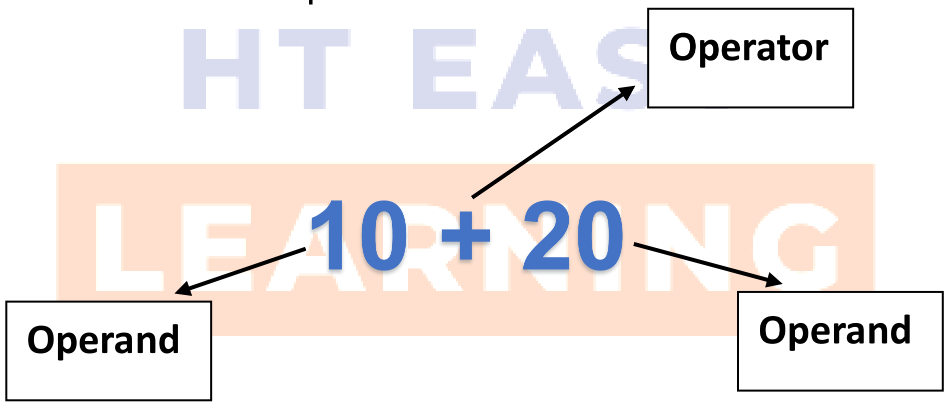
CORE JAVA

In the previous chapter we discussed variables and datatypes in detail. Now we are going to discuss about Java operators and keywords.

Java Operators

Operators are special symbols or keywords that perform operations on one or more operands (variables or values).

Operators are essential in Java programming as they help manipulate data, perform calculations, compare values, and control the flow of execution. Java provides a rich set of operators.



Example of operators are- +,-,*,=,<,>,<=,%,etc.

Types of Operators:

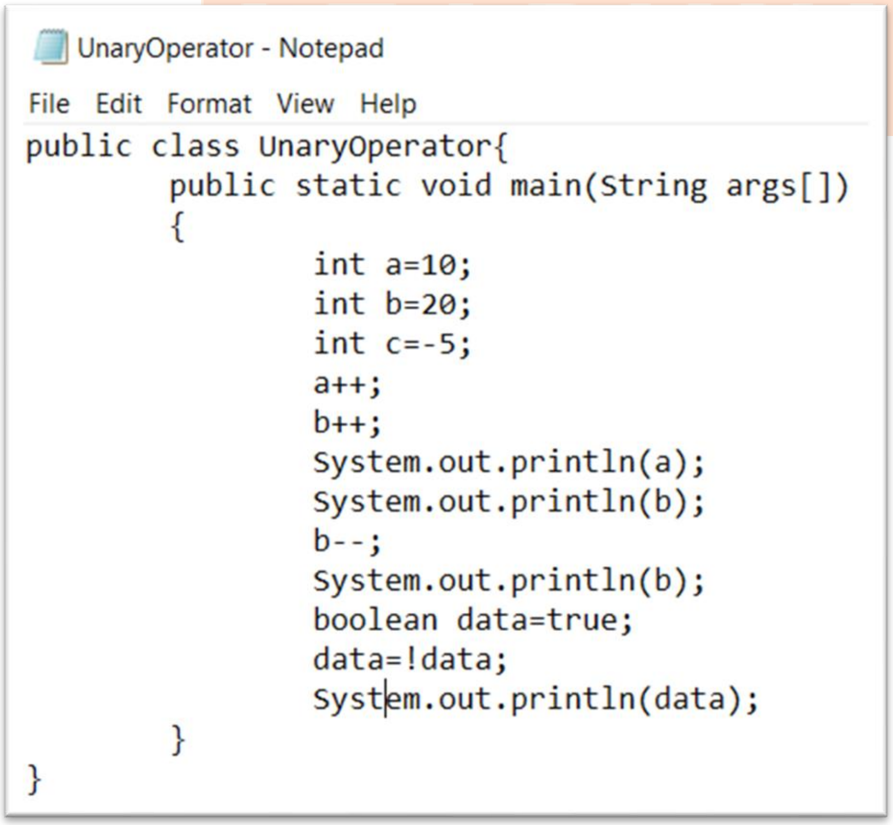
- Unary Operator
- Arithmetic operator
- Logical Operator
- Relational Operator
- Assignment Operator
- Bitwise Operator
- Ternary operator

1.Unary Operator-

These operators operate on a single operand and perform operations such as incrementing/decrementing a value, negating a value, or inverting a boolean value.

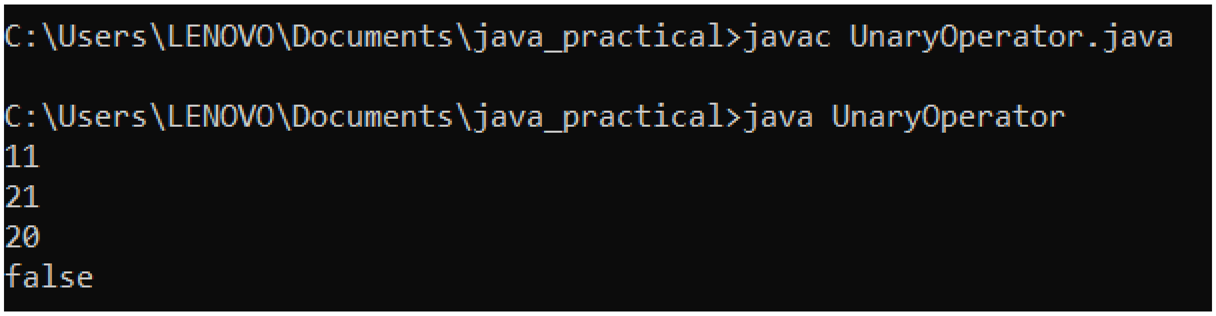
Unary operators- ‘-‘, ‘++’, ‘--‘, ‘!’

Example-



```
public class UnaryOperator{
        public static void main(String args[])
        {
                int a=10;
                int b=20;
                int c=-5;
                a++;
                b++;
                System.out.println(a);
                System.out.println(b);
                b--;
                System.out.println(b);
                boolean data=true;
                data=!data;
                System.out.println(data);
        }
}
```

Output:



```
C:\Users\LENOVO\Documents\java_practical>javac UnaryOperator.java

C:\Users\LENOVO\Documents\java_practical>java UnaryOperator
11
21
20
false
```

In Java, the increment operator '++' and decrement operator '--' can be used in two different forms: prefix and postfix. Both forms increment/decrement the value of the variable by 1, but they do so at different times relative to the expression they are used in. Here's a detailed explanation of the differences between prefix ++,-- and postfix ++,--:

Prefix Increment/Decrement-

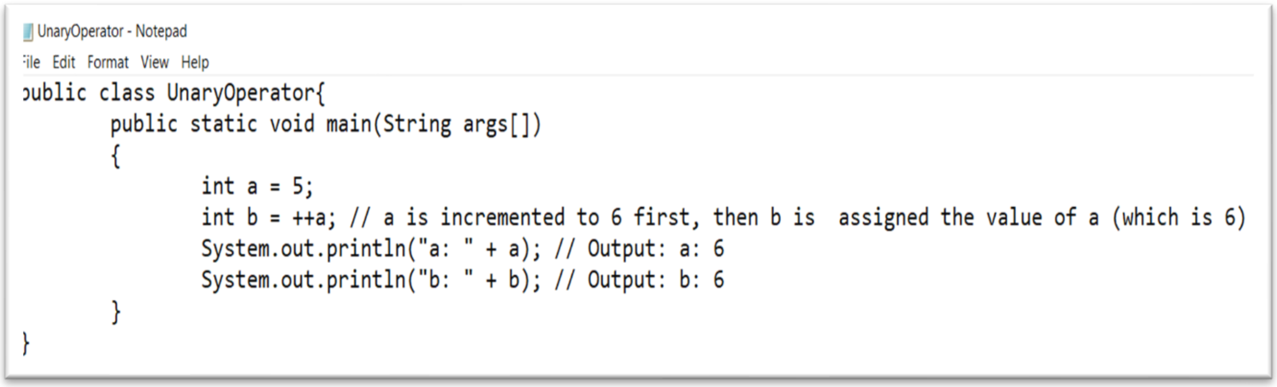
++variable;

--variable;

What happens in prefix increment/decrement?

- The value of the variable is incremented/decremented first.
- The incremented/decremented value is then used in the expression.

Let's understand this with an example-



```
public class UnaryOperator{
        public static void main(String args[])
        {
                int a = 5;
                int b = ++a; // a is incremented to 6 first, then b is  assigned the value of a (which is 6)
                System.out.println("a: " + a); // Output: a: 6
                System.out.println("b: " + b); // Output: b: 6
        }
}
```

Postfix Increment/decrement-

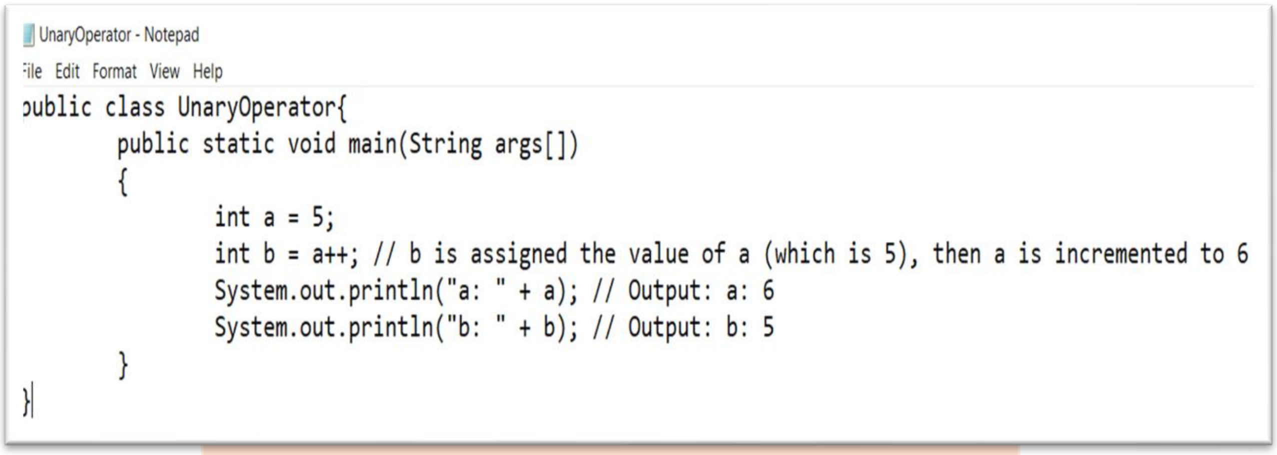
variable++;

variable--;

What happens in postfix increment/decrement?

- The current value of the variable is used in the expression.
- The value of the variable is then incremented.

Let’s understand this with an example-



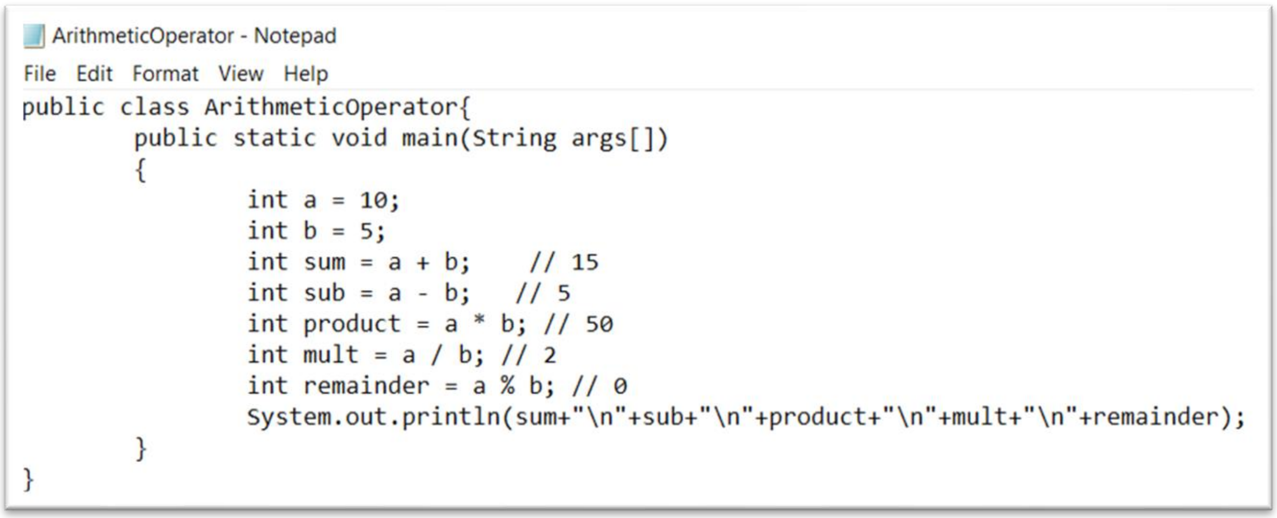
```
public class UnaryOperator{
        public static void main(String args[])
        {
                int a = 5;
                int b = a++; // b is assigned the value of a (which is 5), then a is incremented to 6
                System.out.println("a: " + a); // Output: a: 6
                System.out.println("b: " + b); // Output: b: 5
        }
}
```

2.Arithmetic Operator-

Arithmetic operator are used to perform arithmetic operations such as +,-,/,*,% between two or more operands.

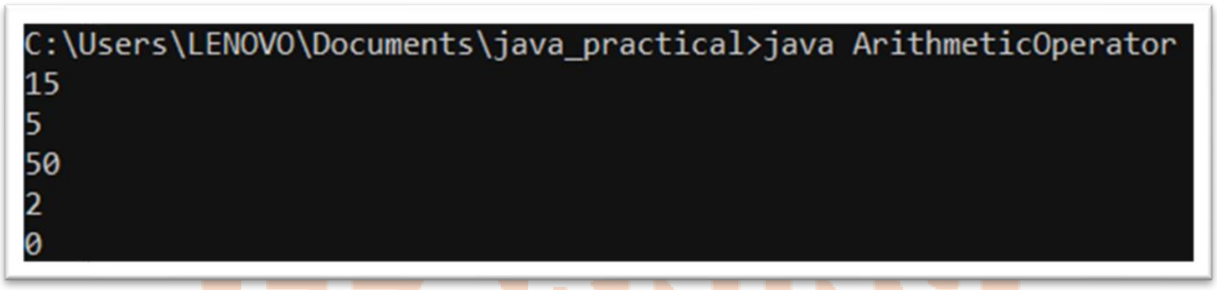
These operators perform basic arithmetic operations such as addition, subtraction, multiplication, division, and modulus.

Let’s see an example-



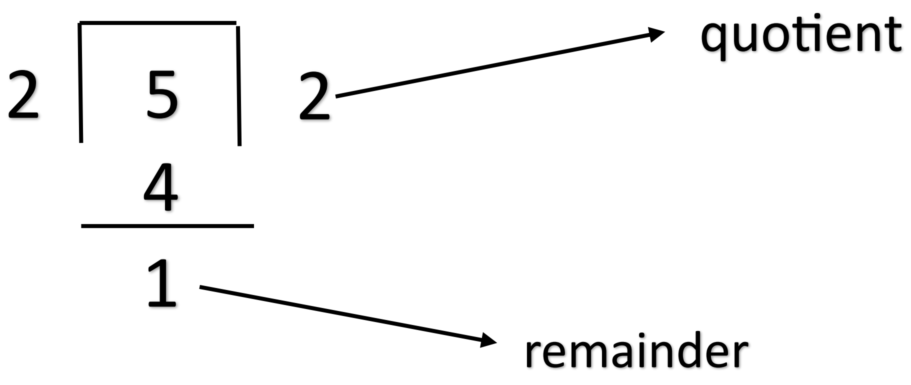
```
public class ArithmeticOperator{
        public static void main(String args[])
        {
                int a = 10;
                int b = 5;
                int sum = a + b;    // 15
                int sub = a - b;   // 5
                int product = a * b; // 50
                int mult = a / b; // 2
                int remainder = a % b; // 0
                System.out.println(sum+"\n"+sub+"\n"+product+"\n"+mult+"\n"+remainder);
        }
}
```

Output-



```
C:\Users\LENOVO\Documents\java_practical>java ArithmeticOperator
15
5
50
2
0
```

Modulus operator %- gives the remainder -



3.Relational Operator-

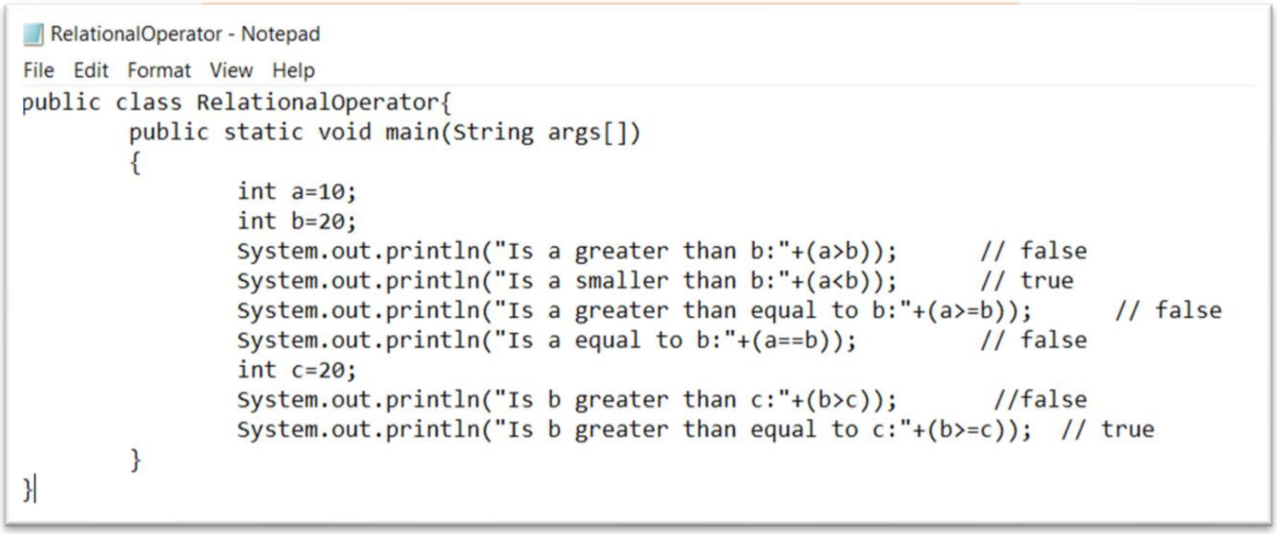
These operators compare two values and return a boolean result.

- **==** : Equal to
- **!=** : Not equal to
- **>** : Greater than
- **>=** : Greater than or equal to
- **<** : Less than
- **<=** : Less than or equal to

For example-

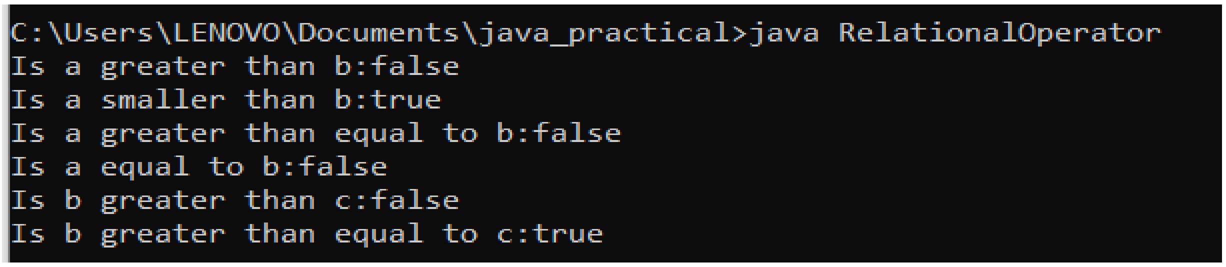
5>2 // True

5<2 // False


RelationalOperator - Notepad

File Edit Format View Help

```java
public class RelationalOperator{
        public static void main(String args[])
        {
                int a=10;
                int b=20;
                System.out.println("Is a greater than b:"+(a>b));      // false
                System.out.println("Is a smaller than b:"+(a<b));      // true
                System.out.println("Is a greater than equal to b:"+(a>=b));      // false
                System.out.println("Is a equal to b:"+(a==b));         // false
                int c=20;
                System.out.println("Is b greater than c:"+(b>c));       //false
                System.out.println("Is b greater than equal to c:"+(b>=c));  // true
        }
}
```

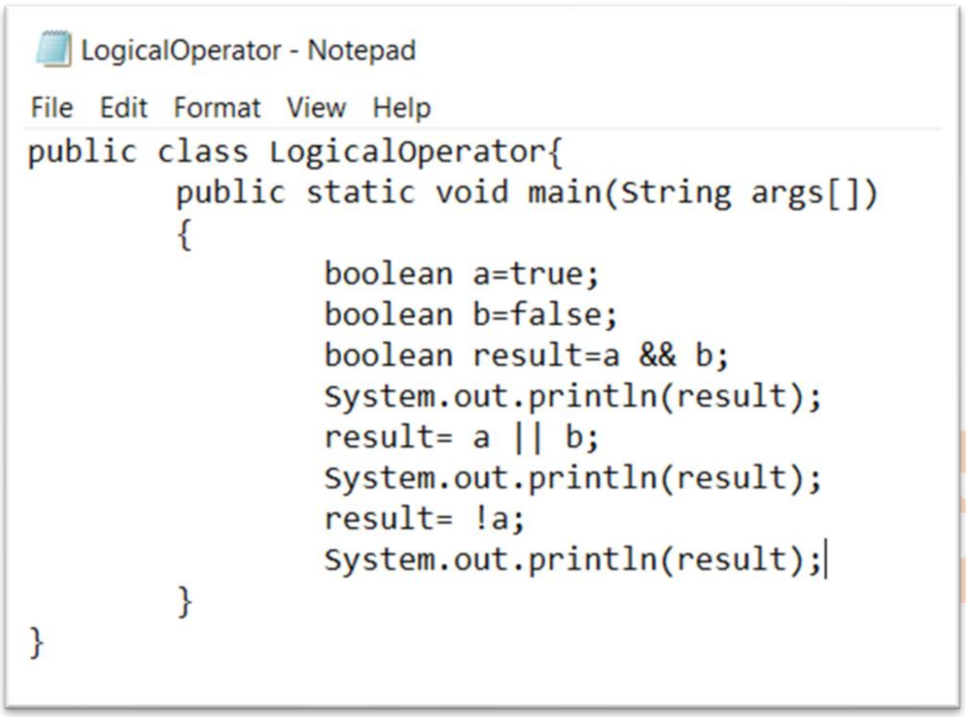

```
C:\Users\LENOVO\Documents\java_practical>java RelationalOperator
Is a greater than b:false
Is a smaller than b:true
Is a greater than equal to b:false
Is a equal to b:false
Is b greater than c:false
Is b greater than equal to c:true
```

4.Logical Operator-

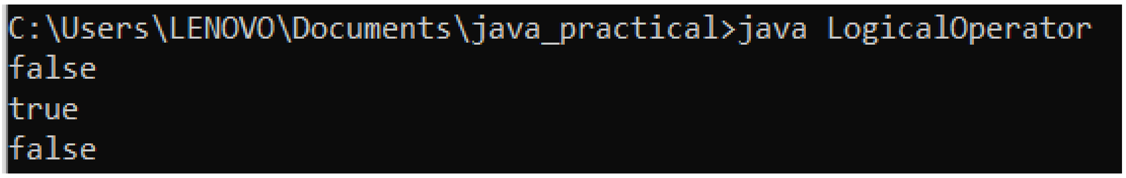
It checks if both the condition or any one condition is true or not. For this we use logical operator between two boolean expressions or conditions.

Logical Operators- && (AND), || (OR), ! (NOT)

Let's understand with an example-



```
public class LogicalOperator{
        public static void main(String args[])
        {
                boolean a=true;
                boolean b=false;
                boolean result=a && b;
                System.out.println(result);
                result= a || b;
                System.out.println(result);
                result= !a;
                System.out.println(result);
        }
}
```



```
C:\Users\LENOVO\Documents\java_practical>java LogicalOperator
false
true
false
```

Logical operators are used mainly with conditional statements like if statement, if-else statements. We will learn about conditional statement in the upcoming chapters.

5. **Bitwise Operators**

These operators perform bit-level operations on integer types.

- **&** : Bitwise AND
- **|** : Bitwise OR
- **^** : Bitwise XOR
- **~** : Bitwise complement
- **<<** : Left shift
- **>>** : Right shift
- **>>>** : Unsigned right shift

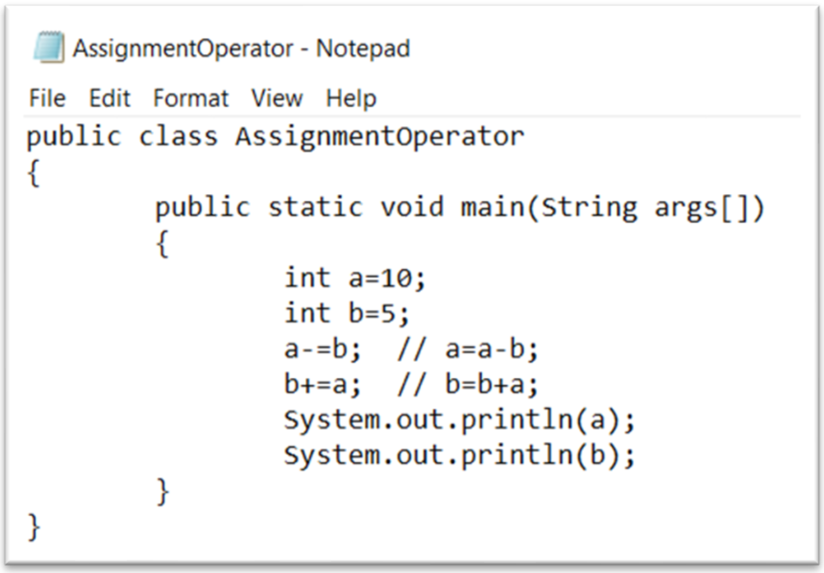
We will cover bitwise operator in detail in a separate chapter. In which we will also discuss how to convert decimal to binary and vice versa and logic gates and truth table.

6. Assignment Operators

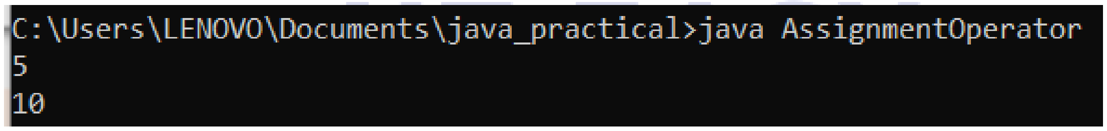
These operators are used to assign values to variables.

- **=** : Simple assignment
- **+=** : Add and assign
- **-=** : Subtract and assign
- ***=** : Multiply and assign
- **/=** : Divide and assign
- **%=** : Modulus and assign

Example-



```
public class AssignmentOperator
{
        public static void main(String args[])
        {
                int a=10;
                int b=5;
                a-=b;  // a=a-b;
                b+=a;  // b=b+a;
                System.out.println(a);
                System.out.println(b);
        }
}
```



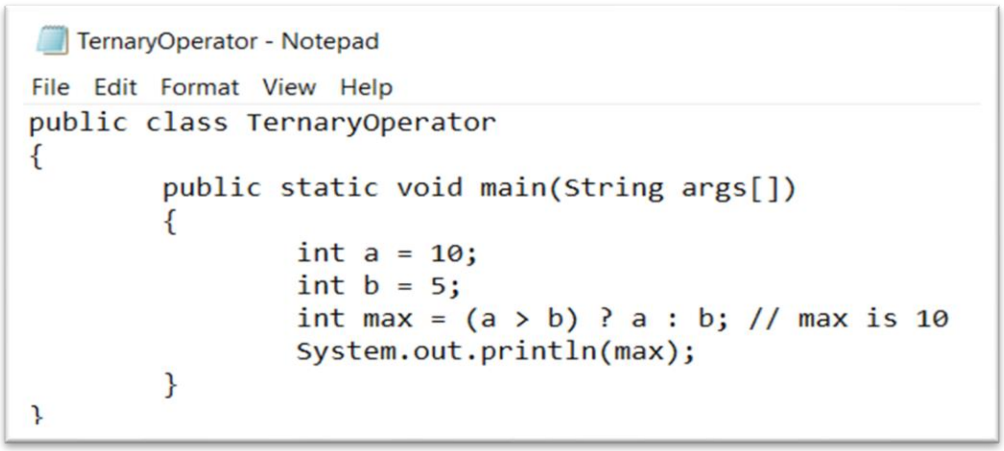
```
C:\Users\LENOVO\Documents\java_practical>java AssignmentOperator
5
10
```

7.Ternary Operator

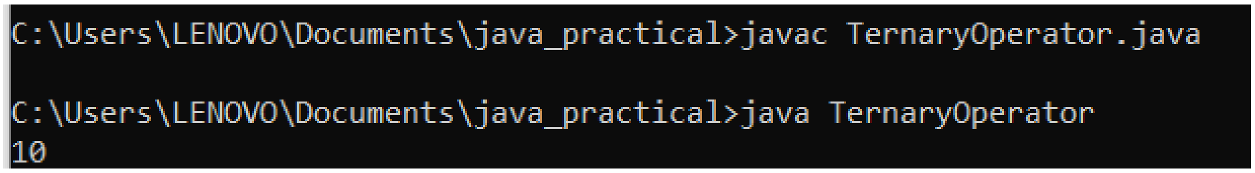
This is a shorthand for the if-else statement. It takes three operands and is used for making quick decisions.

? : : Ternary operator

Example-



```
public class TernaryOperator
{
        public static void main(String args[])
        {
                int a = 10;
                int b = 5;
                int max = (a > b) ? a : b; // max is 10
                System.out.println(max);
        }
}
```



```
C:\Users\LENOVO\Documents\java_practical>javac TernaryOperator.java

C:\Users\LENOVO\Documents\java_practical>java TernaryOperator
10
```

Operator Precedence-

Operator precedence in Java refers to the order in which operators are evaluated in expressions. Operators with higher precedence are evaluated before operators with lower precedence. If operators have the same precedence, their associativity (either left-to-right or right-to-left) determines the order of evaluation.

Operator Precedence

Operator Type	Category	Symbol(s)
Postfix	Increment	expr++
	Decrement	expr--
Unary	Increment	++expr
	Decrement	--expr
	Unary plus	+expr
	Unary minus	-expr
	Bitwise complement	~
	Logical NOT	!
Arithmetic	Multiplication	*
	Division	/
	Modulus	%
	Addition	+
	Subtraction	-
Shift	Left shift	<<
	Right shift	>>
	Unsigned right shift	>>>
Relational	Less than	<
	Less than or equal to	<=
	Greater than	>
	Greater than or equal to	>=

	instanceof	instanceof
Equality	Equal to	==
	Not equal to	!=
Bitwise	AND	&
	XOR	^
	OR	|
Logical	AND	&&
	OR	||
Ternary	Conditional	? :
Assignment	Assignment	=, +=, -=, *=, /=, %=,&=, ^=, `

Example-

int result = 10 - 4 + 3 * 2 / 1;

Ans- 12.

Here's the step-by-step evaluation based on precedence:

1. Multiplication and Division have the highest precedence among the operators in the expression.
2. Evaluate 3 * 2 first, resulting in 6.
3. Next, evaluate 6 / 1, resulting in 6.
4. Now the expression is reduced to 10 - 4 + 6.
5. Addition and Subtraction have the same precedence and are evaluated left to right (left associativity).
6. Evaluate 10 - 4 first, resulting in 6.
7. Finally, evaluate 6 + 6, resulting in 12.

So, the value of result is 12.

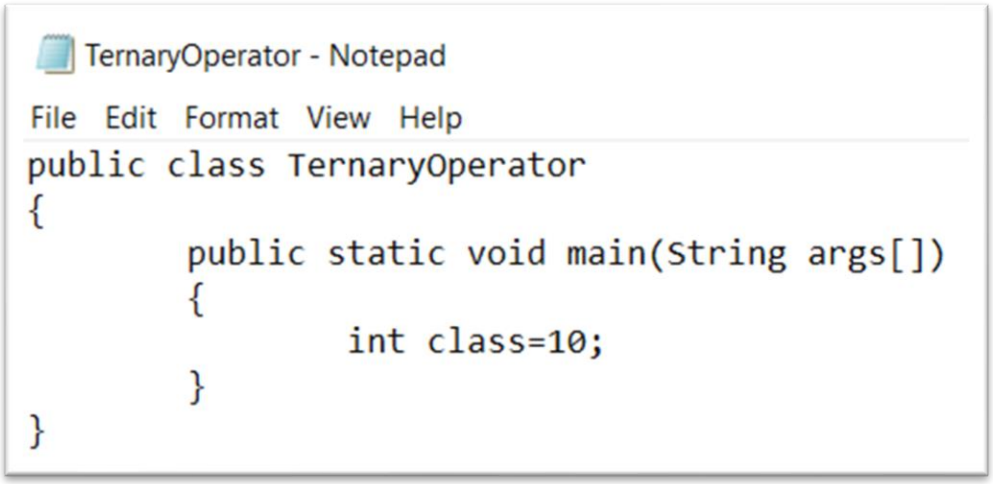
JAVA KEYWORDS

In Java, a keyword is a reserved word that has a predefined meaning in the language. Because they have specific purposes, they cannot be used as identifiers (such as variable names, function names, or class names).

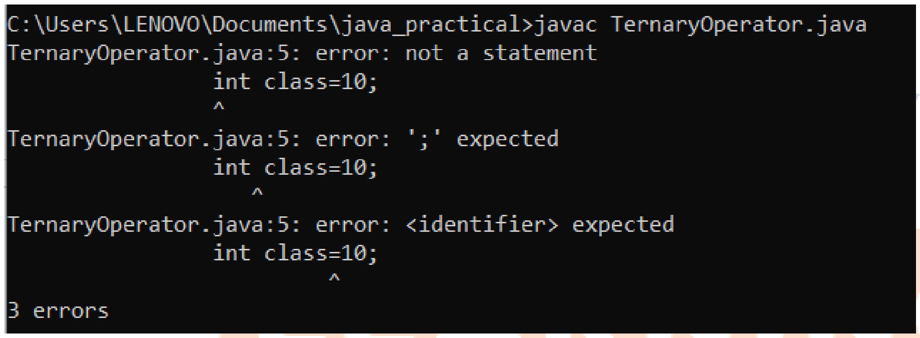
Here are some examples of keywords-

- abstract: Indicates that a class or method is abstract. An abstract class cannot be instantiated, and an abstract method must be implemented by subclasses.
- assert: Used for making assertions, which are a way to test assumptions in the code.
- boolean: Defines a boolean data type, which can hold only true or false.
- break: Exits from a loop or switch statement.
- byte: Defines a byte data type, which is an 8-bit signed integer.
- case: Defines a branch in a switch statement.
- catch: Used to handle exceptions that occur in a try block.
- char: Defines a character data type, which is a 16-bit Unicode character.
- class: Defines a class.
- const: Reserved keyword but not used. final is used instead.
- continue: Skips the current iteration of a loop and proceeds to the next iteration.
- default: Specifies the default block of code in a switch statement.
- do: Used in the do-while loop, which executes the block of code once before checking the condition.

What happens if we use a keyword as a variable name?



```java
public class TernaryOperator
{
        public static void main(String args[])
        {
                int class=10;
        }
}
```



```
C:\Users\LENOVO\Documents\java_practical>javac TernaryOperator.java
TernaryOperator.java:5: error: not a statement
                int class=10;
                ^
TernaryOperator.java:5: error: ';' expected
                int class=10;
                   ^
TernaryOperator.java:5: error: <identifier> expected
                int class=10;
                         ^
3 errors
```

If you use a keyword as a variable name in Java, the compiler will generate a syntax error. This is because keywords have predefined meanings and roles in the Java language syntax, and they cannot be used for any other purpose, such as naming variables, methods, classes, or other identifiers.

Join our growing community of tech enthusiasts! Subscribe to our YouTube channel, where we simplify programming languages in the easiest way possible. Gain a deeper understanding with our clear explanations and receive exclusive notes to enhance your learning journey. Don't miss out on valuable insights – hit that subscribe button now and embark on a programming adventure with us!

Subscribe to our channel:
https://www.youtube.com/@HTEASYLEARNING