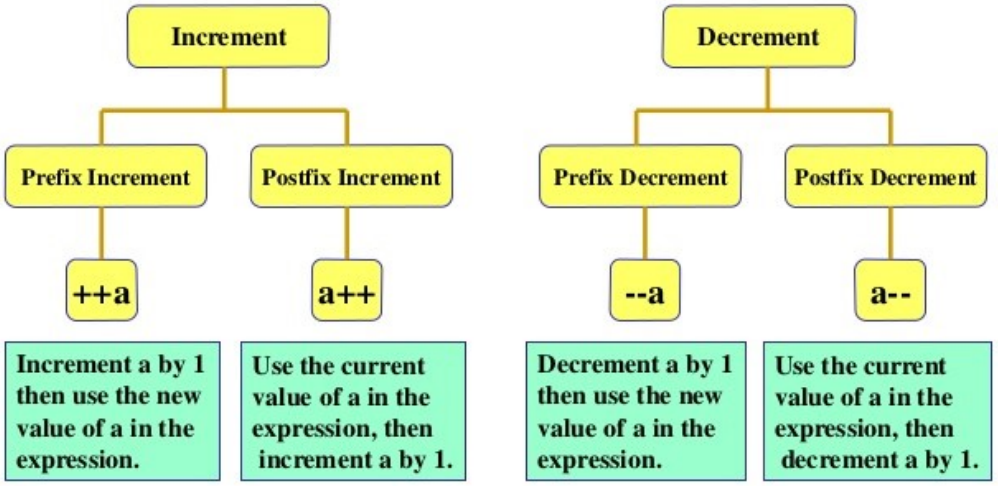
Operators

Operators	Symbols Used
Increment/decrement	"++,--"
Arithmatic	"+,-,*,/,%"
String concatenation	"+"
Relational	"<,<=,>,>="
Equality	"==,!="
BitWise	"&,|,^"
Short Circut	"&&,||"
Conditional	"? :"

Increment and Decrement Operator



Expression	Initial value of a	value of b	final value of a
b = ++a	10	11	11
b = a++	10	10	11
b = --a	10	9	9
b = a--	10	10	9

24- WAP to demonstrate increment operators

```
public class IncrementOperators {
    public static void main(String[] args) {
        int a = 10;
        int b = a++;
        System.out.println("prefix increment");
        System.out.println("a :: " + a);
        System.out.println("b :: " + b);

        int x = 10;
        int y = ++x;
        System.out.println("postfix increment");
        System.out.println("x :: " + x);
        System.out.println("y :: " + y);
    }
}
```

Output:

prefix increment
a :: 11
b :: 10
postfix increment
x :: 11
y :: 11

25- WAP to demonstrate decrement operators

```
public class DecrementOperators {
    public static void main(String[] args) {
        int a = 10;
        int b = a--;
        System.out.println("prefix decrement");
        System.out.println("a :: " + a);
        System.out.println("b :: " + b);

        int x = 10;
        int y = --x;
        System.out.println("postfix decrement");
        System.out.println("x :: " + x);
        System.out.println("y :: " + y);
    }
}
```

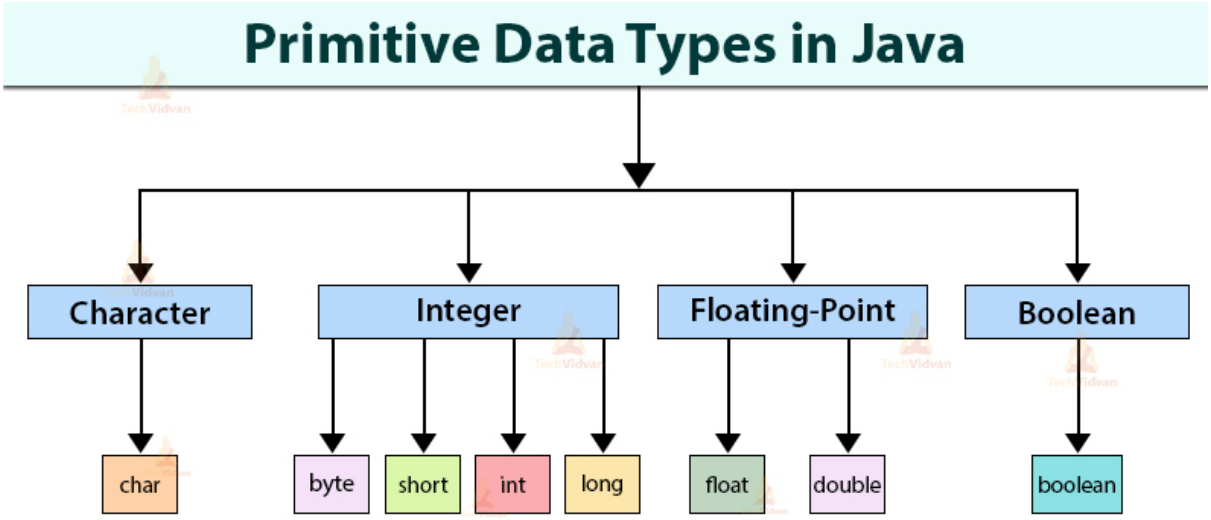
Output:

prefix decrement
a :: 9
b :: 10
postfix decrement
x :: 9
y :: 9

Arithmetic Operators:

Operator	Description	Example
+ Addition	Adds values on either side of the operator	A+B=30
– Subtraction	Subtracts the right-hand operator with left-hand operator	A-B=-10
* Multiplication	Multiplies values on either side of the operator	A*B=200
/ Division	Divides left hand operand with right hand operator	A/B=0
% Modulus	Divides left hand operand by right hand operand and returns remainder	A%B=0

Before we go into arithmetic operators, lets understand the primitive data types in java:



Below is the memory consumed by each data type:

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

Let's discuss the rules of the above in coming lectures.

String concatenation operator:

1. '+' is the only overloaded operator in java.
2. It is used for both arithmetic operations and string concatenation operations.

Rule:

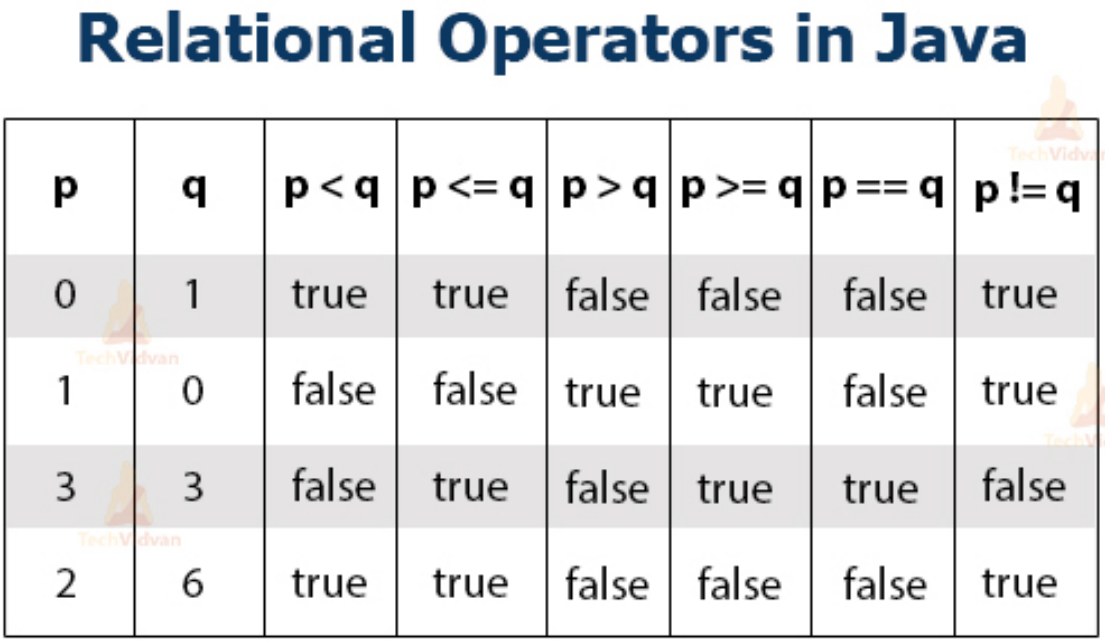
1. If one of the arguments is String, it acts as concatenation .
2. If both are numbers, then it acts as an arithmetic operator.

```
public class StringConcOperator {
    public static void main(String[] args) {
        int a = 10, b = 20, c = 30;
        int d = a + b + c;
        System.out.println(d);
        System.out.println("------");
        String x = "java";
        int y = 10, z = 20;
        System.out.println(x);
        System.out.println(y);
        System.out.println(z);
        System.out.println(y + z);
        System.out.println(x + y);
        System.out.println(x + y + z);
        System.out.println(y + z + x);
    }
}
```

Output:

60
---
java
10
20
30
java10
java1020
30java

Relational Operators:




```
public class RelationalOperators {
    public static void main(String[] args) {
        int p = 2;
        int q = 6;
        int r = 6;
        System.out.println(p < q);
        System.out.println(p <= r);
        System.out.println(p > q);
        System.out.println(p >= r);
        System.out.println(p == r);
        System.out.println(p != q);
    }
}
```

Output:

true
true
false
false
false
true

Equality Operator: (== , !=)

This operators can be applied to both primitives and object types in java

```
public class EqualityOperators {
   public static void main(String[] args) {
       int a = 20;
       int b = 30;
       int c = 30;
       System.out.println(a == b);
       System.out.println(b == c);
       System.out.println(a != c);
       System.out.println("=====");
       Thread t1=new Thread();
       Thread t2=new Thread();
       Thread t3 = t1;
       System.out.println(t1==t2);
       System.out.println(t1==t3);
   }
}
```

Output:

false
true
true
=====
false
true

Bitwise Operators: (&, |, ^)

Bitwise Operator Truth Table

X	Y	X&Y	X|Y	X^Y	~(X)
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Decimal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Operation	4&5	4|5	4^5
4	100	100	100
5	101	101	101
Result in binary	**100**	**101**	**001**
Result in decimal	**4**	**5**	**1**

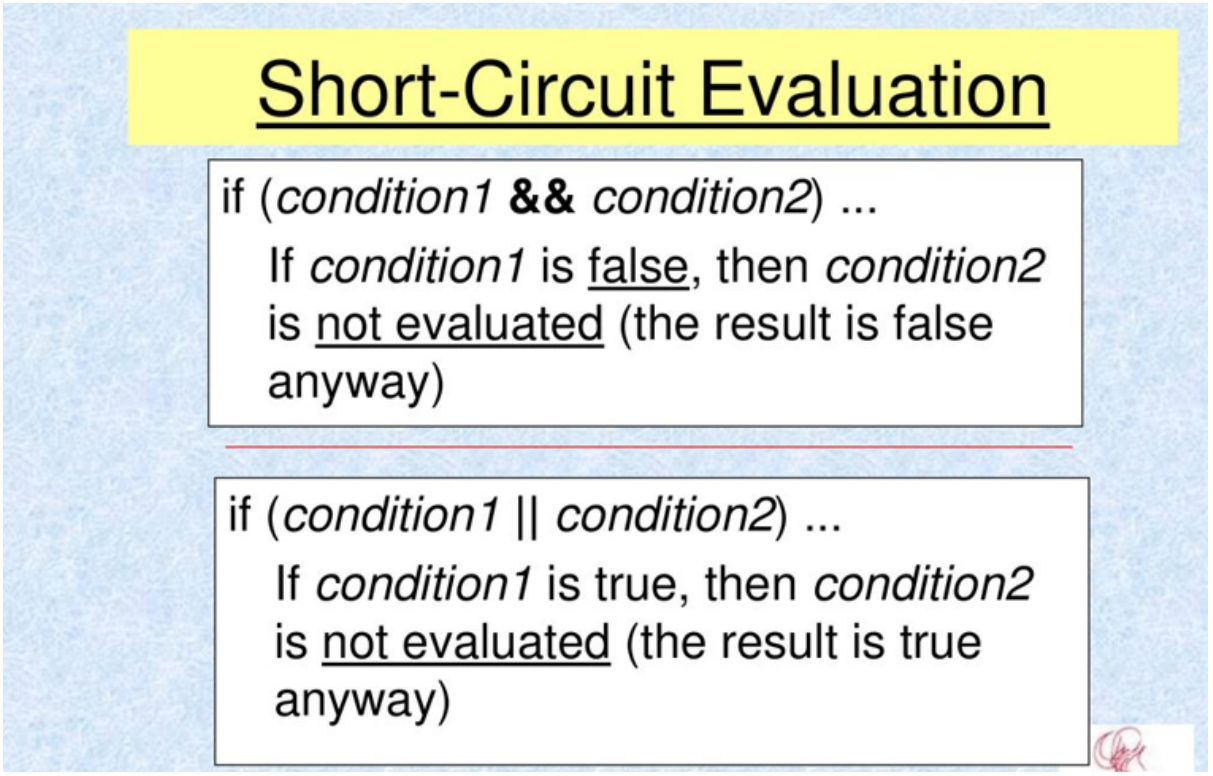
```
public class BitWiseOperators {
   public static void main(String[] args) {
        int x = 4;
        int y = 5;
        System.out.println(x & y);
        System.out.println(x | y);
        System.out.println(x ^ y);
   }
}
```

Output:

4
5
1

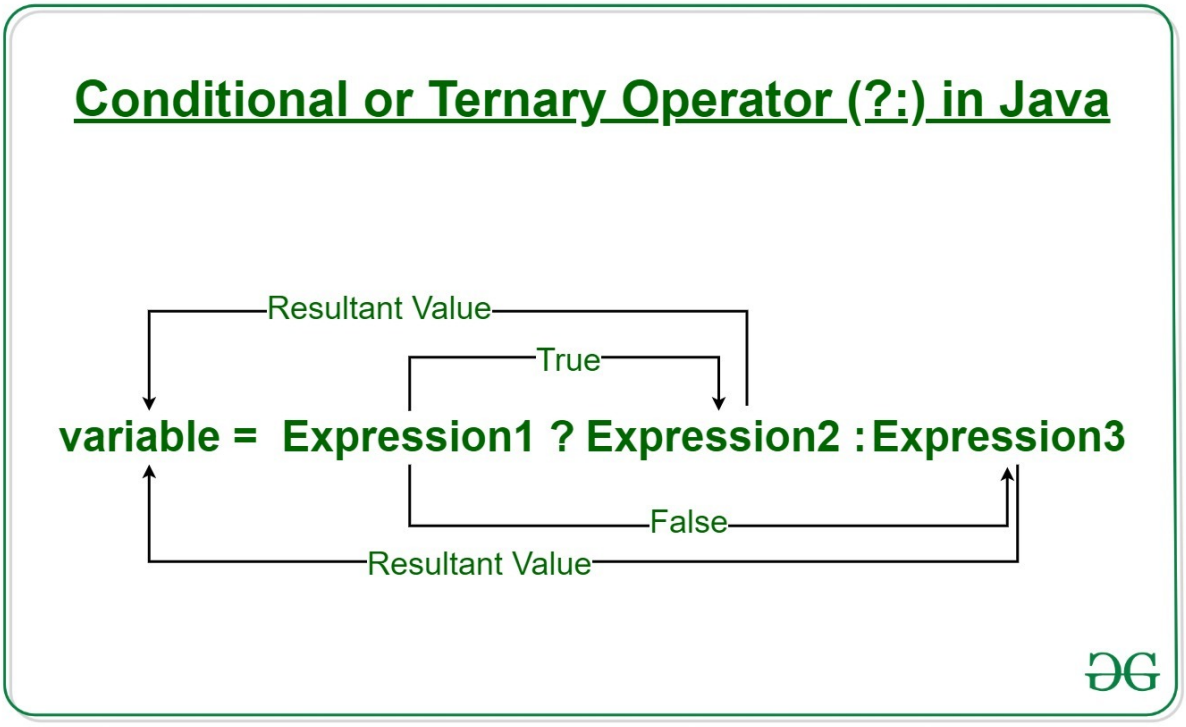
Short circuit operators(&& , ||)
Rules:

1. Only applicable for boolean.
2. Second argument evaluation is optional.




```
public class shortCircuitOperators {
   public static void main(String[] args) {
        int a = 10;
        int b = 20;
        if (a > 5 && b < 10) {
            System.out.println("Both conditions are true.");
        } else {
            System.out.println("At least one condition is false.");
        }
        System.out.println("=====");
        int x = 10;
        int y = 5;
        if (x > 5 || y > 10) {
            System.out.println("At least one condition is true.");
        } else {
            System.out.println("Both conditions are false.");
        }
   }
}
```

Conditional Operator:(? :)

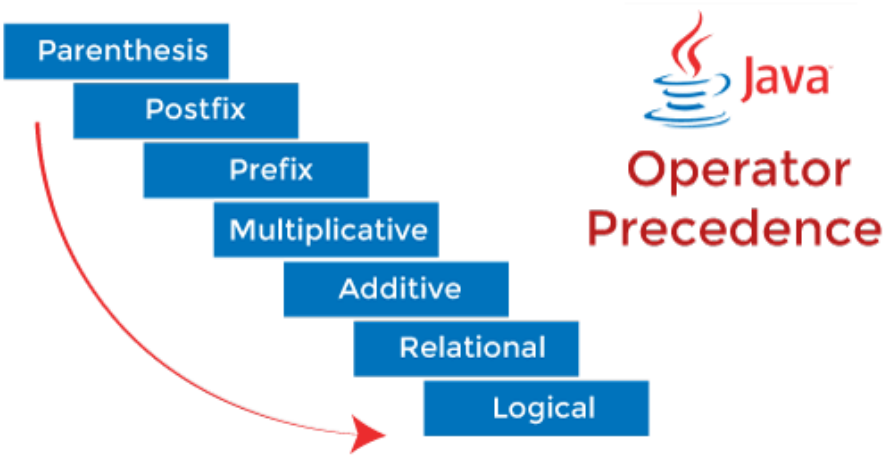



```
public class ConditionalOperator {
    public static void main(String[] args) {
        int marks = 35;
        int passMarks = 40;
        String result = "";
        if (marks >= passMarks) {
            result = "PASS";
        } else {
            result = "FAIL";
        }
        System.out.println(result);

        result = marks>=passMarks ? "PASS" : "FAIL";
        System.out.println(result);
    }
}
```

Operator **Precedence and Associativity**:

While solving an expression two things must be kept in mind the first is a precedence and the second is associativity.



Precedence	Operator	Type	Associativity
15	() [] .	Parentheses Array subscript Member selection	Left to Right
14	++ --	Unary post-increment Unary post-decrement	Right to left
13	++ -- + - ! ~ (type)	Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast	Right to left
12	* / %	Multiplication Division Modulus	Left to right

11	+ -	Addition Subtraction	Left to right
10	<< >> >>>	Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension	Left to right
9	< <= > >= instanceof	Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only)	Left to right
8	== !=	Relational is equal to Relational is not equal to	Left to right
7	&	Bitwise AND	Left to right
6	^	Bitwise exclusive OR	Left to right
5	|	Bitwise inclusive OR	Left to right
4	&&	Logical AND	Left to right
3	||	Logical OR	Left to right
2	? :	Ternary conditional	Right to left
1	= += -= *= /= %=	Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment	Right to left

Example:

1+5*3 = 16. It is not 18 because of precedence rules.