# OPERATORS IN JAVA

Velocity

JULY 25, 2022

VELOCITY

Version : 1.1

# Operators in Java

"Operators are symbols that perform operations on variables and values."

Java provides many types of operators which can be used according to the need. They are classified based on the functionality they provide. Some of the types are:

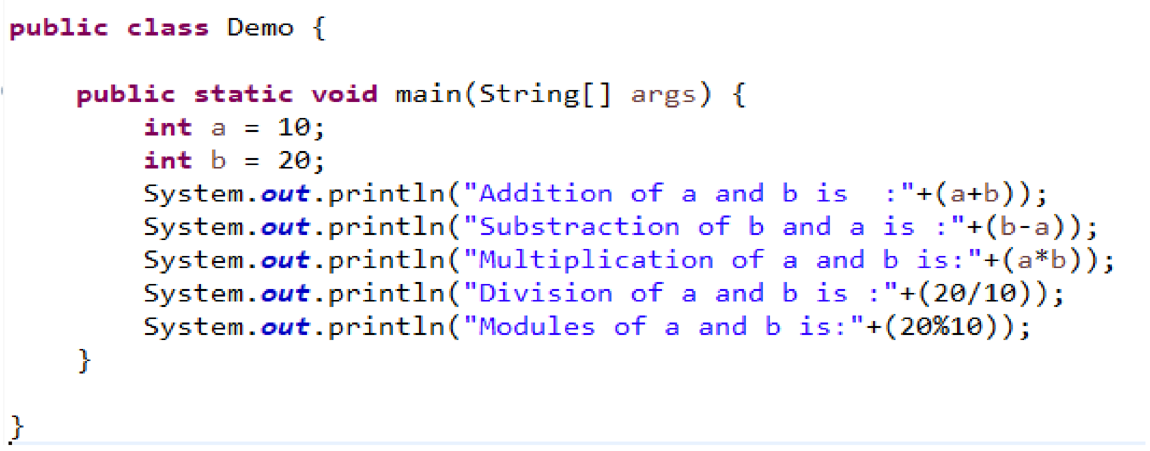
- Arithmetic operators
- Logical operators
- Relational operators
- Assignment operators
- Bitwise operators
- Unary operators
- Ternary operators
- Shift operators
- New operator
- . operator
- Instanceof operator

Let's look at them in details

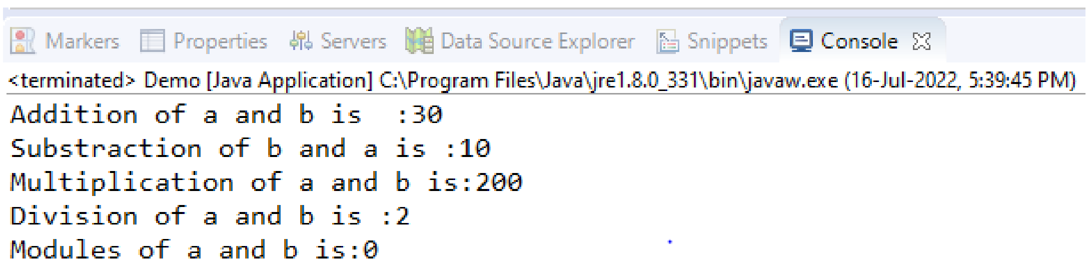
1. Arithmetic operators: They are used to perform simple arithmetic operations on primitive data types.

- *** :** Multiplication
- **/ :** Division
- **% :** Modulo
- **+ :** Addition
- **– :** Subtraction

**Example**



```
public class Demo {

    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        System.out.println("Addition of a and b is  :"+(a+b));
        System.out.println("Substraction of b and a is :"+(b-a));
        System.out.println("Multiplication of a and b is:"+(a*b));
        System.out.println("Division of a and b is :"+(20/10));
        System.out.println("Modules of a and b is:"+(20%10));
    }

}
```



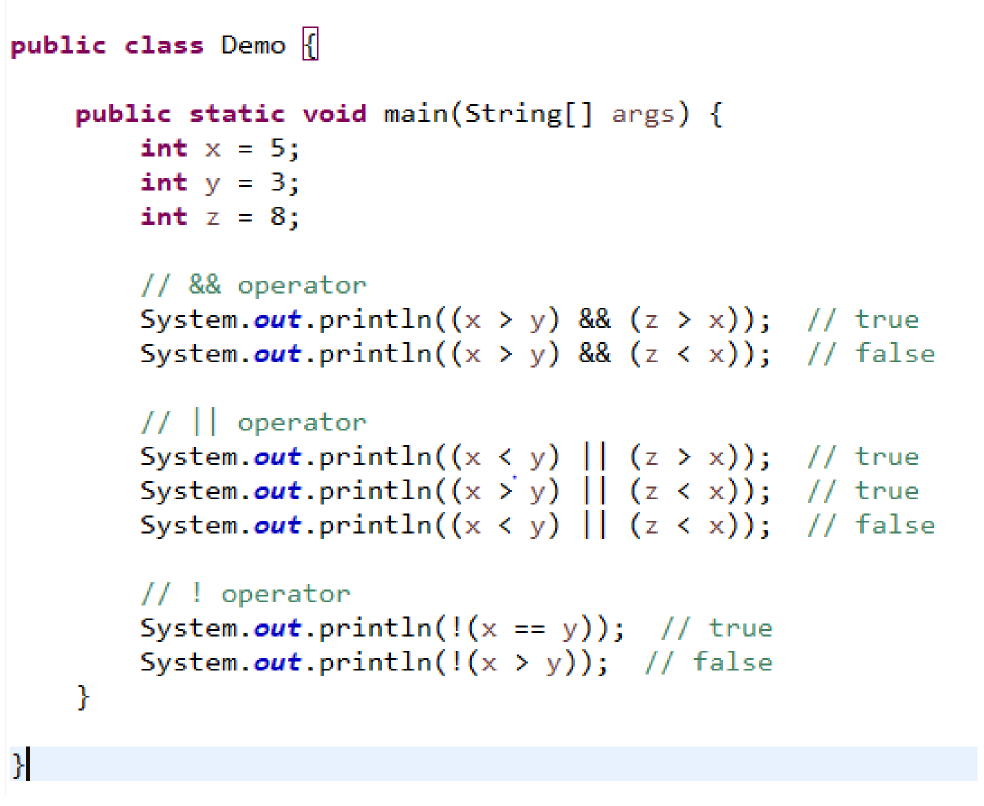
```
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_331\bin\javaw.exe (16-Jul-2022, 5:39:45 PM)
Addition of a and b is  :30
Substraction of b and a is :10
Multiplication of a and b is:200
Division of a and b is :2
Modules of a and b is:0
```

2. Logical Operators: These operators are used to perform “logical AND” and “logical OR” operations, i.e., a function similar to AND gate and OR gate in digital electronics.

-Logical operators are used to check whether an expression is true or false. They are used in decision making.

| Operator | Example | Meaning |
|---|---|---|
| && (Logical AND) | expression1 && expression2 | `true` only if both `expression1` and `expression2` are `true` |
| \|\| (Logical OR) | expression1 \|\| expression2 | `true` if either `expression1` or `expression2` is `true` |
| ! (Logical NOT) | ! expression | `true` if `expression` is `false` and vice versa |

Example:

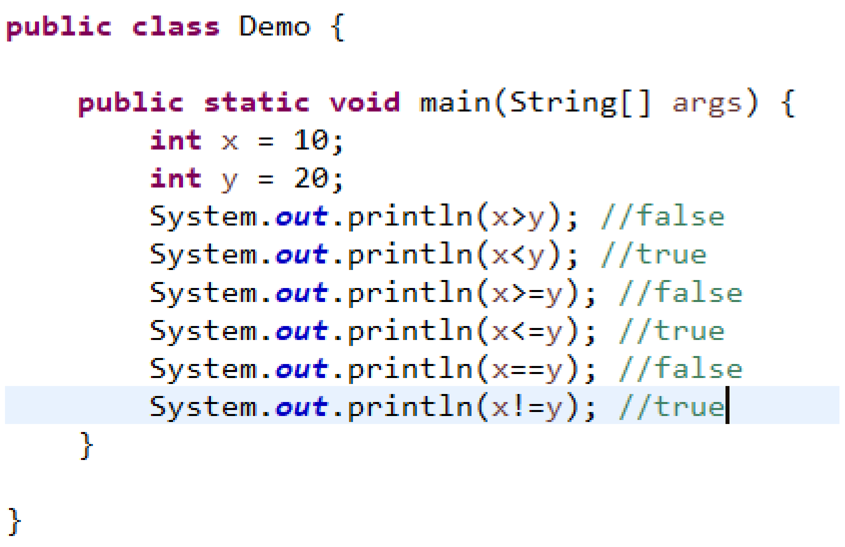


```java
public class Demo {

    public static void main(String[] args) {
        int x = 5;
        int y = 3;
        int z = 8;

        // && operator
        System.out.println((x > y) && (z > x));  // true
        System.out.println((x > y) && (z < x));  // false

        // || operator
        System.out.println((x < y) || (z > x));  // true
        System.out.println((x > y) || (z < x));  // true
        System.out.println((x < y) || (z < x));  // false

        // ! operator
        System.out.println(!(x == y));  // true
        System.out.println(!(x > y));  // false
    }

}
```

3. Relational Operators: These operators are used to check for relations like equality, greater than, and less than. They return Boolean results after the comparison and are extensively used in looping statements as well as conditional if-else statements.

| Operator | Description | Example |
|---|---|---|
| == | Is Equal To | 3 == 5 returns **false** |
| != | Not Equal To | 3! = 5 returns **true** |
| > | Greater Than | 3 > 5 returns **false** |
| < | Less Than | 3 < 5 returns **true** |
| >= | Greater Than or Equal To | 3 >= 5 returns **false** |
| <= | Less Than or Equal To | 3 <= 5 returns **true** |

Example

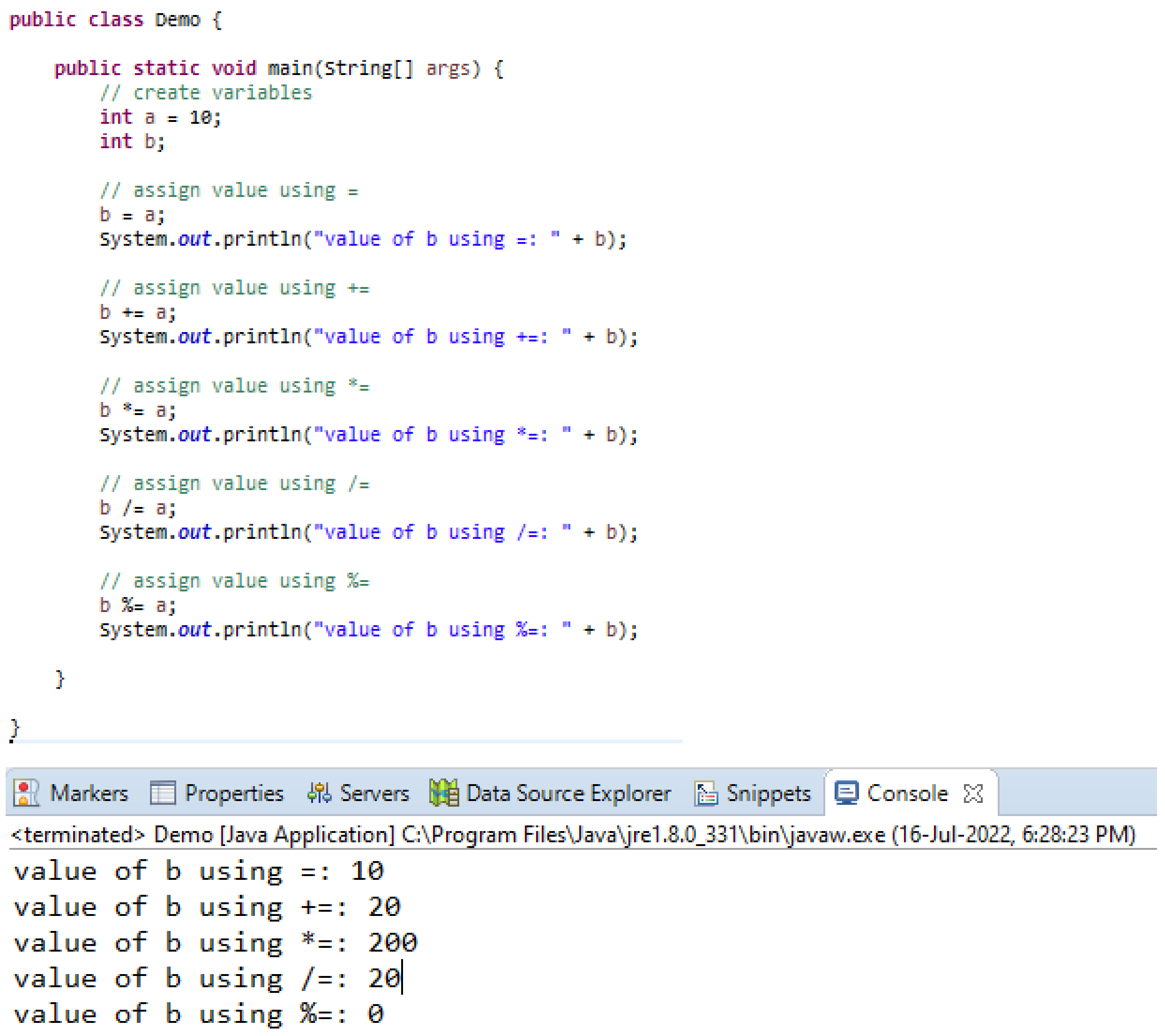


```
public class Demo {

    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        System.out.println(x>y); //false
        System.out.println(x<y); //true
        System.out.println(x>=y); //false
        System.out.println(x<=y); //true
        System.out.println(x==y); //false
        System.out.println(x!=y); //true
    }

}
```

4. Assignment Operators: Assignment operators are used in Java to assign values to variables.

| Operator | Example | Equivalent to |
|---|---|---|
| = | a = b; | a = b; |
| += | a += b; | a = a + b; |
| -= | a -= b; | a = a - b; |
| *= | a *= b; | a = a * b; |
| /= | a /= b; | a = a / b; |
| %= | a %= b; | a = a % b; |

Example :



```java
public class Demo {

    public static void main(String[] args) {
        // create variables
        int a = 10;
        int b;

        // assign value using =
        b = a;
        System.out.println("value of b using =: " + b);

        // assign value using +=
        b += a;
        System.out.println("value of b using +=: " + b);

        // assign value using *=
        b *= a;
        System.out.println("value of b using *=: " + b);

        // assign value using /=
        b /= a;
        System.out.println("value of b using /=: " + b);

        // assign value using %=
        b %= a;
        System.out.println("value of b using %=: " + b);

    }

}
```

<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_331\bin\javaw.exe (16-Jul-2022, 6:28:23 PM)

```
value of b using =: 10
value of b using +=: 20
value of b using *=: 200
value of b using /=: 20
value of b using %=: 0
```

5. Bitwise operators: This operator is used to perform bitwise AND & OR operation.

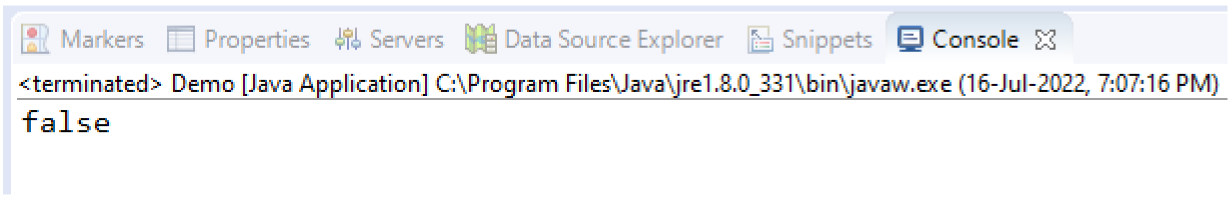
a) Bitwise AND (&): The bitwise & operator always checks both conditions whether first condition is true or false.

| Expression-1 | Expression-2 | Result |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

Example-1

```java
public class Demo {

    public static void main(String[] args) {

        int a = 10;
        int b = 20;
        int c = 30;

        System.out.println((a > b) & (a < c));

    }

}
```

In this example, first condition 10>20 become false and second condition 10<30 becomes true, hence output is false.

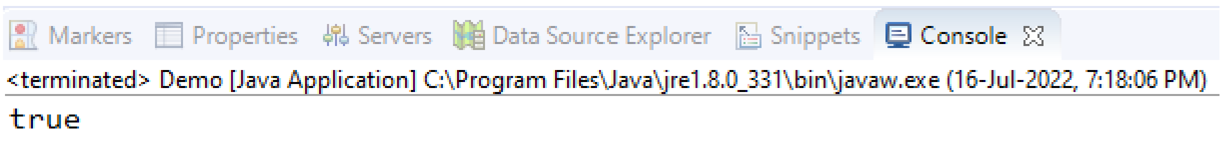


```
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_331\bin\javaw.exe (16-Jul-2022, 7:07:16 PM)
false
```

Example-2

```java
public class Demo {

    public static void main(String[] args) {

        int a = 10;
        int b = 20;
        int c = 30;

        System.out.println((a < b) & (a < c));

    }

}
```

In second example, first condition 10<20 becomes true and second condition 10<30 becomes true, hence output is true.



```
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_331\bin\javaw.exe (16-Jul-2022, 7:18:06 PM)
true
```

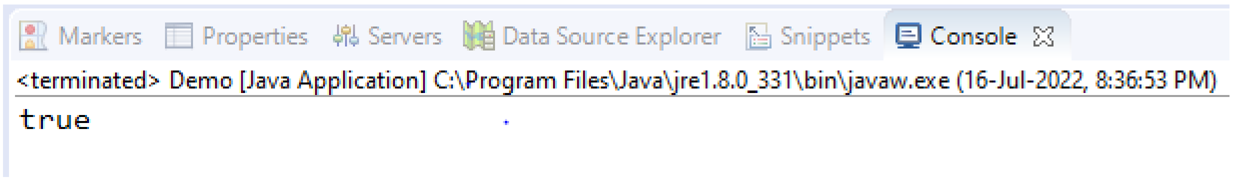
b) Bitwise OR (|) : The bitwise | operator always checks both conditions whether first condition is true or false.

| Expression-1 | Expression-2 | Result |
|---|---|---|
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

Example-1

```
public class Demo {

    public static void main(String[] args) {

        int a = 10;
        int b = 20;
        int c = 30;

        System.out.println((a > b) | (a < c));

    }

}
```

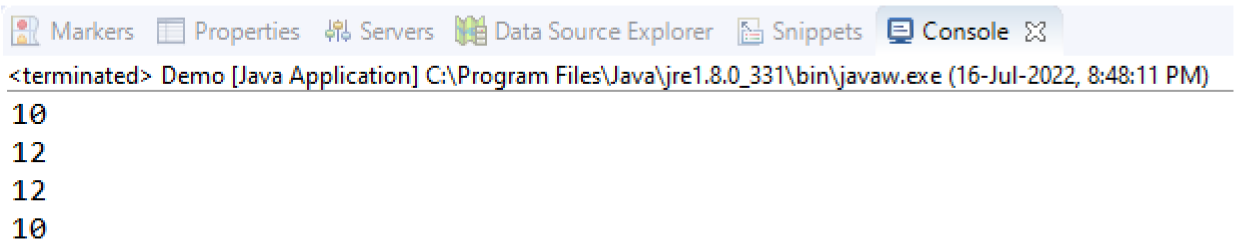
In this example, first condition 10>20 becomes false and second condition 10<30 becomes true; hence output is true.



```
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_331\bin\javaw.exe (16-Jul-2022, 8:36:53 PM)
true
```

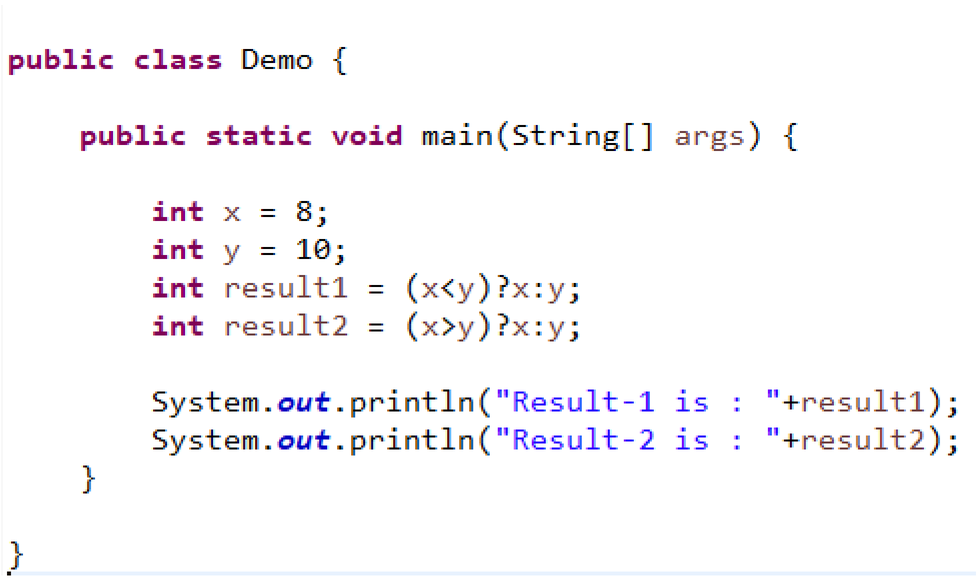
6. Unary operators: These operators are used to perform an operation like increment (++) or decrement (--).

```
public class Demo {

    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

    }

}
```

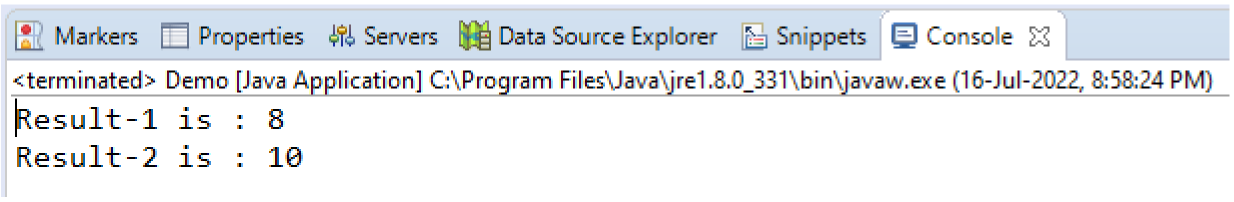


```
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_331\bin\javaw.exe (16-Jul-2022, 8:48:11 PM)
10
12
12
10
```

7. Ternary operators: Java ternary operator is the only conditional operator that takes three operands. It’s a one-liner replacement for the if-then-else statement and is used a lot in Java programming.



```java
public class Demo {

    public static void main(String[] args) {

        int x = 8;
        int y = 10;
        int result1 = (x<y)?x:y;
        int result2 = (x>y)?x:y;

        System.out.println("Result-1 is : "+result1);
        System.out.println("Result-2 is : "+result2);
    }

}
```

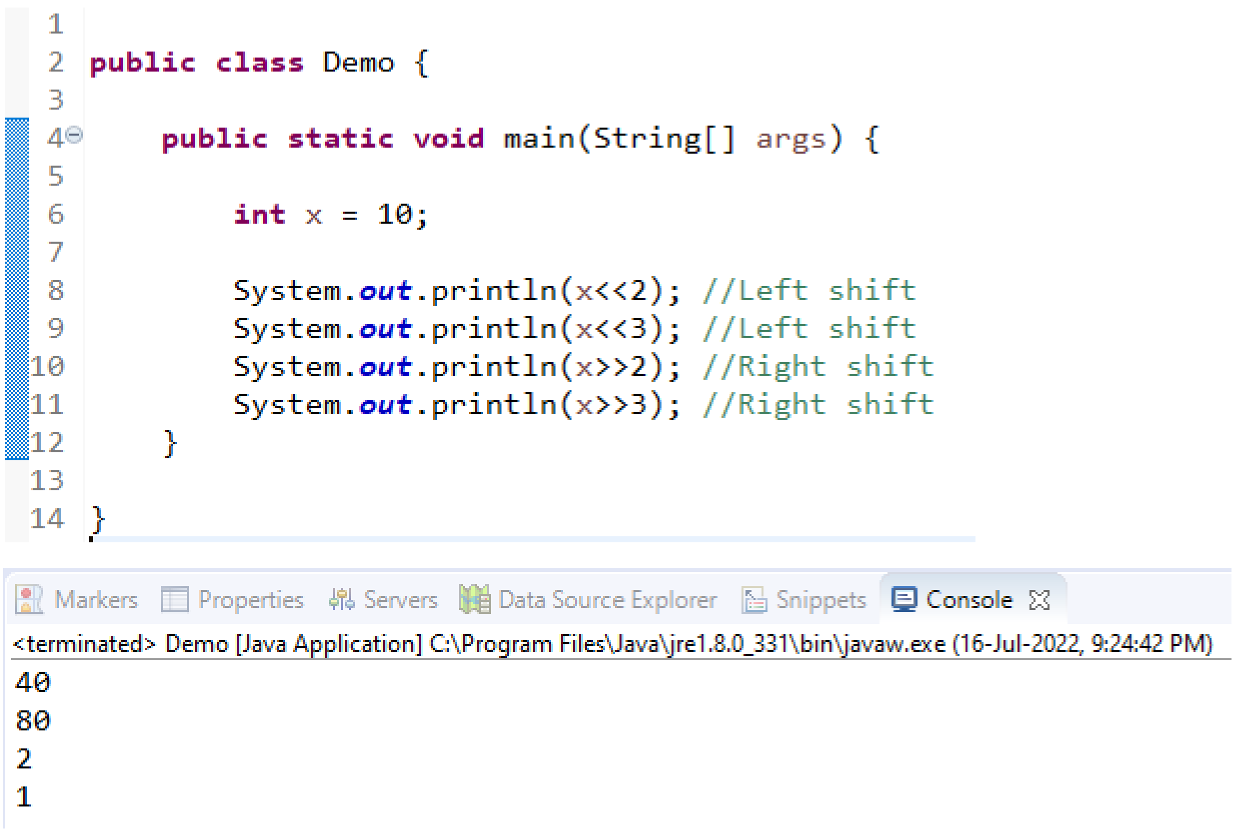


```
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_331\bin\javaw.exe (16-Jul-2022, 8:58:24 PM)
Result-1 is : 8
Result-2 is : 10
```

8. Shift operators:

Right shift operator (>>) : it is used to move left operands value to right by the number of bits specified by the right operand.

Left shift operator (<<) : it is used to shift all of the bits in a value to the left side of a specified number of times.



```
 1
 2 public class Demo {
 3
 4     public static void main(String[] args) {
 5
 6         int x = 10;
 7
 8         System.out.println(x<<2); //Left shift
 9         System.out.println(x<<3); //Left shift
10         System.out.println(x>>2); //Right shift
11         System.out.println(x>>3); //Right shift
12     }
13
14 }
```

```
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_331\bin\javaw.exe (16-Jul-2022, 9:24:42 PM)
40
80
2
1
```

Explanation:

i. On line 8, left shift operators occurs two times (<<), so we write it as 2, On right hand side we shift the position by 2 bits (i.e., numeric 2), Hence statement is $2^2$.
We will always perform the multiplication operation on left shift operators. So, we are putting value of a variable is 10.
Then will calculate, $10 * 2^2$ =?
Square of 2 is 4, so 10 *4 =40.
We will get the output as 40

ii. On line 9, left shift operators occurs two times (<<), so we write it as 2, On right hand side we shift the position by 3 bits (i.e., numeric 3), Hence statement is $2^3$.
We will always perform the multiplication operation on left shift operators. So, we are putting value of a variable is 10.
Then will calculate, $10 * 2^3$ =?
Cube of 2 is 8, so 10 *8 =80.
We will get the output as 80

iii. On line 10, right shift operators occur two times (>>), so we write it as 2, On right hand side we shift the position by 2 bits (i.e., numeric 2), Hence statement is $2^2$.
We will always perform the division operation on right shift operators. So, we are putting value of a variable is 10.
Then will calculate, $10 / 2^2$ =?
Square of 2 is 4, so 10 /4 =2.
We will get the output as 2

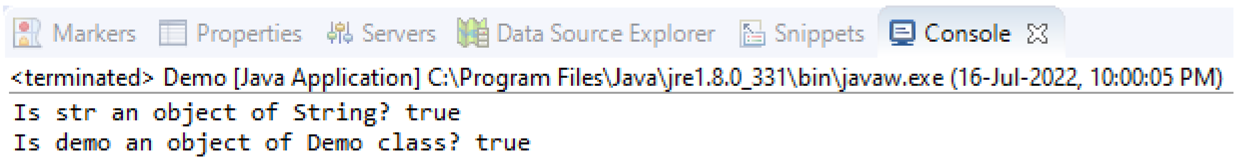
iv. On line 11, right shift operators occur two times (>>), so we write it as 2, On right hand side we shift the position by 3 bits (i.e., numeric 3), Hence statement is $2^3$. We will always perform the division operation on right shift operators. So, we are putting value of a variable is 10.
Then will calculate, 10 / $2^3$ =?
Cube of 2 is 8, so 10 /8 =1.25 but the rounded value is 1.
We will get the output as 1

9. (.) operators: It is used to refer the member of class using class name or objects.

10. new operator: It is used to create the object of class.

11. instanceof Operator: The instanceof operator checks whether an object is an instanceof a particular class.

```java
public class Demo {

    public static void main(String[] args) {

        String str = "Velocity";
        boolean result;
        // checks if str is an instance of
        // the String class
        result = str instanceof String;
        System.out.println("Is str an object of String? " + result);

        Demo demo = new Demo();
        // checks if demo is an instance of
        // the Demo class
        result = demo instanceof Demo;
        System.out.println("Is demo an object of Demo class? " + result);

    }

}
```



```
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.0_331\bin\javaw.exe (16-Jul-2022, 10:00:05 PM)
Is str an object of String? true
Is demo an object of Demo class? true
```