BCA – 401: Java Programming

Rahul Kumar Singh

In today's Class we have discussed on various Operators in Java.

Java Operators:-

Operators are used to perform operations on variables and values. It is a symbol that is used to perform operations. For example: +, -, *, / etc.

Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups −

- **Arithmetic Operators**
- **Relational Operators**
- **Bitwise Operators**
- **Logical Operators**
- **Assignment Operators**
- **Misc Operators**

Arithmetic Operators:-

Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. These operators are used to perform common mathematical operations.

Types of Arithmetic operators:-

There are following Arithmetic operators supported by Java language.

Addition(+) operator:-

This operator is used to adds values on either side of the operator.

Example:-

Assume integer variable A holds 10 and variable B holds 20, then –

A + B will give 30

Subtraction(-) operator:-

This operator is used to subtracts right-hand operand from left-hand operand.

Example:-

Assume integer variable A holds 10 and variable B holds 20, then –

A - B will give -10

Multiplication(*) operator:-

This operator is used to multiplies values on either side of the operator.

Example:-

Assume integer variable A holds 10 and variable B holds 20,

then −

A * B will give 200

Division(/) operator:-

This operator is used to divides left-hand operand by right-hand operand.

Example:-

Assume integer variable A holds 10 and variable B holds 20, then −

B / A will give 2

Modulus(%) operator:-

This operator is used to divides left-hand operand by right-hand operand and returns remainder.

Example:-

Assume integer variable A holds 10 and variable B holds 20, then −

B % A will give 0

Increment(++) operator:-

This operator is used to increases the value of operand by 1.

Example:-

Assume integer variable B holds 20, then −

B++ gives 21

Decrement(--) operator:-

This operator is used to decreases the value of operand by 1.

Example:-

Assume integer variable B holds 20, then –

B-- gives 19

Example:- The following program is a simple example which demonstrates the arithmetic operators.

```
public class Test
{
   public static void main(String args[])
{
      int a = 10;
      int b = 20;
      int c = 25;
      int d = 25;
      System.out.println("a + b = " + (a + b) );
      System.out.println("a - b = " + (a - b) );
      System.out.println("a * b = " + (a * b) );
      System.out.println("b / a = " + (b / a) );
```

```
        System.out.println("b % a = " + (b % a) );
        System.out.println("c % a = " + (c % a) );
        System.out.println("a++   = " +  (a++) );
        System.out.println("a-- = " + (a--) );
        // Check the difference in d++ and ++d
        System.out.println("d++   = " +  (d++) );
        System.out.println("++d   = " +  (++d) );
    }
}
```

Output:-

```
a + b = 30
a - b = -10
a * b = 200
b / a = 2
b % a = 0
c % a = 5
a++   = 10
b--   = 11
d++   = 25
++d   = 27
```

Relational Operators:-

Relational operators are also known as Comparison operators. These operators are used to compare two values (or variables).

The return value of a comparison is either true or false. These values are known as Boolean values. This is important in programming, because it helps us to find answers and make decisions.

Types of Relational Operator:-

There are following relational operators supported by Java language.

== (equal to) operator:-

Checks if the values of two operands are equal or not, if yes then condition becomes true.

Example:-

Assume variable A holds 10 and variable B holds 20, then −

(A == B) is not true.

!= (not equal to) operator:-

Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.

Example:-

Assume variable A holds 10 and variable B holds 20, then −

(A != B) is true.

> (greater than) operator:-

Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.

Example:-

Assume variable A holds 10 and variable B holds 20, then −

(A > B) is not true.

< (less than) operator:-

Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.

Example:-

Assume variable A holds 10 and variable B holds 20, then −

(A < B) is true.

>= (greater than or equal to) operator:-

Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.

Example:-

Assume variable A holds 10 and variable B holds 20, then −

(A >= B) is not true.

<= (less than or equal to) operator:-

Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes

true.

Example:-

Assume variable A holds 10 and variable B holds 20, then −

(A <= B) is true.

Example:- The following program is a simple example that demonstrates the relational operators.

```
public class Test
{
   public static void main(String args[])
{
      int a = 10;
      int b = 20;
      System.out.println("a == b = " + (a == b) );
      System.out.println("a != b = " + (a != b) );
      System.out.println("a > b = " + (a > b) );
      System.out.println("a < b = " + (a < b) );
      System.out.println("b >= a = " + (b >= a) );
      System.out.println("b <= a = " + (b <= a) );
   }
}
```

Output:-

This will produce the following result −

a == b = false

a != b = true

a > b = false

a < b = true

b >= a = true

b <= a = false

Bitwise Operators:-

Java defines several bitwise operators, which can be applied to the integer types, long, int, short, char, and byte.

Bitwise operator works on bits and performs bit-by-bit operation.

Example:- Assume if a = 60 and b = 13; now in binary format they will be as follows −

a = 0011 1100

b = 0000 1101

---

a&b = 0000 1100

a|b = 0011 1101

a^b = 0011 0001

~a = 1100 0011

Types of Bitwise Operators:-

There are following Bitwise Operators supported by Java language.

& (bitwise and) operator:-

Binary AND Operator copies a bit to the result if it exists in both operands.

Example:-

Assume integer variable A holds 60 and variable B holds 13 then −

(A & B) will give 12 which is 0000 1100

| (bitwise or) operator:-

Binary OR Operator copies a bit if it exists in either operand.

Example:-

Assume integer variable A holds 60 and variable B holds 13 then −

(A | B) will give 61 which is 0011 1101

^ (bitwise XOR) operator:-

Binary XOR Operator copies the bit if it is set in one operand but not both.

Example:-

Assume integer variable A holds 60 and variable B holds 13 then −

(A ^ B) will give 49 which is 0011 0001

~ (bitwise compliment) operator:-

Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.

Example:-

Assume integer variable A holds 60 then −

(~A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number.

<< (left shift) operator:-

Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.

Example:-

Assume integer variable A holds 60 then −

A << 2 will give 240 which is 1111 0000

>> (right shift) operator:-

Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.

Example:-

Assume integer variable A holds 60 then −

A >> 2 will give 15 which is 1111

>>> (zero fill right shift) operator:-

Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.

Example:-

Assume integer variable A holds 60 then –

A >>>2 will give 15 which is 0000 1111

Example:- The following program is a simple example that demonstrates the bitwise operators.

```
public class Test
{
  public static void main(String args[])
{
    int a = 60; /* 60 = 0011 1100 */
    int b = 13; /* 13 = 0000 1101 */
    int c = 0;
    c = a & b;       /* 12 = 0000 1100 */
    System.out.println("a & b = " + c );
    c = a | b;       /* 61 = 0011 1101 */
```

```
      System.out.println("a | b = " + c );
      c = a ^ b;        /* 49 = 0011 0001 */
      System.out.println("a ^ b = " + c );
      c = ~a;           /*-61 = 1100 0011 */
      System.out.println("~a = " + c );
      c = a << 2;       /* 240 = 1111 0000 */
      System.out.println("a << 2 = " + c );
      c = a >> 2;       /* 15 = 1111 */
      System.out.println("a >> 2  = " + c );
      c = a >>> 2;      /* 15 = 0000 1111 */
      System.out.println("a >>> 2 = " + c );
   }
}
```

Output:-

This will produce the following result −

```
a & b = 12
a | b = 61
a ^ b = 49
~a = -61
a << 2 = 240
```

a >> 2 = 15

a >>> 2 = 15

Logical Operators:-

Logical operators are used to determine the logic between variables or values. You can also test for true or false values with logical operators.

Types of Logical operator:-

There are following Logical Operators supported by Java language.

&& (logical and) operator:-

It is called Logical AND operator. If both the operands are non-zero, then the condition becomes true.

Example:-

Assume Boolean variables A holds true and variable B holds false, then −

(A && B) is false

|| (logical or) operator:-

It is called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.

Example:-

Assume Boolean variables A holds true and variable B holds

false, then −

(A || B) is true

! (logical not) operator:-

It is called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.

Example:-

Assume Boolean variables A holds true and variable B holds false, then −

!(A && B) is true

Example:- The following simple example program demonstrates the logical operators.

```
public class Test
{
   public static void main(String args[])
{
      boolean a = true;
      boolean b = false;
      System.out.println("a && b = " + (a&&b));
      System.out.println("a || b = " + (a||b) );
```

```
      System.out.println("!(a && b) = " + !(a && b));
   }
}
```

Output:-

This will produce the following result –

```
a && b = false
a || b = true
!(a && b) = true
```

Assignment Operators:-

Assignment operators are used to assign values to variables.

Types of Assignment operators:-

Following are the assignment operators supported by Java language –

= Simple assignment operator:-

Assigns values from right side operands to left side operand.

Example:-

C = A + B will assign value of A + B into C

+= Add AND assignment operator:-

It adds right operand to the left operand and assign the result to left operand.

Example:-

C += A is equivalent to C = C + A

-= Subtract AND assignment operator:-

It subtracts right operand from the left operand and assign the result to left operand.

Example:-

C -= A is equivalent to C = C – A

***= Multiply AND assignment operator:-**

It multiplies right operand with the left operand and assign the result to left operand.

Example:-

C *= A is equivalent to C = C * A

/= Divide AND assignment operator:-

It divides left operand with the right operand and assign the result to left operand.

Example:-

C /= A is equivalent to C = C / A

%= Modulus AND assignment operator:-

It takes modulus using two operands and assign the result to left operand.

Example:-

C %= A is equivalent to C = C % A

<<= Left shift AND assignment operator:-

Example:-

C <<= 2 is same as C = C << 2

>>= Right shift AND assignment operator:-

Example:-

C >>= 2 is same as C = C >> 2

&= Bitwise AND assignment operator:-

Example:-

C &= 2 is same as C = C & 2

^= Bitwise exclusive OR and assignment operator:-

Example:-

C ^= 2 is same as C = C ^ 2

|= Bitwise inclusive OR and assignment operator:-

Example:-

C |= 2 is same as C = C | 2

Example:- The following program is a simple example that demonstrates the assignment operators.

```
public class Test
{
   public static void main(String args[])
{
      int a = 10;
      int b = 20;
      int c = 0;
      c = a + b;
      System.out.println("c = a + b = " + c );
      c += a ;
      System.out.println("c += a  = " + c );
      c -= a ;
      System.out.println("c -= a = " + c );
      c *= a ;
      System.out.println("c *= a = " + c );
      a = 10;
      c = 15;
      c /= a ;
      System.out.println("c /= a = " + c );
```

```
      a = 10;
      c = 15;
      c %= a ;
      System.out.println("c %= a  = " + c );
      c <<= 2 ;
      System.out.println("c <<= 2 = " + c );
      c >>= 2 ;
      System.out.println("c >>= 2 = " + c );
      c >>= 2 ;
      System.out.println("c >>= 2 = " + c );
      c &= a ;
      System.out.println("c &= a  = " + c );
      c ^= a ;
      System.out.println("c ^= a   = " + c );
      c |= a ;
      System.out.println("c |= a   = " + c );
   }
}
```

Output:-

This will produce the following result −

c = a + b = 30

c += a = 40

c -= a = 30

c *= a = 300

c /= a = 1

c %= a = 5

c <<= 2 = 20

c >>= 2 = 5

c >>= 2 = 1

c &= a = 0

c ^= a   = 10

c |= a   = 10

Miscellaneous Operators:-

There are few other operators supported by Java Language.

Conditional Operator/ternary operator (? :):-

Conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide, which value should be assigned to the variable.

Syntax:- This operator is written as −

variable x = (expression) ? value if true : value if false

Example:-

Following is an example −

```
public class Test
{
   public static void main(String args[])
{
      int a, b;
      a = 10;
      b = (a == 1) ? 20: 30;
      System.out.println( "Value of b is : " +  b );
      b = (a == 10) ? 20: 30;
      System.out.println( "Value of b is : " + b );
   }
}
```

Output:-

This will produce the following result −

```
Value of b is : 30
Value of b is : 20
```

instanceof Operator:-

This operator is used only for object reference variables. The operator checks whether the object is of a particular type (class type or interface type).

Syntax:- instanceof operator is written as –

(Object reference variable) instanceof (class/interface type)

If the object referred by the variable on the left side of the operator passes the IS-A check for the class/interface type on the right side, then the result will be true.

Example:-

Following is an example –

```
public class Test
{
  public static void main(String args[])
{
    String name = "Daivik";
    // following will return true since name is type of String
    boolean result = name instanceof String;
    System.out.println( result );
  }
}
```

Output:-

This will produce the following result −

true