

OPERATORS AND CONTROL STATEMENTS IN JAVA

1. Operators :

Operators are used to perform operations on variables and values.

These include arithmetic, relational, logical, bitwise, assignment, and miscellaneous operators.

a) Arithmetic Operators:

These are used for basic math operations.

Operator	Example	Description
		·
+	a+6	Addition
_	а-ь	Substraction
*	a*b	Multiplication
/	a/b	Division
%	a%b	Modulus (remainder)



Example:

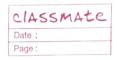
int
$$a = 10$$
, $b = 5$;
int $sum = a + b$; // $sum = 15$
int remainder = a % b // remainder = 0

b) Relational Operators:

These operators compare values to see if they are equal, greater, or smaller than each other.

Operator	Example	Description
==	a=b	Equal l o
!=	a!=b	Not equal to
>	аль	Gireater Than
<	a <b< th=""><th>Less than</th></b<>	Less than
>=	a >=b	Greater than or equal to
<=	a<=b	Less than or equal to

Example:



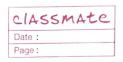
c) Logical Operators:

These are used when you want to check multiple conditions at once (for example, if two things are true).

Operator	Example	Description
8.8	a&& b	AND (both must be true)
11	allb	OR (one of them is true)
!	!a	NOT (inverts the condition)

<u>Example</u>:

Logical operators help combine conditions like checking if two conditions are true at the same time.



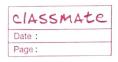
d) Bitwise Operators

These work with the binary form of numbers (bits) and perform operations on individual bits.

Operator	Example	Description
&	a&b	AND
	аЬ	OR
^	а^ь	XOR
~	~a	NOT
<<	a << 2	Left shift
>>	a >> 2	Right shift

Example :

These are used for low-level programming and dealing with individual bits of numbers



e) Assignment Operators:

These are used to assign a value to a variable.

Operator	Example	Description
	<u>a=b</u>	Assigns b to a
+=	a+=b	Adds b to a and stores the results in a
-=	a-=b	Subtracts b from a and stores the result in a
=	a=b	Multiplies a and b and stores result in a
/=	a/=b	Divides a by b and stores the result in a
º/o =	a%=b	Stores the remainder of a dividend by b in a

Example:

Assignment operators make it easy to store results in variables after doing operations.

f) Miscellaneous Operators:

These are special operators used for specific tasks.

Operator	Example	Description
ን:	a>b ?a:b	Ternary (shortend if-else)
sizeof	sizeof (a)	Gives the size of a yariable in memory

Example:

int
$$max = (a>b)$$
? $a:b$;

If $max = a$ if $a>b$, else $max=b$

2. Control Flow Statements:

Control flow statements help your program decide what to do based on conditions or repeat certain actions.

a) if statement:

The if statement runs a block of code only if a condition is true.

Example:

if (a > b) {
 printf ("a is greater than b");
}

It is used when you want to perform an action only if a condition is met.

b) if - else Statement:

The if-else statement lets you choose between two blocks of code: one if the condition is true, and one if it's Jalse.

Example:

```
if (a>b) {
    printf ("a is greater than b");
    } else {
    printf ("b is greater than a");
}
```

It is used when there are two parallel actions to take based on conditions

@ Curious - programmer

c) Switch Statement:

The switch statement is a cleaner way to handle multiple possible conditions by checking one variable against many options

It is used when you have many different options to check for, like menu choices.

3. Loops:

Loops are used to repeat a set of actions multiple times, like running a task over and over until a condition changes

a) for Loop:

A for loop is used when you know how many times you want to repeat an action.

Example:

It's used when you need to repeat something a fixed number of times, like looping through a list of items.

b) while Loop:

A while loop repeats an action and long as a condition is true. You might not know how many times it will repeat.

Examples:

```
int i = 0;
while (i < 5) {
    printf ("%d", i); // Prints 01234
    i++;
}</pre>
```

It's useful when you don't know how many repetitions you need, but just want to keep going until something changes.

c) do-while Loop:

A do-while loop is similar to a while loop, but it always runs at least once before checking the condition.

```
int i = 0;
    do {
        printf ("%d.", i);
        i++;
        while (i < 5);</pre>
```

It's used when you want to ensure the code runs at least once like showing a menu before checking the condition.

4. break and continue Statement:

a) break

The break statement stops the loop completely and moves on to the next part of the program.

Example:

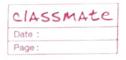
for (int i=0; i<10; i++) {

if (i==5) break; // Stops the loop when i is 5

Use break when you want to stop a loop before it finishes.

b) <u>Continue</u>:

The continue statement skips the current iteration of a loop and moves on the next one



Example:

```
for (int i = 0; i < 10; i++) {
    if (i==5) continue; // skips printing 5
    printf ("/d", i);
    // Prints 012346789
}
```

Use continue when you want to skip certain steps in a loop but keep the loop going.