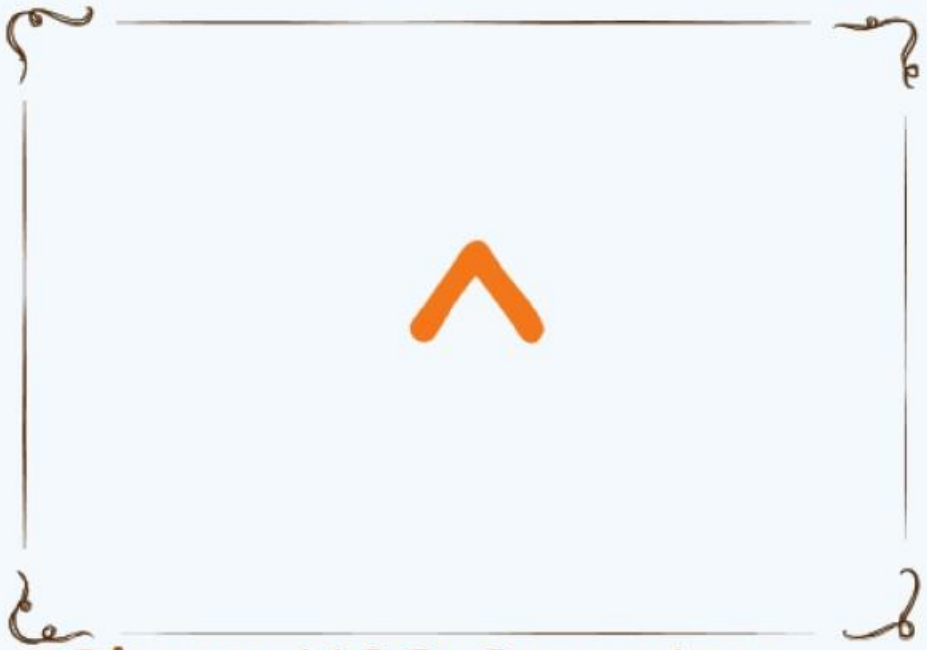
```
int a = 9;
int b = 2;
int c = a & b;
System.out.println(c);
```

```
9=1001
2=0010
&=0000
 |=1011
```

```
int a = 9;
int b = 2;
int c = a | b;
System.out.println(c);
```

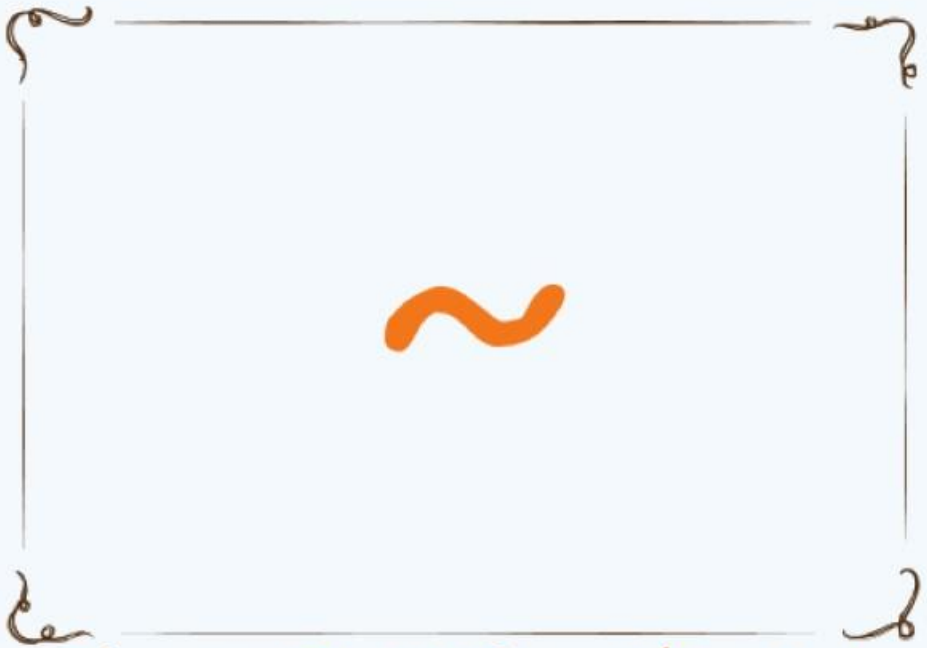
Output: 0

Output: 11



Binary XOR Operator copies the bit if it is set in one operand but not both.

a ^ b



Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.

a ~ b

Example:

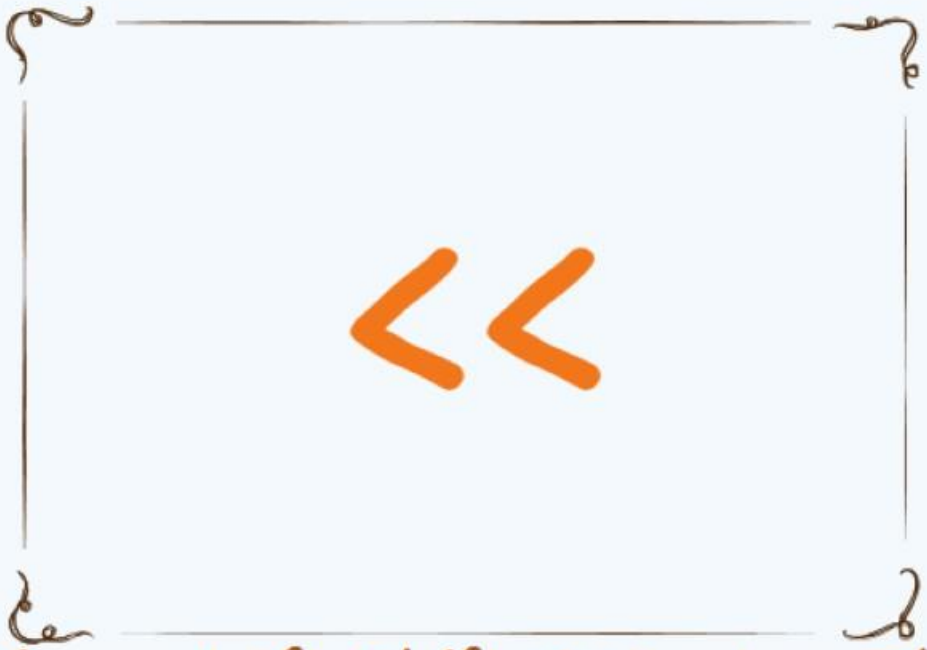
```
int a = 9;

int b = 3;

int c = a ^ b;

System.out.println(c);
```

```
9=1001
2=0010
3=0011
^=1010
~=11111
1111111
0110
```

```
int a = 9;

int b = 2;

int c = ~ a;
System.out.println(c);
```

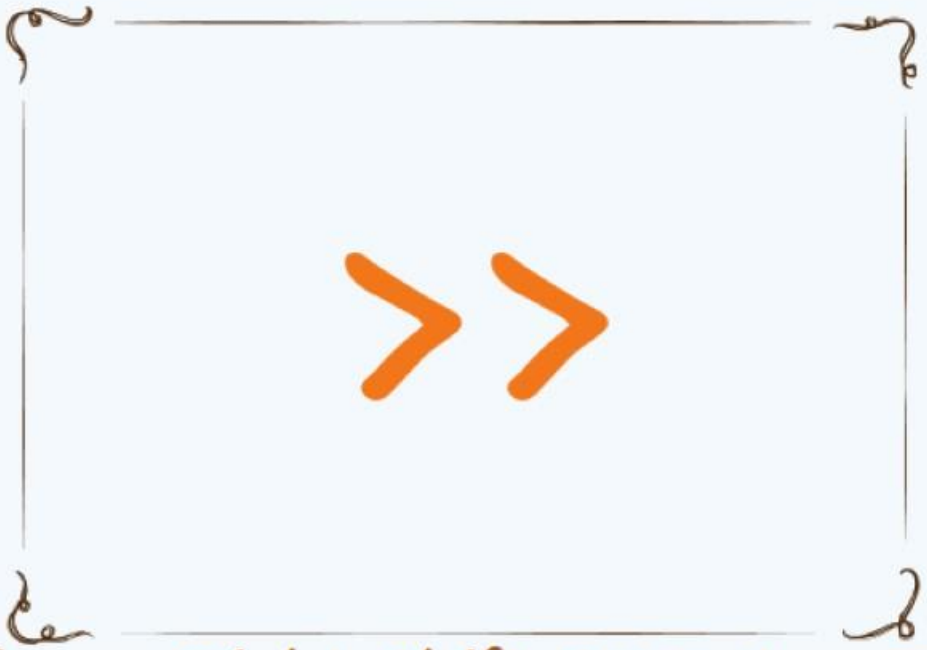
Output: 10

Output: -10



Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.

a << b



Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.

a >> b

Example:

```
int a = 9;

int b = 2;

int c = a << b;
System.out.println(c);
```

Output: 36

```
9=1001
<<=1001
00
>>=0010
```

```
int a = 9;

int b = 2;

int c = a >>> b;
System.out.println(c);
```

Output: 2

>>>

Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.

a >>> b

```
int a = 9;
int b = 2;
int c = a >>> b;
System.out.println(c);
```

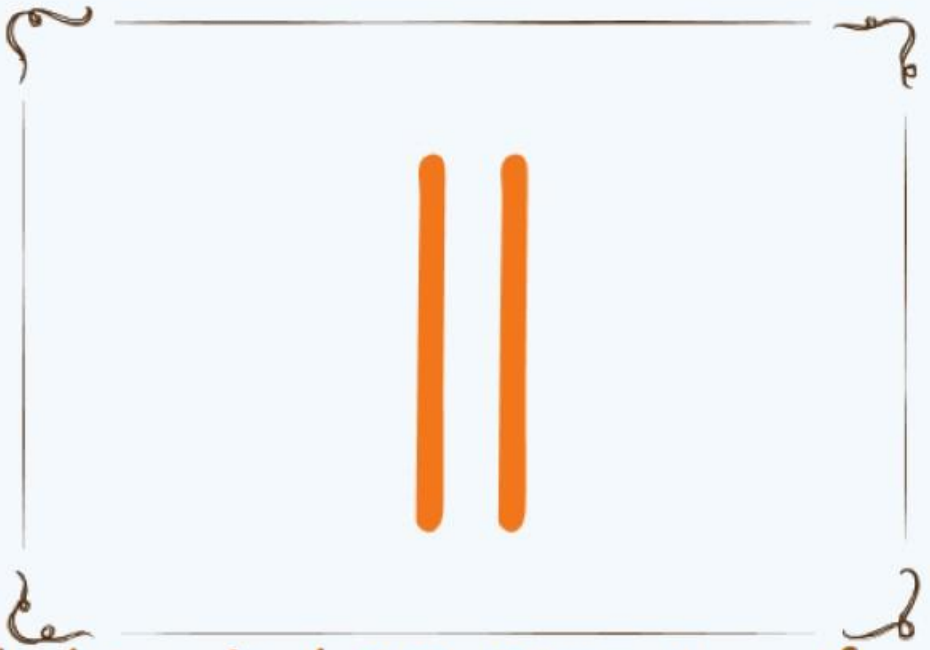
9=1001
>>>=001
0

Output: 2

Logical Operators

Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.

a && b



Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.

a || b

Example:

```
int a = 4;
int b = 2;
if (a == 4 && b == 2) {
System.out.println("Correct");
}
```

Output: Correct

```
int a = 4;
int b = 2;
if (a == 4 || b == 9) {
System.out.println("Correct");
}
```

Output: Correct

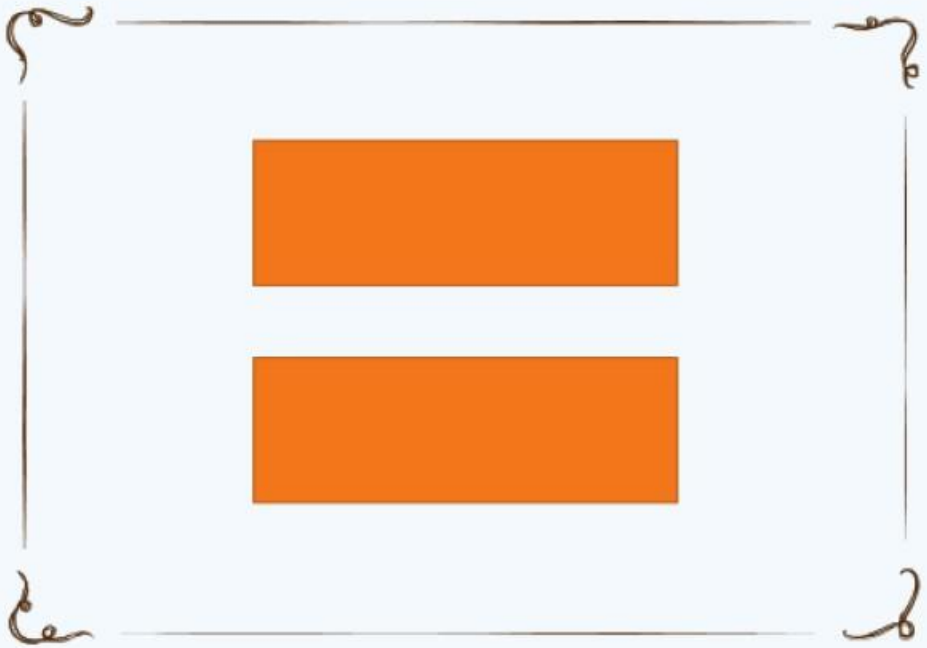
Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.

a != b

```
int a = 4;
int b = 2;
if (a != b) {
System.out.println("Not Match");
                    }
```

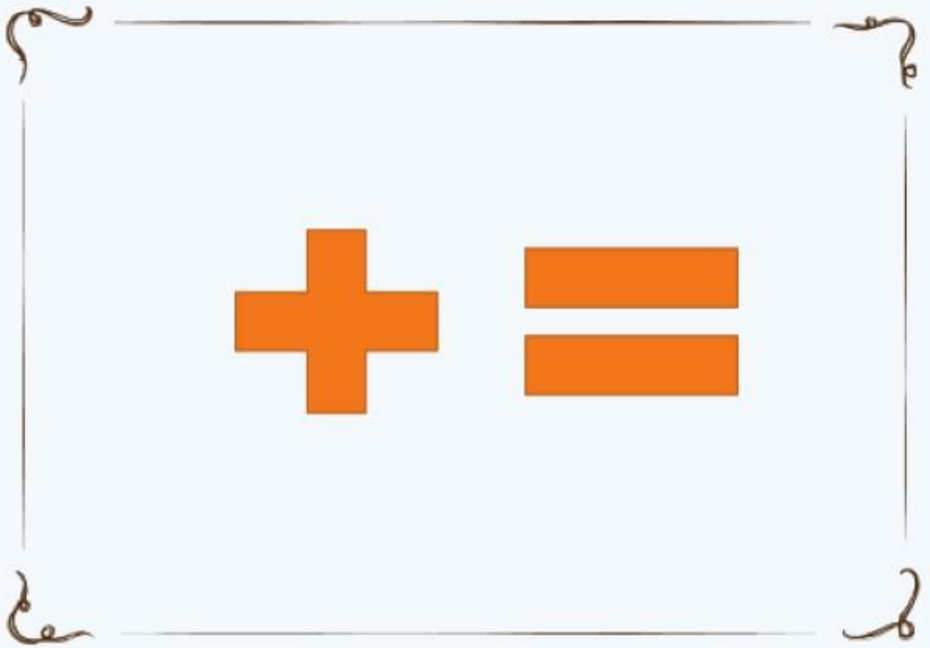
Output: Not Match

Assignment Operators



Assign Value

a = 9



a += b

a = a + b

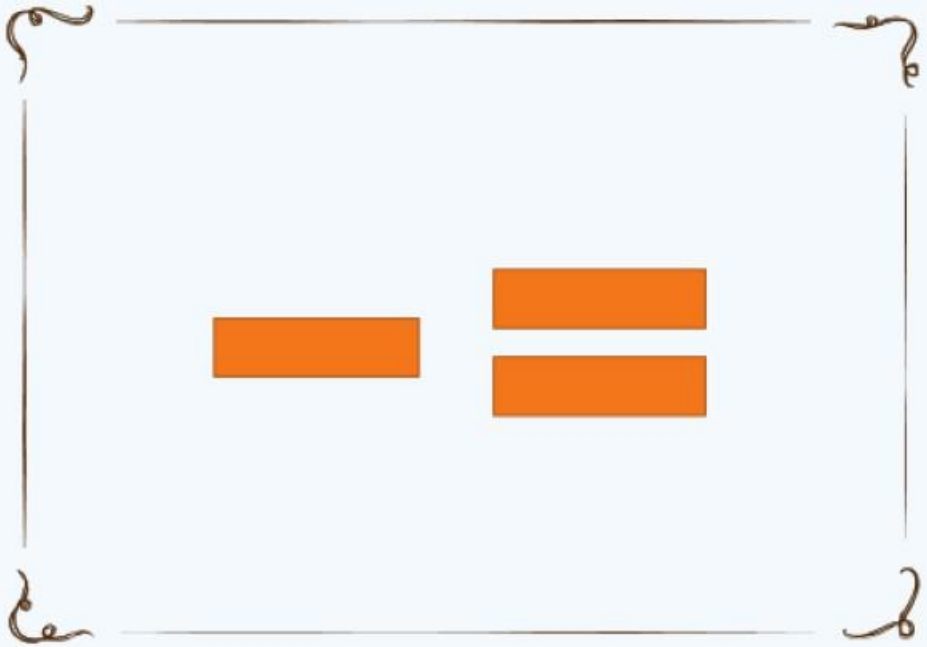
Example:

```
int a=9;
System.out.println(a);
```

Output: 9

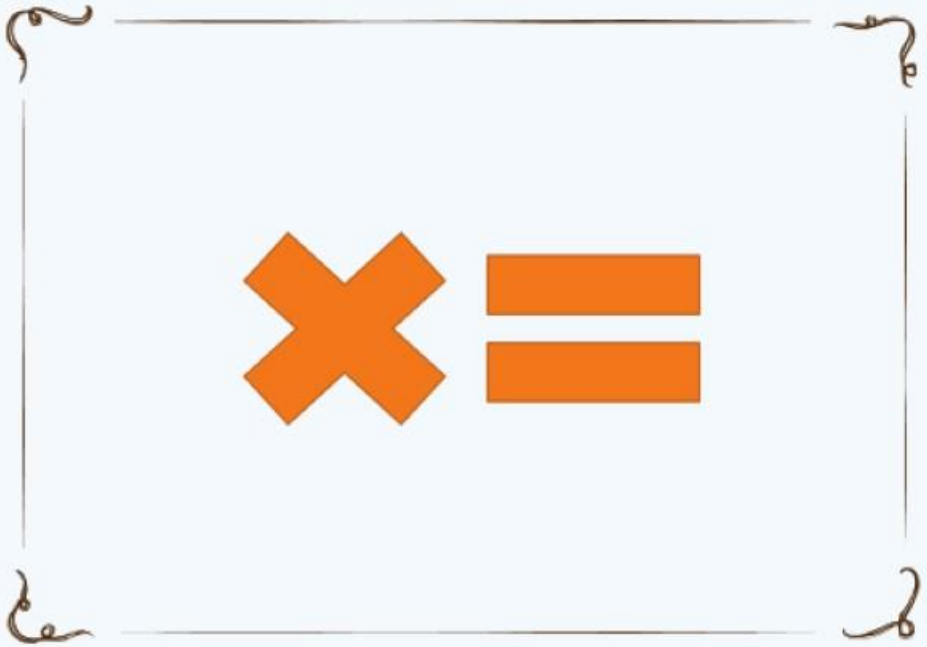
```
int a=9;
 int b=2;
 a += b;
System.out.println(a);
```

Output: 11



a -= b

a = a - b



a *= b

a = a * b

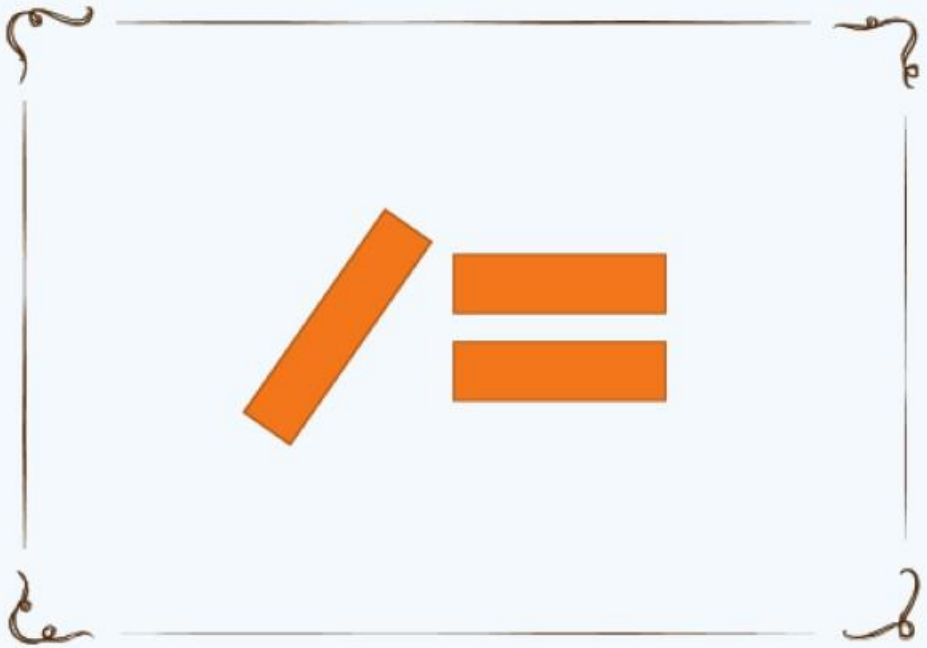
Example:

```
int a=9;

int b=2;

a -= b;
System.out.println(a);
```

Output: 7

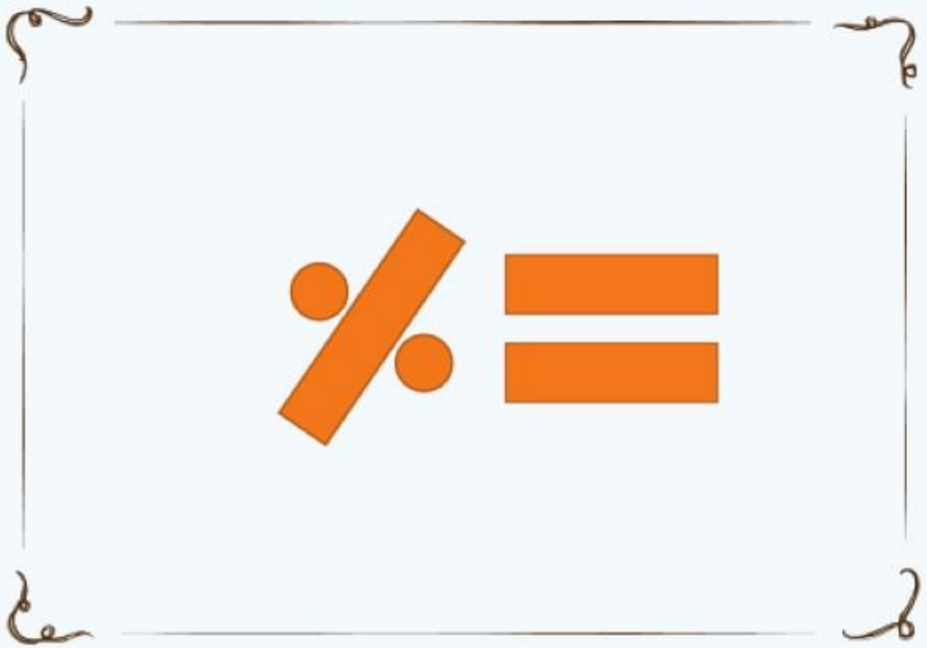
```
int a=9;
int b=2;
a *= b;
System.out.println(a);
```

Output: 18



a /= b

a = a / b



a %= b

a = a % b

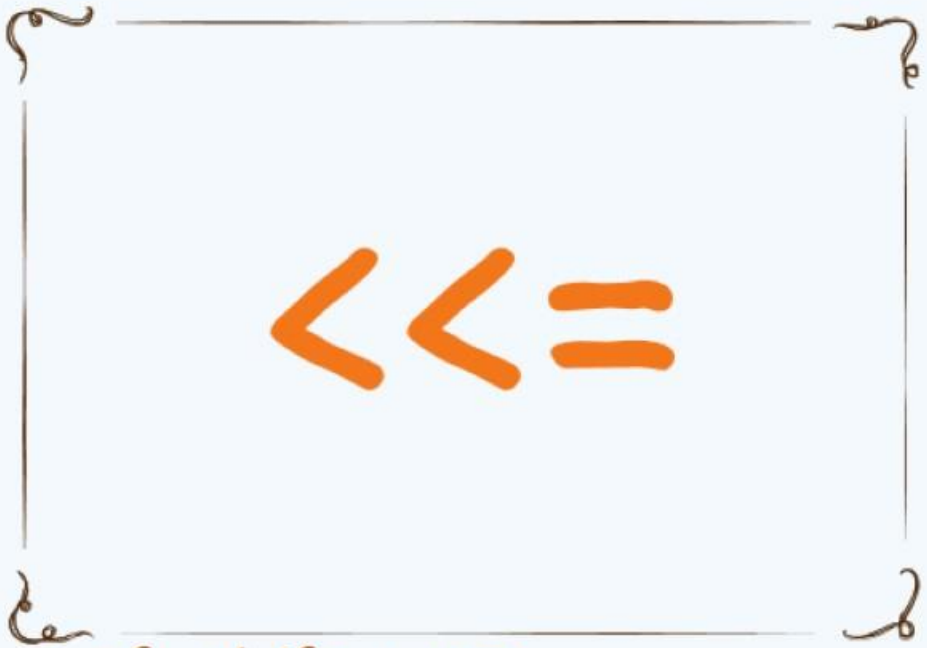
Example:

```
int a=9;
int b=2;
a /= b;
System.out.println(a);
```

Output: 4

```
int a=9;
int b=2;
a %= b;
System.out.println(a);
```

Output: 1



Left shift AND assignment operator.

a <<= 2

a = a << 2

Right shift AND assignment operator.

a >>= 2

a = a >> 2

Example:

```
int a = 9;
a <<= 2;
System.out.println(a);
```

Output: 36

```
int a = 9;
a >>= 2;
System.out.println(a);
```

Output: 2

&=

Bitwise AND assignment operator.

a &= 2

a = a & 2

^=

bitwise exclusive OR and assignment operator.

a ^= 2

a = a ^ 2

Example:

```
int a = 9;
a &= 2;
System.out.println(a);
```

Output: 0

```
int a = 9;
a ^= 2;
System.out.println(a);
```

Output: 11

bitwise inclusive OR and assignment operator.

a |= 2


```
int a = 9;
a |= 2;
System.out.println(a);
```

Output: 11