CORE JAVA - OPERATORS

Lecture Notes

A COMPLETE GUIDE FOR BEGINNERS

Softech Solutions Inc.

www.technosolution.tech

saney.alam@techsolutions.tech

Core Java - Operators

Operators

Operators are special characters within the Java language to manipulate primitive data types. Java operators can be classified as:
Unary: Takes one argument. These operators appear before (prefix) its argument or after (postfix) its argument.
Binary: Takes two arguments. These operators appear between its arguments.
Ternary: Takes three arguments. These operators appear between its arguments.

Different types of Operators in java

Assignment Operators: **=**
Arithmetic Operators: **- + * / % ++ --**
Relational Operators : **> < >= <= == !=**
Logical Operators: **&& || &| ! ^**
Conditional Operator: **?**
Bit wise Operator: **& | ^ >> >>>**
Compound Assignment Operators: **+= -= *= /= %=**

Assignment Operators

Assignment Operator is denoted by the symbol ***"="*** and it is the most commonly used operator. It simply assigns the value on the right to the variable on the left. Syntax of the assignment operator is **<variable> = <expression>.**

Example

```
public class Assignment_Operators {
        //Normal Assignment
        static int speed = 80; // speed variable gets the value 80
        static int distance = 20; // distance variable gets the value 20
        static int time = 10; // time variable gets the value 10
        static String name = "ToolQA"; // name variable gets the value ToolQA
        static boolean isGood = true; // isGood variable gets the value true

        public static void main(String[] args) {

                System.out.println("Value stored in the speed variable is : " + speed);
                System.out.println("Value stored in the distance variable is : " + distance);
                System.out.println("Value stored in the time variable is : " + time);
                System.out.println("Value stored in the name variable is : " + name);
                System.out.println("Value stored in the isGood variable is : " + isGood);

                speed = 100; // Previous value of speed is overwritten with 100
                time = distance; // Previous value of time is overwritten with distance value
                name = "ForumsQA"; // Previous value of name is overwritten with ForumsQA
                isGood = false; // Previous value of isGood is overwritten with false
                System.out.println("Value stored in the speed variable is : " + speed);
                System.out.println("Value stored in the time variable is : " + time);
                System.out.println("Value stored in the name variable is : " + name);
                System.out.println("Value stored in the isGood variable is : " + isGood);

                //Multiple Assignments
                speed = distance = 0; // 100 (20 = 0)
                System.out.println("Value stored in the speed variable is : " + speed);

                //Illegal Assignments - Compile time errors
                speed = "ToolQA"; //String cannot be assign to integer
                name = 10; // Integer cannot be assign to String
                isGood = "ToolQa" // String cannot be assign to Boolean
                }
        }
```

Output:

```
Value stored in the speed variable is : 80
Value stored in the distance variable is : 20
Value stored in the time variable is : 10
Value stored in the name variable is : ToolQA
Value stored in the isGood variable is : true
Value stored in the speed variable is : 100
Value stored in the time variable is : 20
```

Value stored in the name variable is : ForumsQA
Value stored in the isGood variable is : false
Value stored in the speed variable is : 0

Arithmetic Operators

Arithmetic operators perform the same basic operations you would expect if you used them in mathematics. They take two operands and return the result of the mathematical calculation. There are seven important arithmetic operators available in Java:

Addition '+': This add two numbers or concatenate two strings
Subtraction '-': This subtracts right side operand from the left side operand
Multiplication '*': This multiplies two numbers
Division '/': This divides left side operand by the right side operand
Modulo '%': This divides left side operand by the right side operand and returns remainder
Increment '++': This increases the value by 1
Decrement '- -': This decreases the value by 1

Example:

```
package javaTutorials;

public class Arithmetic_Operators {

	public static void main(String[] args) {
		int a, b = 10, c = 5;
		a = b + c;
		System.out.println("Value of 'a' after '+' Arithmetic operation is " + a);
		a = b - c;
		System.out.println("Value of 'a' after '-' Arithmetic operation is " + a);
		a = b * c;
		System.out.println("Value of 'a' after '*' Arithmetic operation is " + a);
		a = b / c;
		System.out.println("Value of 'a' after '/' Arithmetic operation is " + a);
		a = b % c;
		System.out.println("Value of 'a' after '%' Arithmetic operation is " + a);
		b++;
		System.out.println("Value of 'b' after '++' Arithmetic operation is " + b);
		c--;
		System.out.println("Value of 'c' after '--' Arithmetic operation is " + c);
	}

}
```

This will produce the following result –

Output

Value of 'a' after '+' Arithmetic operation is 15
Value of 'a' after '-'Arithmetic operation is 5
Value of 'a' after '*' Arithmetic operation is 50
Value of 'a' after '/' Arithmetic operation is 2
Value of 'a' after '%' Arithmetic operation is 0
Value of 'b' after '++' Arithmetic operation is 11
Value of 'c' after '–'Arithmetic operation is 4

Relational Operators

Relational Operators are used to determine the comparison between two or more objects. These operators always return the boolean value either true or false when used in an expression. In java we have six different relational operators:

Greater than '>' : This checks if the value of left operand is greater than value of right operand

Less than '<' : This checks if the value of left operand is less than the value of right operand

Greater than or Equal to '>=' : This checks if the value of left operand is greater than or equal to the value of right operand

Less than or Equal to '<=' : This checks if the value of left operand is less than or equal to the value of right operand

Equal '==' : This checks if the value of both operands are equal

Not Equal '!=' : This checks if the value of two operands are not equal

Example

```
package javaTutorials;

public class Relational_Operators {

	public static void main(String[] args) {
			int Ten = 10;
			int Twenty = 20;
			int Thirty = 30;

			System.out.println("<<<<<< GREATER THAN OPERATOR >>>>>>");
			System.out.println(" Ten > Twenty ==> " + (Ten > Twenty)); //false
			System.out.println(" Twenty > Ten ==> " + (Twenty > Ten)); //true
			System.out.println(" Thirty > Twenty ==> " + (Thirty > Twenty)); //true
```

```
		System.out.println("<<<<<< GREATER THAN OR EQUAL TO OPERATOR >>>>>>");
			System.out.println(" Ten >= Twenty ==> " + (Ten >= Twenty)); //false
			System.out.println(" Twenty >= Ten ==> " + (Twenty >= Ten)); //true
			System.out.println(" Thirty >= Twenty ==> " + (Thirty >= Twenty)); //true

			System.out.println("<<<<<< LESS THAN OPERATOR >>>>>>");
			System.out.println(" Ten < Twenty ==> " + (Ten < Twenty)); //true
			System.out.println(" Twenty < Ten ==> " + (Twenty < Ten)); //false
			System.out.println(" Thirty < Twenty ==> " + (Thirty < Twenty)); //false

			//less than or equal to
			System.out.println("<<<<<< LESS THAN OR EQUAL TO OPERATOR
>>>>>>");
			System.out.println(" Ten <= Twenty ==> " + (Ten <= Twenty)); //true
			System.out.println(" Twenty <= Ten ==> " + (Twenty <= Ten)); //false
			System.out.println(" Thirty <= Twenty ==> " + (Thirty <= Twenty)); //false

			//equal to
			System.out.println("<<<<<< EQUAL TO OPERATOR >>>>>>");
			System.out.println(" Ten == Twenty ==> " + (Ten == Twenty)); //false
			System.out.println(" Thirty == Twenty + Ten ==> " + (Thirty == Twenty+Ten));
//true

			//not equal to
			System.out.println("<<<<<< NOT EQUAL TO OPERATOR >>>>>>");
			System.out.println(" Ten != Twenty ==> " + (Ten != Twenty)); //true
			System.out.println(" Thirty != Twenty + Ten ==> " + (Thirty != Twenty + Ten));
//false
		}
}
```

This will produce the following result –

Output

```
<<<<<< GREATER THAN OPERATOR >>>>>>
Ten > Twenty ==> false
Twenty > Ten ==> true
Thirty > Twenty ==> true
<<<<<< GREATER THAN OR EQUAL TO OPERATOR >>>>>>
Ten >= Twenty ==> false
Twenty >= Ten ==> true
Thirty >= Twenty ==> true
<<<<<< LESS THAN OPERATOR >>>>>>
Ten < Twenty ==> true
Twenty < Ten ==> false
Thirty < Twenty ==> false
<<<<<< LESS THAN OR EQUAL TO OPERATOR >>>>>>
Ten <= Twenty ==> true
Twenty <= Ten ==> false
Thirty <= Twenty ==> false
<<<<<< EQUAL TO OPERATOR >>>>>>
Ten == Twenty ==> false
Thirty == Twenty + Ten ==> true
<<<<<< NOT EQUAL TO OPERATOR >>>>>>
Ten != Twenty ==> true
Thirty != Twenty + Ten ==> false
```

Logical Operators

Logical operators return a **true** or **false** value based on the state of the Variables. Each argument to a logical operator must be a ***boolean*** data type, and the result is always a ***boolean*** data type. Below are the three most commonly used logical operators:

And Operator '&&': This returns true if the output of both the operands are true
OR Operator '||': This returns true if the output of either operands are true
NOT Operator '!': This invert the state of the condition

Example:

```
package javaTutorials;

public class Logical_Operators {

	public static void main(String[] args) {
		boolean Output_1 = true;
		boolean Output_2 = false;
		System.out.println("Check if both the boolean variables are true : " + (Output_1 &&
Output_2));
		System.out.println("Check if even one of the boolean varibale is true : " + (Output_1 ||
Output_2));
		System.out.println("Change the state of the Output_1 to false : " + (!Output_1));

	}

}
```

Output

Check if both the boolean variables are true: false
Check if even one of the boolean variable is true: true
Change the state of the Output_1 to false: false

Conditional Operator

The **conditional operator** is the only operator that takes three arguments in Java. The ***conditional operator*** is equivalent to **if-else** statement. It 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. The operator is written as:
variable = (expression)? Value if true: value if false

```
package javaTutorials;

public class Conditional_Operators {

	public static void main(String[] args) {
		int Ten = 10;
		int Twenty = 20;
		int Thirty = 30;
	    boolean bValue;
	    int iValue;

	    bValue = (Thirty == Twenty + Ten) ? true: false;
	    System.out.println( "The boolean value of the variable 'bValue' is : " +  bValue ); //true

	    iValue = ((Thirty == Twenty + Ten)) ? 50: 100;
	    System.out.println( "The int Value of the variable iValue is : " + iValue ); //50

	    //This is a use of Not Logical Operator
	    iValue = (!(Thirty == Twenty + Ten)) ? 50: 100;
	    System.out.println( "The int Value of the variable iValue is : " + iValue ); //100
	}

}
```

Output

The boolean value of the variable 'bValue' is: true

The int Value of the variable iValue is: 50

The int Value of the variable iValue is: 100