**Operators in Java**

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:

1. Arithmetic Operators
2. Unary Operators
3. Assignment Operator
4. Relational Operators
5. Logical Operators
6. Ternary Operator
7. Bitwise Operators
8. Shift Operators
9. instance of operator

Let’s take a look at them in detail.

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

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

1. **Unary Operators:** Unary operators need only one operand. They are used to increment, decrement or negate a value.

* **– :** **Unary minus**, used for negating the values.
* **+ :** **Unary plus** indicates the positive value (numbers are positive without this, however). It performs an automatic conversion to int when the type of its operand is the byte, char, or short. This is called unary numeric promotion.
* **++ :** **Increment operator**, used for incrementing the value by 1. There are two varieties of increment operators.
  + **Post-Increment:**Value is first used for computing the result and then incremented.
  + **Pre-Increment:**Value is incremented first, and then the result is computed.
* **— : Decrement operator**, used for decrementing the value by 1. There are two varieties of decrement operators.
  + **Post-decrement:**Value is first used for computing the result and then decremented.
  + **Pre-Decrement:**Value is decremented first, and then the result is computed.
* **! : Logical not operator**, used for inverting a boolean value.

1. **Assignment Operator: ‘=’** Assignment operator is used to assign a value to any variable. It has a right to left associativity, i.e. value given on the right-hand side of the operator is assigned to the variable on the left, and therefore right-hand side value must be declared before using it or should be a constant.

The general format of the assignment operator is:

variable **=** value;

In many cases, the assignment operator can be combined with other operators to build a shorter version of the statement called a **Compound Statement**. For example, instead of a **=** a+5, we can write a **+=** 5.

* **+=**, for adding left operand with right operand and then assigning it to the variable on the left.
* **-=**, for subtracting right operand from left operand and then assigning it to the variable on the left.
* **\*=**, for multiplying left operand with right operand and then assigning it to the variable on the left.
* **/=**, for dividing left operand by right operand and then assigning it to the variable on the left.
* **%=**, for assigning modulo of left operand by right operand and then assigning it to the variable on the left.

1. **Relational Operators:** These operators are used to check for relations like equality, greater than, less than. They return boolean results after the comparison and are extensively used in looping statements as well as conditional if-else statements. The general format is,

variable **relation\_operator** value

Some of the relational operators are-

* **==, Equal to:**returns true if the left-hand side is equal to the right-hand side.
* **!=, Not Equal to:**returns true if the left-hand side is not equal to the right-hand side.
* **<, less than:**returns true if the left-hand side is less than the right-hand side.
* **<=, less than or equal to**returns true if the left-hand side is less than or equal to the right-hand side.
* **>, Greater than:**returns true if the left-hand side is greater than the right-hand side.
* **>=, Greater than or equal to:**returns true if the left-hand side is greater than or equal to the right-hand side.

1. **Logical Operators:** These operators are used to perform “logical AND” and “logical OR” operations, i.e., the function similar to AND gate and OR gate in digital electronics. One thing to keep in mind is the second condition is not evaluated if the first one is false, i.e., it has a short-circuiting effect. Used extensively to test for several conditions for making a decision. Java also have “Logical NOT”, it returns true when condition is false and vice-versa

*Conditional operators are:*

* **&&, Logical AND:**returns true when both conditions are true.
* **||, Logical OR:**returns true if at least one condition is true.
* **! , Logical NOT:**returns true when condition is false and vice-versa

1. **Ternary operator:** Ternary operator is a shorthand version of the if-else statement. It has three operands and hence the name ternary.

The general format is:

condition **?** if true **:** if false

The above statement means that if the condition evaluates to true, then execute the statements after the ‘?’ else execute the statements after the ‘:.’

//Java program to illustrate

//max of three numbers using

//ternary operator.

**public** **class** operators {

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

{

**int** a = 20, b = 10, c = 30, result;

// result holds max of three

// numbers

result

= ((a > b) ? (a > c) ? a : c : (b > c) ? b : c);

System.***out***.println("Max of three numbers = "

+ result);

}

}

1. **Bitwise Operators:** These operators are used to perform the manipulation of individual bits of a number. They can be used with any of the integer types. They are used when performing update and query operations of the Binary indexed trees.

* **&, Bitwise AND operator:**returns bit by bit AND of input values.
* **|, Bitwise OR operator:**returns bit by bit OR of input values.
* **^, Bitwise XOR operator:**returns bit by bit XOR of input values.
* **~, Bitwise Complement Operator:**This is a unary operator which returns the one’s complement representation of the input value, i.e., with all bits inverted.

1. **Shift Operators:** These operators are used to shift the bits of a number left or right, thereby multiplying or dividing the number by two, respectively. They can be used when we have to multiply or divide a number by two. General format-

number **shift\_op** number\_of\_places\_to\_shift;

* **<<, Left shift operator:**shifts the bits of the number to the left and fills 0 on voids left as a result. Similar effect as of multiplying the number with some power of two.
* **>>, Signed Right shift operator:**shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit depends on the sign of the initial number. Similar effect as of dividing the number with some power of two.
* **>>>, Unsigned Right shift operator:**shifts the bits of the number to the right and fills 0 on voids left as a result. The leftmost bit is set to 0.

1. **instanceof operator:** The instance of the operator is used for type checking. It can be used to test if an object is an instance of a class, a subclass, or an interface. General format-

* object **instance of** class/subclass/interface