Operators in Java

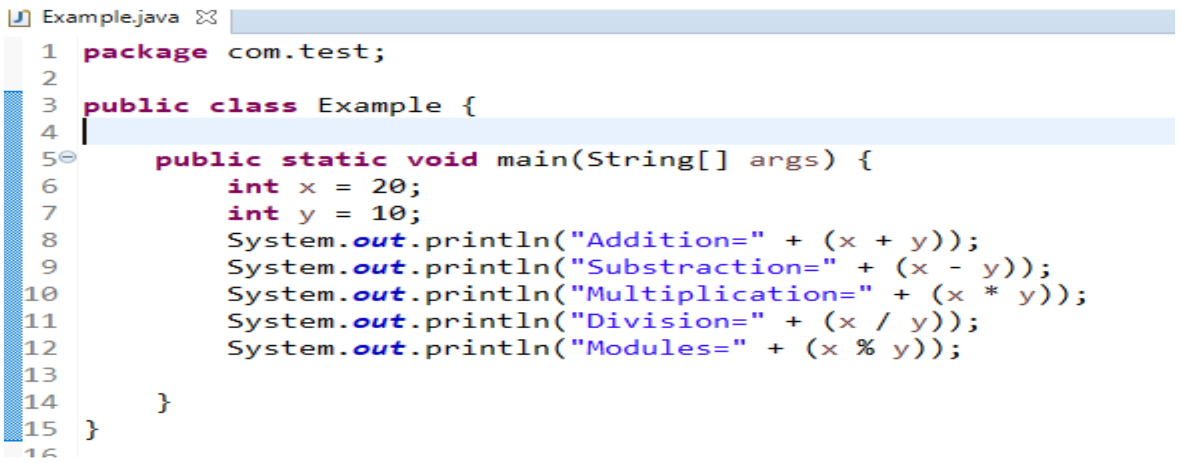
Operators are nothing but to perform some operations on variables called as operators. There is list of types of operators in java are as follow.

- Arithmetic operators
- Logical operators
- Relational operators
- Assignment operators
- Bitwise operators
- Unary operators
- Ternary operators
- Shift operators
- new operators
- . operators

1. Arithmetic operators-

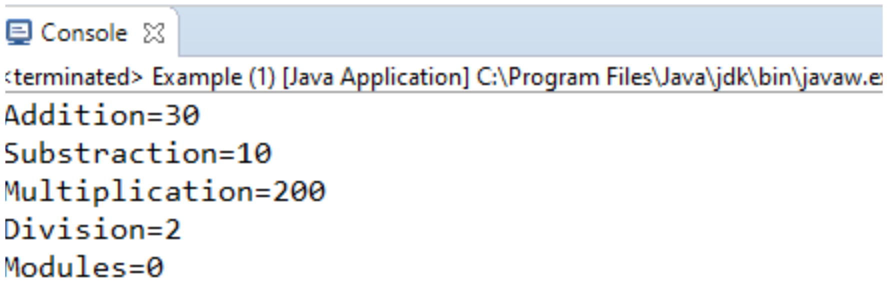
This operator is used to perform some mathematical operation such as addition (+), subtraction (-)
, Multiplication (*), Division (/) and modules (%), etc.

Example-


Example.java

```java
package com.test;

public class Example {

    public static void main(String[] args) {
        int x = 20;
        int y = 10;
        System.out.println("Addition=" + (x + y));
        System.out.println("Substraction=" + (x - y));
        System.out.println("Multiplication=" + (x * y));
        System.out.println("Division=" + (x / y));
        System.out.println("Modules=" + (x % y));

    }
}
```

Output-



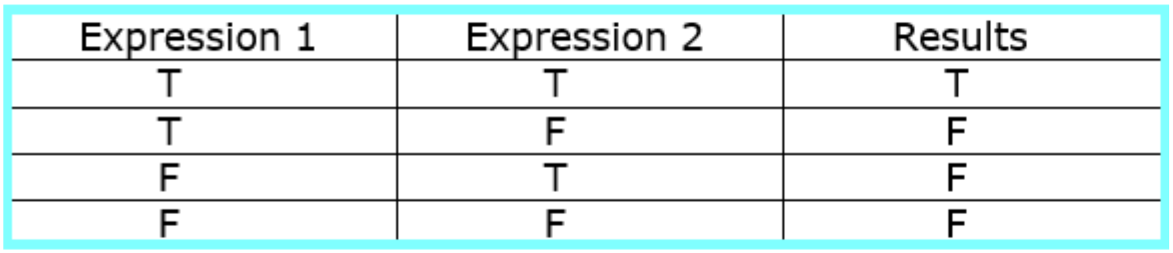
```
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw.e
Addition=30
Substraction=10
Multiplication=200
Division=2
Modules=0
```

2. Logical operators

This operators are used to perform logical AND & OR operation.

1. Logical AND (&&) operators

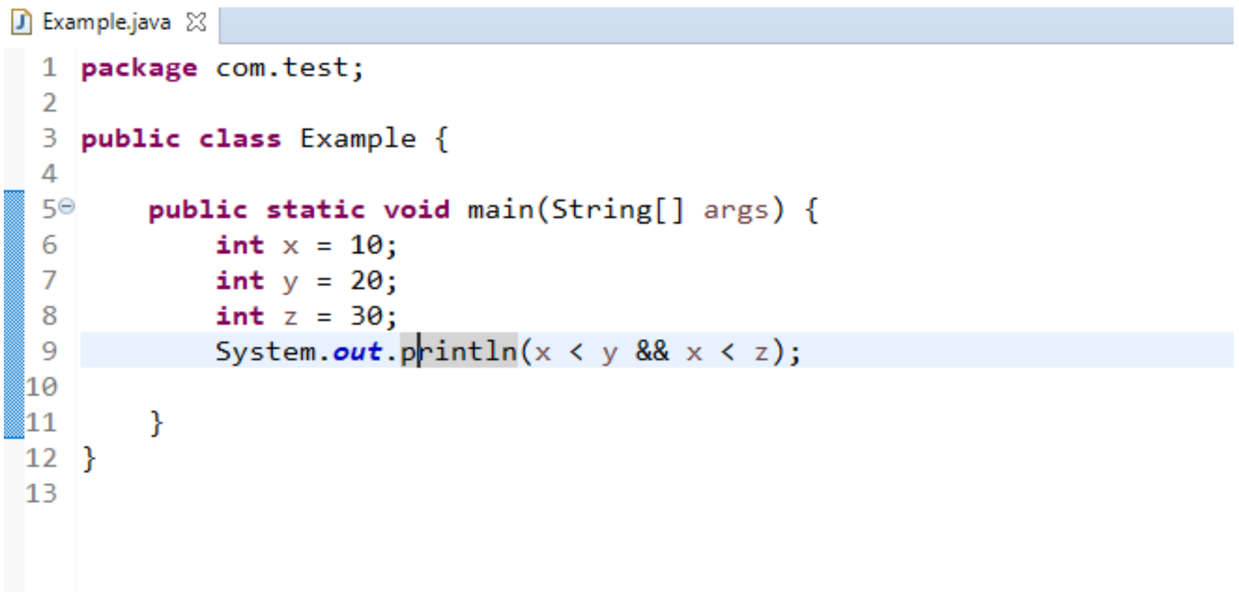
Logical && operator doesn't check second condition if first condition is false. It checks second condition only if first one is true.



Expression 1	Expression 2	Results
T	T	T
T	F	F
F	T	F
F	F	F

Fig- Truth Table for Logical AND operator

Example- Scenario- 1

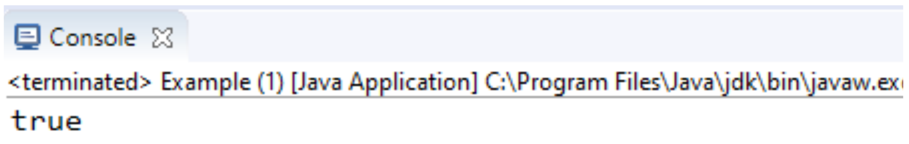


Example.java

```java
package com.test;

public class Example {

    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        int z = 30;
        System.out.println(x < y && x < z);

    }
}
```

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

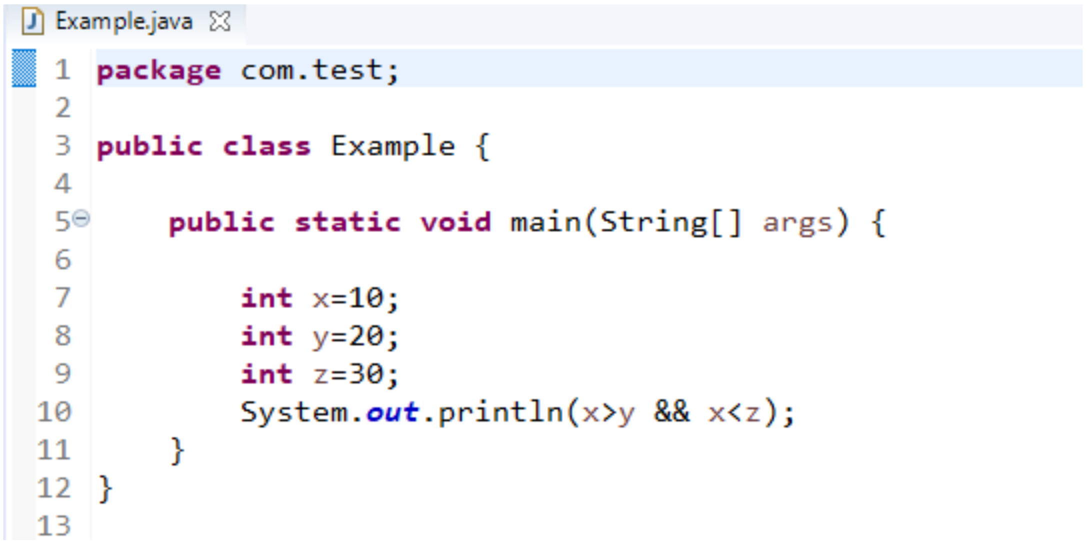
Output-



Console

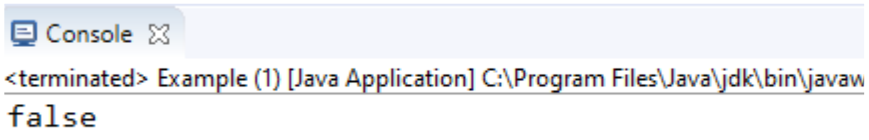
```
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw.ex
true
```

Example- Scenario 2


Example.java

```
package com.test;

public class Example {

    public static void main(String[] args) {

        int x=10;
        int y=20;
        int z=30;
        System.out.println(x>y && x<z);
    }
}

```

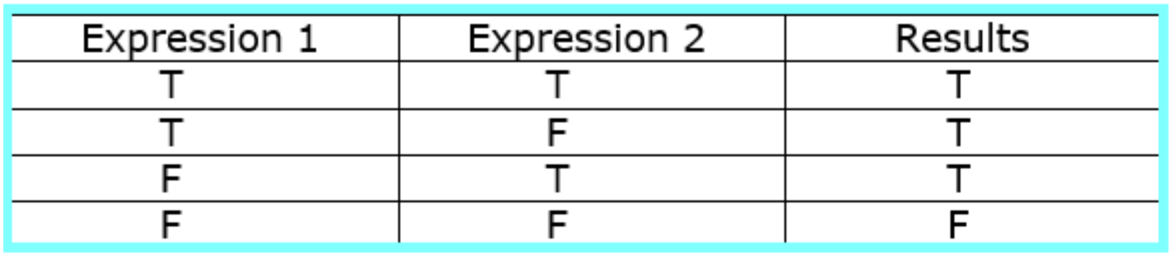
In second example, first condition 10>20 is becomes false and second condition 10<30 is becomes true, hence output is false.


Console

```
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw
false
```

2. Logical OR (||) operators-

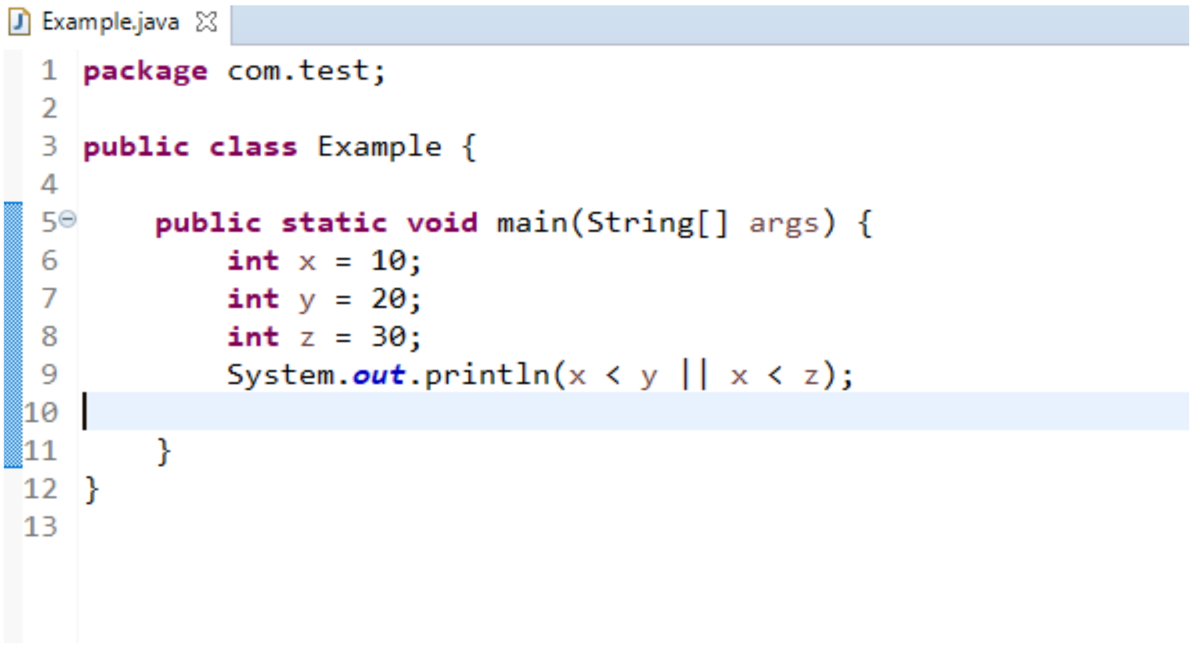
Logical || operator doesn't check second condition if first condition is true. It checks second condition only if first one is false.



Expression 1	Expression 2	Results
T	T	T
T	F	T
F	T	T
F	F	F

Fig- Truth table for Logical OR Operator

Example- Scenario-1

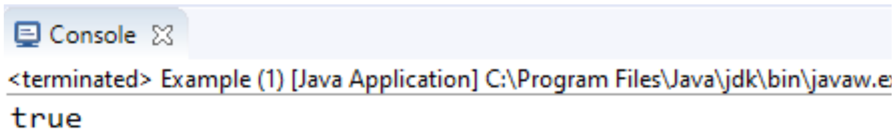


Example.java

```java
package com.test;

public class Example {

    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        int z = 30;
        System.out.println(x < y || x < z);

    }
}

```

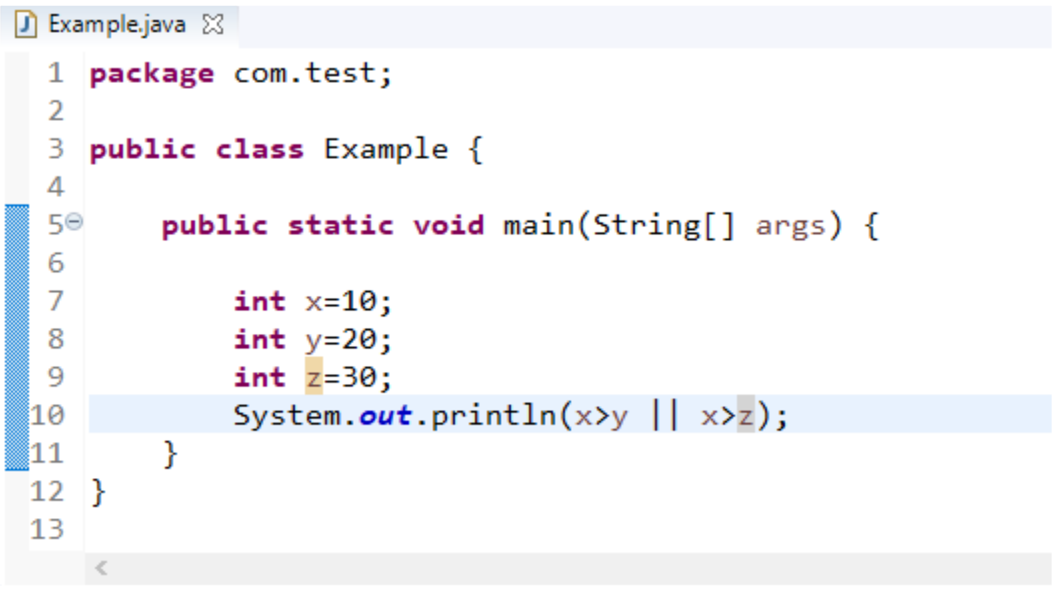
In this example, first condition 10<20 is becomes true and second condition 10<30 is becomes true, hence output is true.

Output-


Console

```
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw.e
true
```

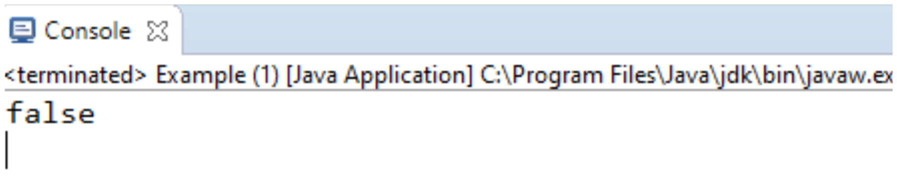
Example- Scenario-2


Example.java

```java
package com.test;

public class Example {

    public static void main(String[] args) {

        int x=10;
        int y=20;
        int z=30;
        System.out.println(x>y || x>z);
    }
}

```

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

Output-



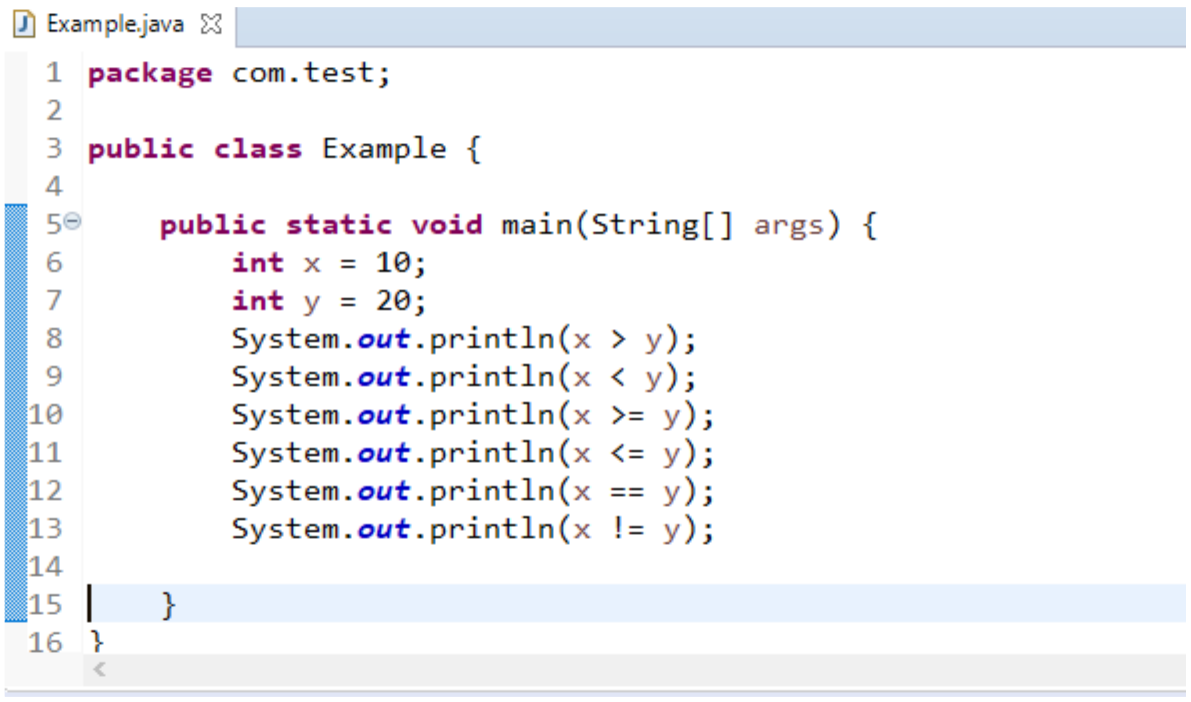
Console

```
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw.ex
false
```

3. Relational Operators-

This operators are used to perform greater than (>), less than (<), greater than or equal to (>=), less than or equal to (<=), equal to (==), not equal to (!=), etc.

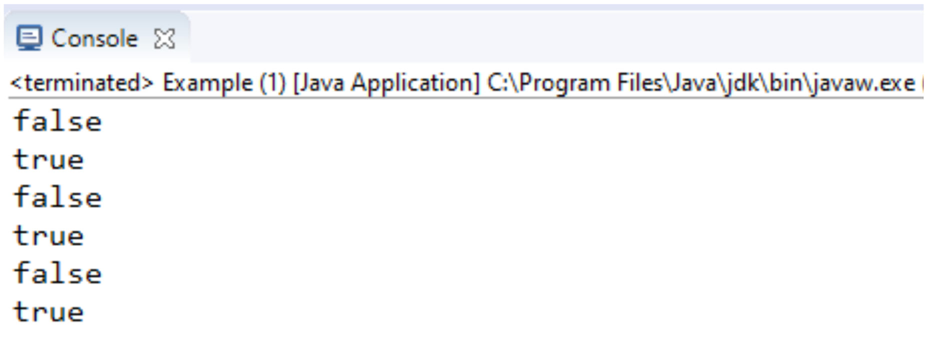
Example-



Example.java

```java
package com.test;

public class Example {

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

    }
}
```

Output-



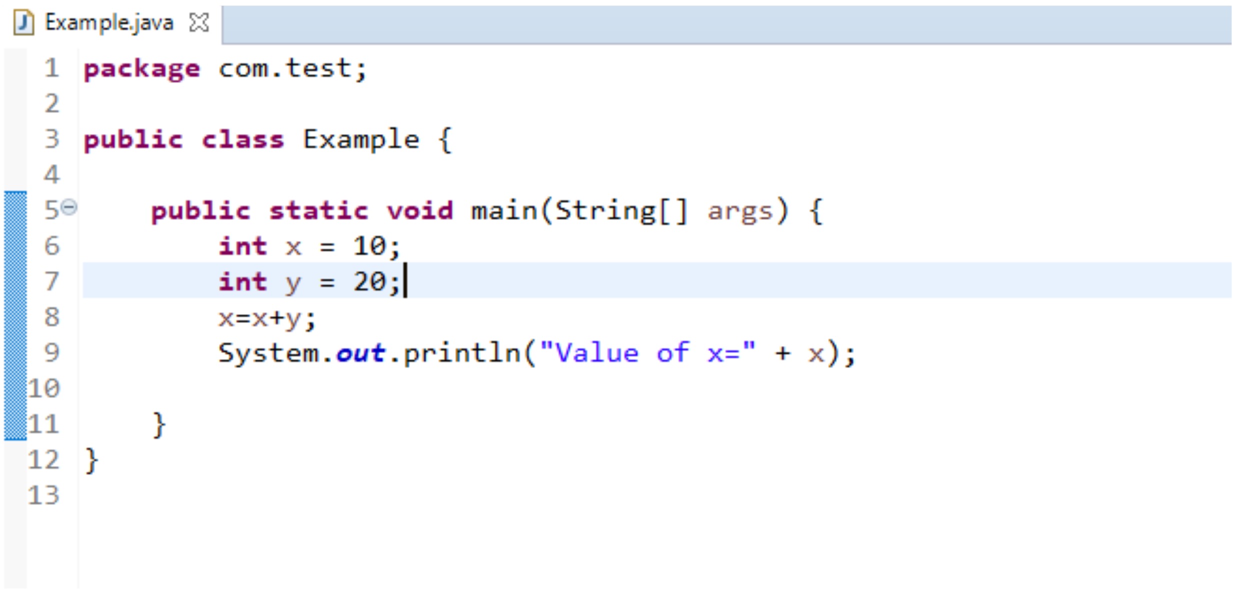
```
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw.exe
false
true
false
true
false
true
```

4. Assignment operators-

This operator is used to assign the values to variable.

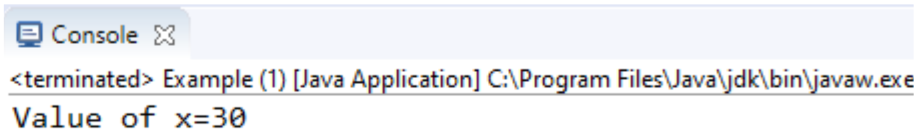
Syntax- Variable =value;

Example-



```java
package com.test;

public class Example {

    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        x=x+y;
        System.out.println("Value of x=" + x);

    }
}

```

Output-

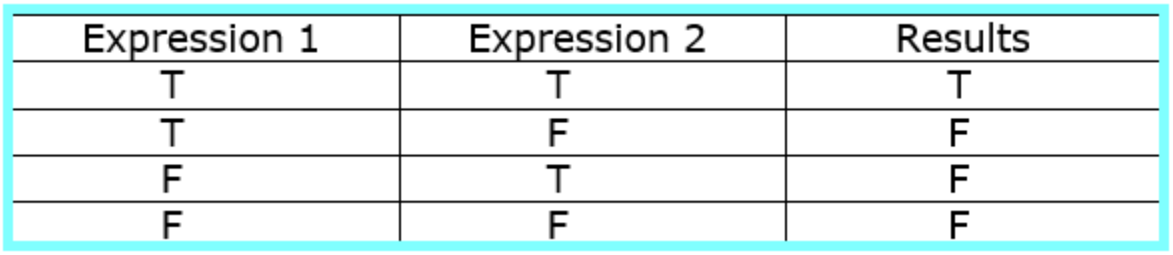


```
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw.exe
Value of x=30
```

5. Bitwise operators-

This operators are used to perform Bitwise AND & OR operation.

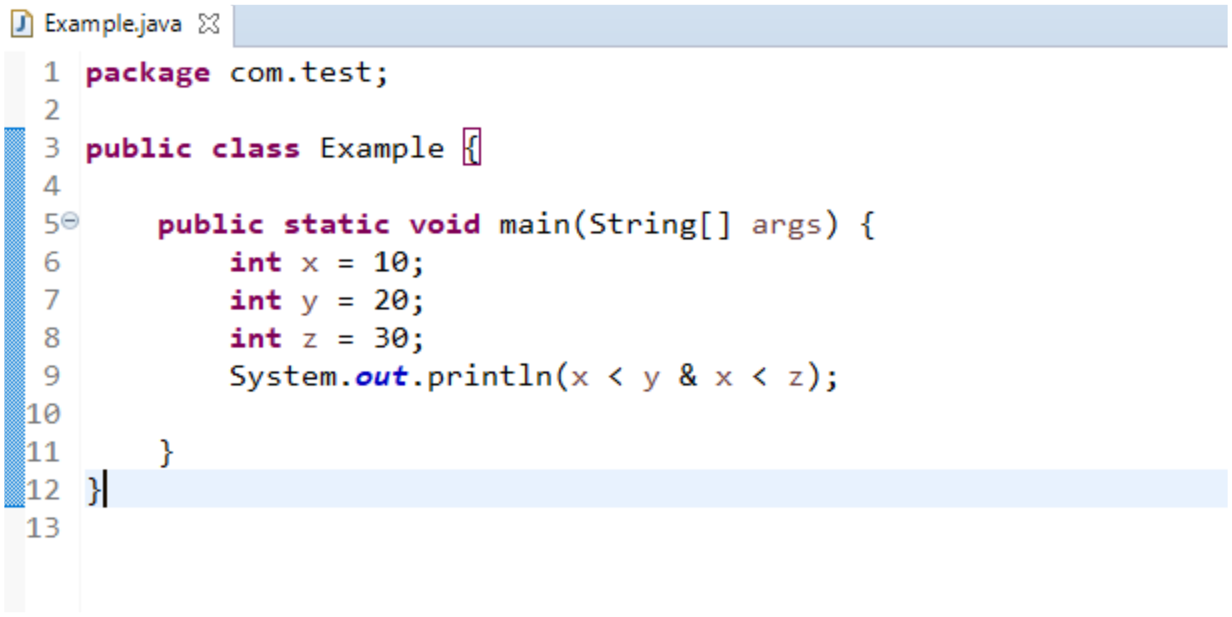
1. **Bitwise AND(&) operators-**
   The bitwise & operator always checks both conditions whether first condition is true or false.



Expression 1	Expression 2	Results
T	T	T
T	F	F
F	T	F
F	F	F

Fig- Truth table for Bitwise AND operator

Example- Scenario-1

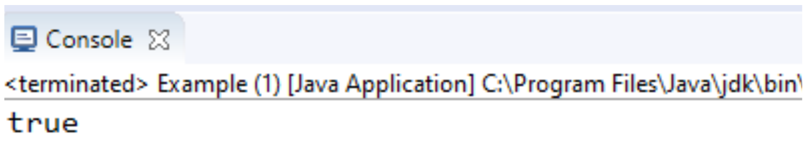


Example.java

```
package com.test;

public class Example {

    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        int z = 30;
        System.out.println(x < y & x < z);

    }
}
```

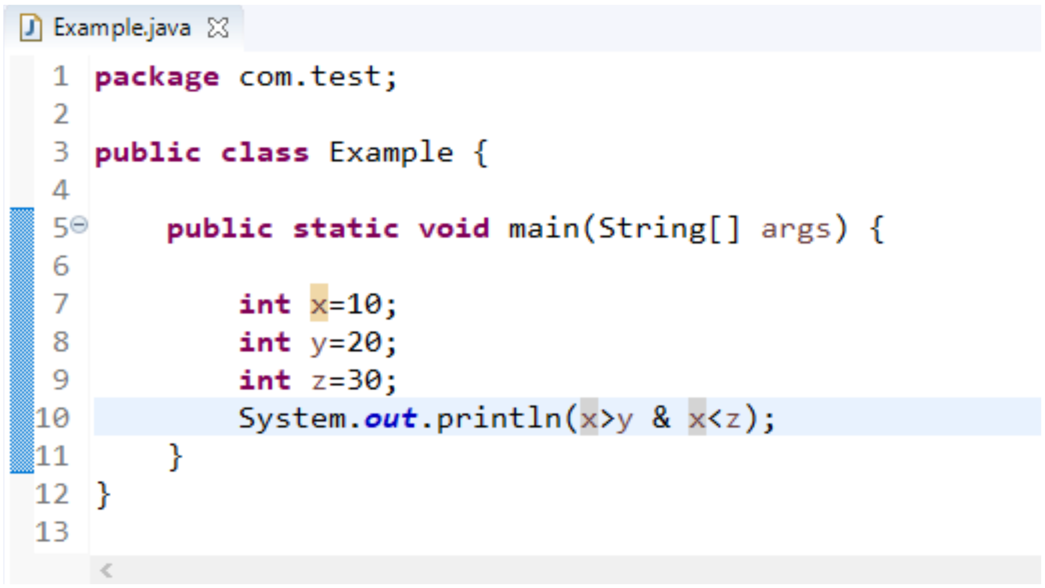
In this example, first condition 10<20 is becomes true and second condition 10<30 is becomes true, hence output is true.

Output-



```
Console
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\
true
```

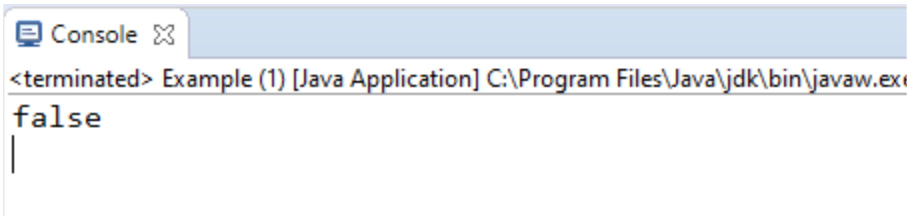
Example- Scenario- 2



```
package com.test;

public class Example {

    public static void main(String[] args) {

        int x=10;
        int y=20;
        int z=30;
        System.out.println(x>y & x<z);
    }
}

```

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

Output-



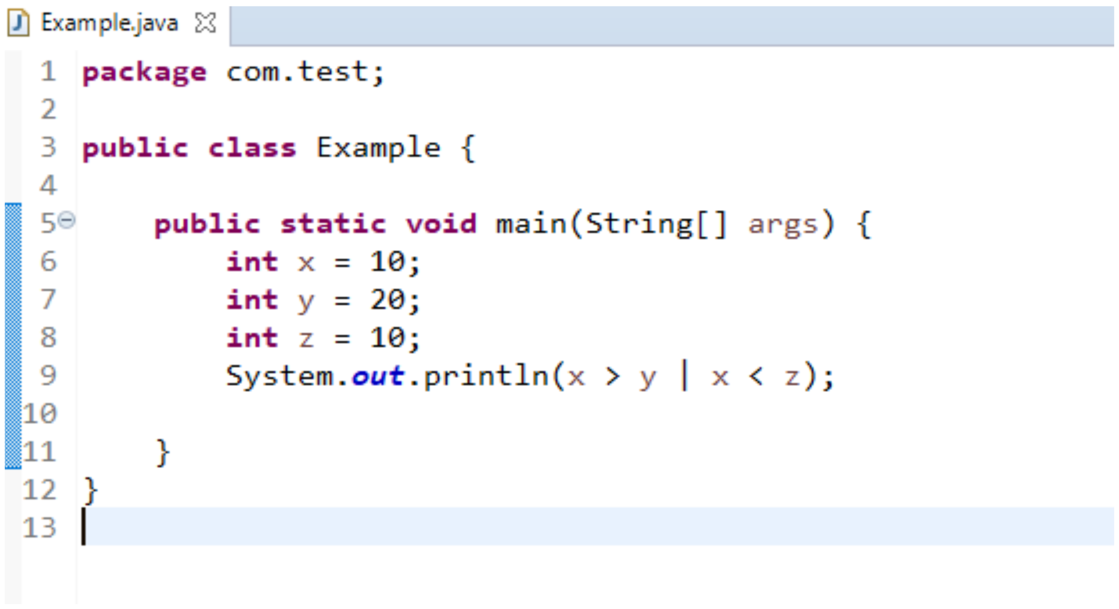
```
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw.exe
false
```

2. **Bitwise OR(|) operators-**
   The bitwise (|) operator always checks both conditions whether first condition is true or false.

Expression 1	Expression 2	Results
T	T	T
T	F	T
F	T	T
F	F	F

Fig- Truth table for Bitwise OR operator

Example-Scenario-1

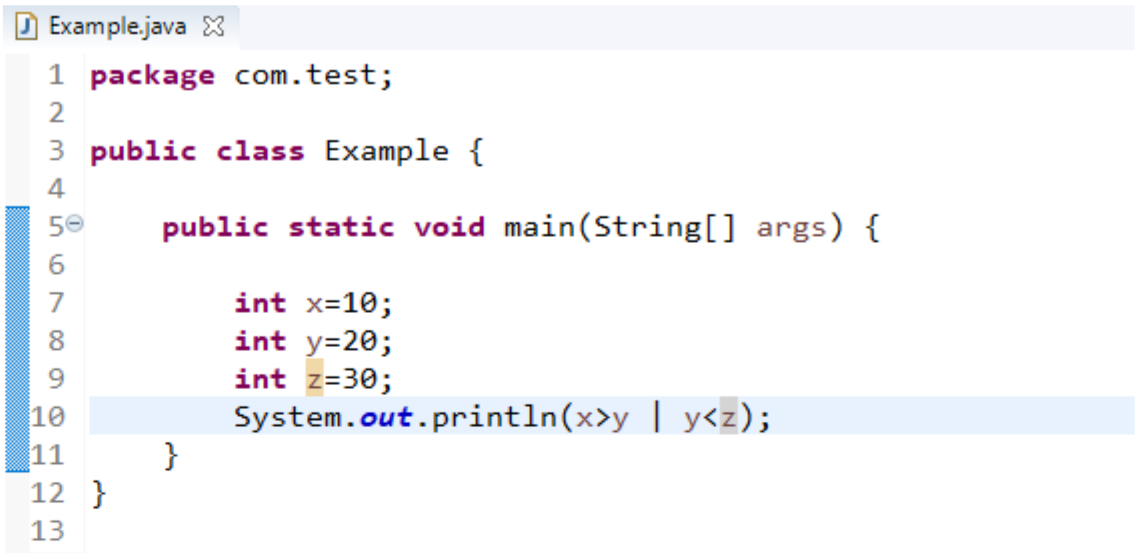


```java
package com.test;

public class Example {

    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        int z = 10;
        System.out.println(x > y | x < z);

    }
}

```

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

Output-

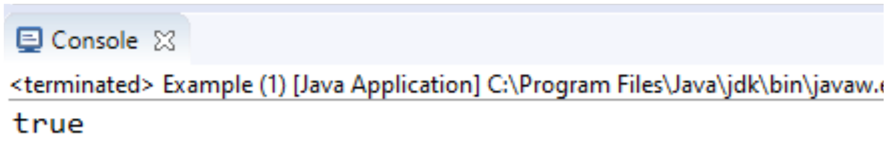
Example- Scenario- 2



```java
package com.test;

public class Example {

    public static void main(String[] args) {

        int x=10;
        int y=20;
        int z=30;
        System.out.println(x>y | y<z);
    }
}

```

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

Output-

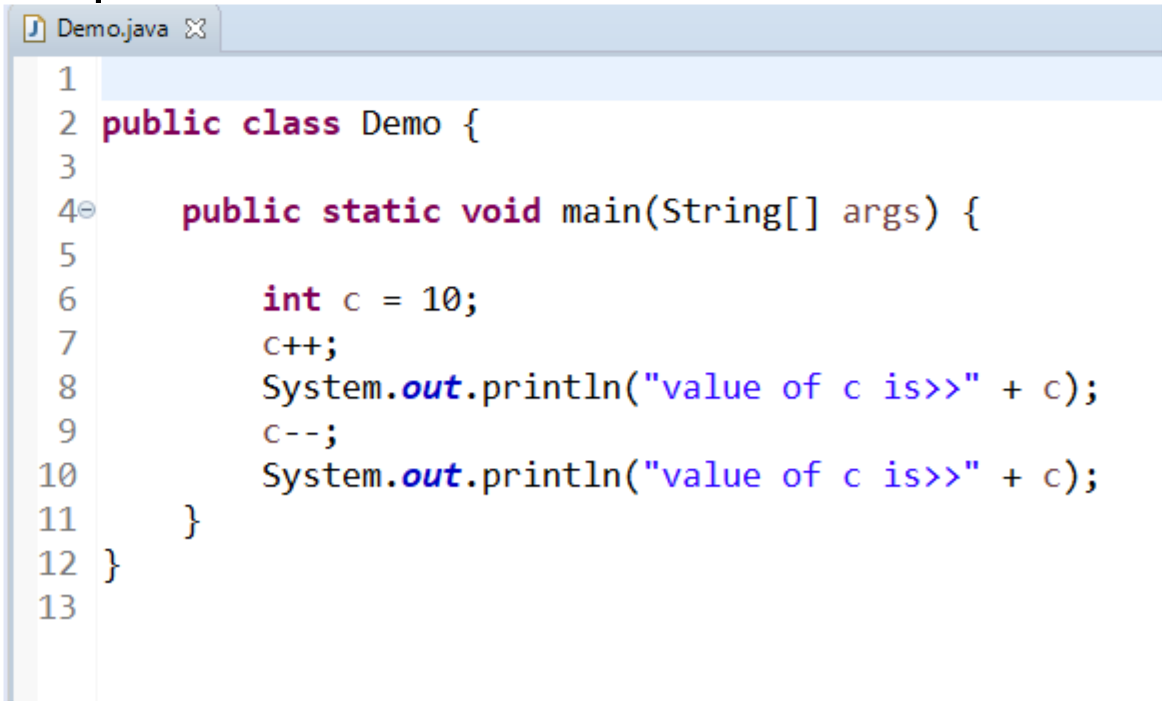


```
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw.
true
```

6. Unary operators-

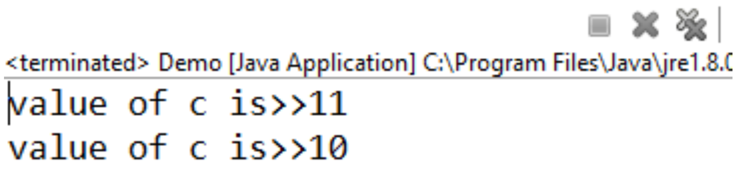
This operators are used to perform an operation like increment (++) or decrement (--).

Example



```java

public class Demo {

    public static void main(String[] args) {

        int c = 10;
        c++;
        System.out.println("value of c is>>" + c);
        c--;
        System.out.println("value of c is>>" + c);
    }
}

```

Output-



```
<terminated> Demo [Java Application] C:\Program Files\Java\jre1.8.
value of c is>>11
value of c is>>10
```

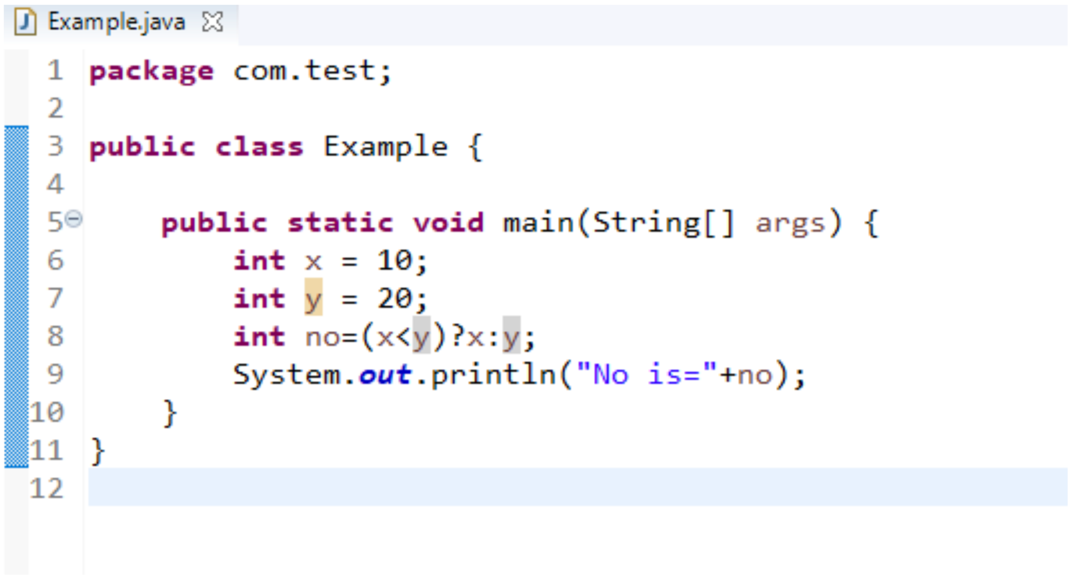
7. Ternary operators-

It includes three operands.

Why?

If else statement requires group of line code to execute the statement but by using this, we can write the code into one line only.

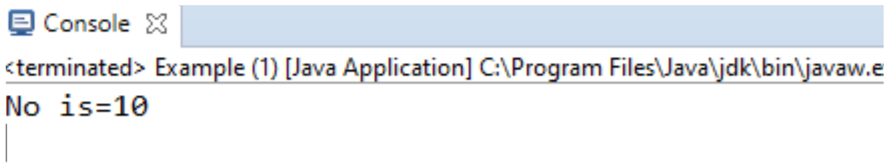
Example-


Example.java

```java
package com.test;

public class Example {

    public static void main(String[] args) {
        int x = 10;
        int y = 20;
        int no=(x<y)?x:y;
        System.out.println("No is="+no);
    }
}

```

In this example, condition 10<20 becomes true, so output is 10.

Output


Console

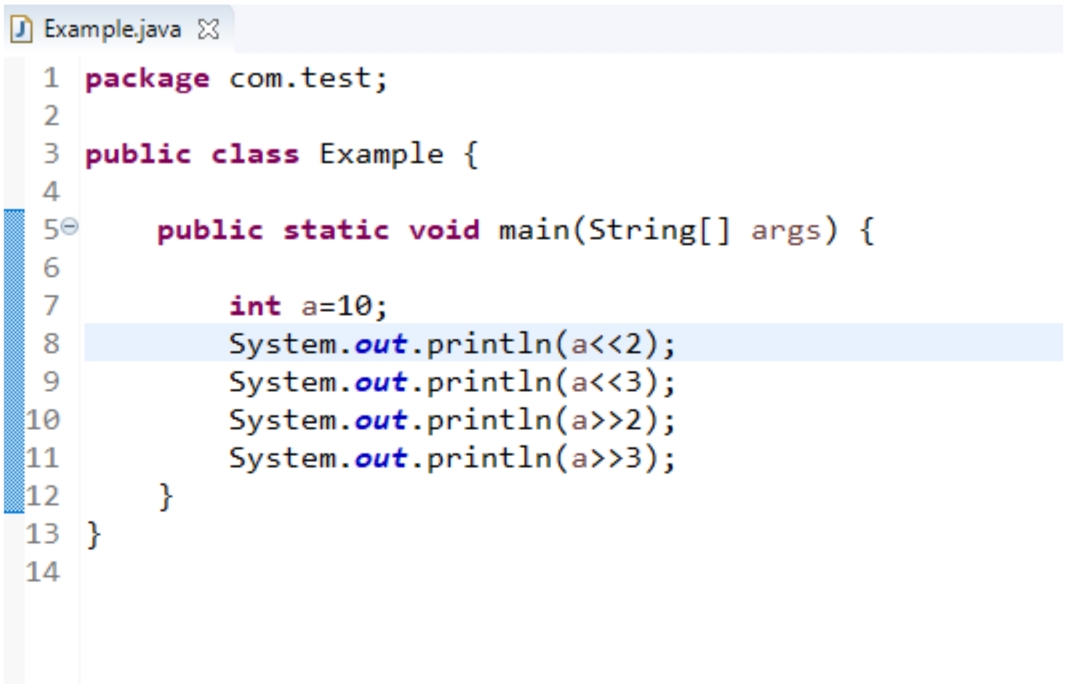
```
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw.e
No is=10
```

8. Shift operators (Right/Left)-

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

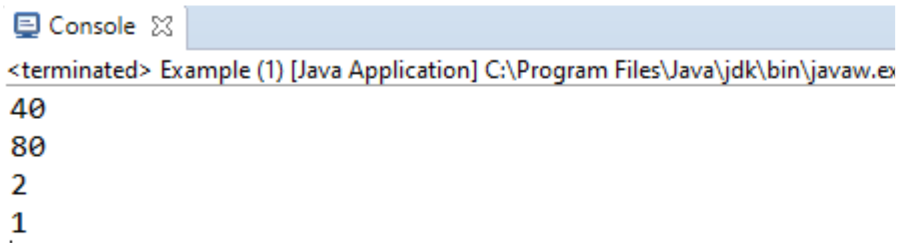
Left shift operator << is used to shift all of the bits in a value to the left side of a specified number of times.

Example-



```
Example.java
 1 package com.test;
 2
 3 public class Example {
 4
 5     public static void main(String[] args) {
 6
 7         int a=10;
 8         System.out.println(a<<2);
 9         System.out.println(a<<3);
10         System.out.println(a>>2);
11         System.out.println(a>>3);
12     }
13 }
14
```

Output-



```
Console
<terminated> Example (1) [Java Application] C:\Program Files\Java\jdk\bin\javaw.ex
40
80
2
1
```

1. On line 9, left shift operators occurs two times (<<), so we write it as 2, 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**

2. On line 11, right shift operators occurs two times (>>), so we write it as 2, 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**.

3. 10 /22 = 2
4. 10 /23 = 1

9. (.) operators

It is used to refer the member of class using class name or objects.

10. new operators

It is used to create the object of class.