**Java Operator**

Java Unary Operator

The Java unary operators require only one operand. Unary operators are used to perform various operations i.e.ox

incrementing/decrementing a value by one

negating an expression

inverting the value of a Boolean

Java Unary Operator Example: ++ and –

**public** **class** OperatorExample{

**public** **static** **void** main(String args[]){

**int** x=10;

System.out.println(x++);//10 (11)

System.out.println(++x);//12

System.out.println(x--);//12 (11)

System.out.println(--x);//10

}} 02

21

Java Unary Operator Example: !

**public** **class** OperatorExample{

**public** **static** **void** main(String args[]){

**boolean** c=**true**;

**boolean** d=**false**;

System.out.println(!c);//false (opposite of boolean value)

System.out.println(!d);//true

}}

-11

Java Arithmetic Operator Example

**public** **class** OperatorExample{

**public** **static** **void** main(String args[]){

**int** a=10;

**int** b=5;

System.out.println(a+b);//15

System.out.println(a-b);//5

System.out.println(a\*b);//50

System.out.println(a/b);//2

System.out.println(a%b);//0

}}

0

Java Arithmetic Operator Example: Expression

**public** **class** OperatorExample{

**public** **static** **void** main(String args[]){

System.out.println(10\*10/5+3-1\*4/2);

}}

**Output:**

**21**

**100/5+3-4/2**

**20+3-2**

**23-2**

**21**

21

Java Left Shift Operator

The Java left shift operator << is used to shift all of the bits in a value to the left side of a specified number of times.

Java Left Shift Operator Example

**public** **class** OperatorExample{

**public** **static** **void** main(String args[]){

System.out.println(10<<2);//10\*2^2=10\*4=40

System.out.println(10<<3);//10\*2^3=10\*8=80

System.out.println(20<<2);//20\*2^2=20\*4=80

System.out.println(15<<4);//15\*2^4=15\*16=240

}}  0

240

Java Right Shift Operator

The Java right shift operator >> is used to move the value of the left operand to right by the number of bits specified by the right operand.

Java Right Shift Operator Example

**public** OperatorExample{

**public** **static** **void** main(String args[]){

System.out.println(10>>2);//10/2^2=10/4=2

System.out.println(20>>2);//20/2^2=20/4=5

System.out.println(20>>3);//20/2^3=20/8=2

}}

11

Java Ternary Operator

Java Ternary operator is used as one line replacement for if-then-else statement and used a lot in Java programming. It is the only conditional operator which takes three operands.

Java Ternary Operator Example

**public** **class** OperatorExample{

**public** **static** **void** main(String args[]){

**int** a=2;

**int** b=5;

**int** min=(a<b)?a:b;

System.out.println(min);

}}

2

2

Another Example:

**public** **class** OperatorExample{

**public** **static** **void** main(String args[]){

**int** a=10;

**int** b=5;

**int** min=(a<b)?a:b;

System.out.println(min);

}}

55

Java Assignment Operator

Java assignment operator is one of the most common operators. It is used to assign the value on its right to the operand on its left.

Java Assignment Operator Example

**public** **class** OperatorExample{

**public** **static** **void** main(String args[]){

**int** a=10;

**int** b=20;

a+=4;//a=a+4 (a=10+4)

b-=4;//b=b-4 (b=20-4)

System.out.println(a);

System.out.println(b);

}}

14

16

Java Assignment Operator Example

**public** **class** OperatorExample{

**public** **static** **void** main(String[] args){

**int** a=10;

a+=3;//10+3

System.out.println(a);

a-=4;//13-4

System.out.println(a);

a\*=2;//9\*2

System.out.println(a);

a/=2;//18/2

System.out.println(a);

}}

13

9

public class Main {

public static void main(String[] args) {

int x = 5;

System.out.println(x > 3 && x < 10); // returns true because 5 is greater than 3 AND 5 is less than 10

}

}

public class Main {

public static void main(String[] args) {

int x = 5;

System.out.println(x > 3 || x < 4); // returns true because one of the conditions are true (5 is greater than 3, but 5 is not less than 4)

}

}

public class Main {

public static void main(String[] args) {

int x = 5;

System.out.println(!(x > 3 && x < 10)); // returns false because ! (not) is used to reverse the result

}

}

**1) Pre-increment operator**: A pre-increment operator is used to increment the value of a variable before using it in an expression. In the Pre-Increment, value is first incremented and then used inside the expression.

**Syntax:**

a = ++x;

Here, if the value of ‘x’ is 10 then the value of ‘a’ will be 11 because the value of ‘x’ gets modified before using it in the expression.

**2) Post-increment operator**: A post-increment operator is used to increment the value of the variable after executing the expression completely in which post-increment is used. In the Post-Increment, value is first used in an expression and then incremented.

**Syntax:**

a = x++;

Here, suppose the value of ‘x’ is 10 then the value of variable ‘a’ will be 10 because the old value of ‘x’ is used.

**int** a = 7;

**int** b = 2

b = ++a;

Use the pre-increment operator

The value of a is 7

The value of b is 2

After using the pre-increment operator

The value of a is 8

The value of b is 10

    a = 5;

    b = 7;

    c = 12;

    d = 15;

 x = ++a + ++b + ++c + ++d

;

The value of x is: 43

The updated value of a = 6, b = 8, c = 13 and d = 16

// declaration of the variables

**int** a = 7;

**int** b = 0;

b = a++;

Before using the post-increment operator

The value of a is 7

The value of b is 0

After using the post-increment operator

The value of a is 8

The value of b is 7

a = 5;

    b = 7;

    c = 12;

    d = 15;

    // use post-increment operator in the mathematical expression

    x = a++ + b++ + c++ + d++;

The value of x is: 39

The updated value of a = 6, b = 8, c = 13 and d = 16

# **Difference between next() and nextLine() methods in Java**

The scanner class consists **next()** and **nextLine()** methods. 

**next() Method:** The *next()* method in java is present in the [Scanner class](https://www.geeksforgeeks.org/scanner-class-in-java/) and is used to get the input from the user. In order to use this method, a Scanner [object](https://www.geeksforgeeks.org/classes-objects-java/) needs to be created. This method can read the input only until a space(” “) is encountered. In other words, it finds and returns the next complete token from the scanner.

The following is an example of how the next() method is implemented in Java:

Java

|  |
| --- |
| // Java program to demonstrate  // the next() method    **import** java.util.Scanner;    **class** next1 {  **public** **static** **void** main(String[] args)      {          // Creating the Scanner object          Scanner sc = **new** Scanner(System.in);            // Use of the next() method          String Input = sc.next();          System.out.println(Input);      }  } |

**Input:**

Khyati solanki

**123**

**Output:**

khyati

[**nextLine() Method**](https://www.geeksforgeeks.org/scanner-nextline-method-in-java-with-examples/)**:** The *nextLine()* method in java is present in the [Scanner class](https://www.geeksforgeeks.org/scanner-class-in-java/) and is used to get the input from the user. In order to use this method, a Scanner [object](https://www.geeksforgeeks.org/classes-objects-java/) needs to be created. This method can read the input till the end of line. In other words, it can take input until the line change or new line and ends input of getting ‘\n’ or press enter. 

The following is an example of how the nextLine() method is implemented in Java:

Java

|  |
| --- |
| // Java program to demonstrate  // the nextLine() method    **import** java.util.Scanner;    **class** next1 {  **public** **static** **void** main(String[] args)      {          // Creating the object of the          // Scanner class          Scanner sc = **new** Scanner(System.in);            // Use of nextLine() method          String Input = sc.nextLine();          System.out.println(Input);      }  } |

**Input:**

**Input:**

Khyati solanki

**123**

**Output:**

khyati

| Next() | NextLine() |
| --- | --- |
| It read input from the input device till the space character. | It read input from the input device till the line change. |
| It cannot read those words having space in it. | It can read those words having space in it. |
| It ends reading the input after getting space. | It ends reading the input after getting ‘\n’ or press enter. |
| It places the cursor in the same line after reading the input. | It places the cursor in the next line after reading the input. |
| The escaping sequence of next() is space. | The escaping sequence of nextLine() is ‘\n’. |