Course code : **CSE1004**

Course title : **Problem Solving using Java**

Java – Operators

Objectives

This session will give the knowledge about

- Operators in Java

Operators

- Java provides a set of operators to manipulate operations.
- Types of operators in java are,
  - Arithmetic Operators
  - Unary Operator
  - Relational Operators
  - Logical Operators
  - Simple Assignment Operator
  - Bitwise Operators

Arithmetic Operators

The following table lists the arithmetic operators

Operator	Description	Example
+	Addition	A + B
-	Subtraction	A - B
*	Multiplication	A * B
/	Division	A/B
%	Modulus	A%B

Arithmetic Operators - Example

```
/* Example to understand Arithmetic operator */
class Sample
{
   public static void main(String[ ] args)
   {
         int a = 10;
         int b = 3;
         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("a % b = " + (a % b) );
   }
}
```

Unary Operators

- The following table lists the unary operators

Operator	Description	Example
+	Unary plus operator	+A
-	Unary minus operator	-A
++	Increment operator	++A or A++
--	Decrement operator	--A or A--

Unary Operator - Example

```
/* Example to understand Unary operator */

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

Quiz

What will be the result, if we try to compile and execute the following code?

```
class Test
{
    public static void main(String [ ] args)
    {
        int x=10;
        int y=5;
        System.out.println(++x+(++y));
  }
}
```

Relational Operators

- The following table lists the relational operators

Operator	Description	Example
==	Two values are checked, and if equal, then the condition becomes true	(A == B)
!=	Two values are checked to determine whether they are equal or not, and if not equal, then the condition becomes true	(A != B)
>	Two values are checked and if the value on the left is greater than the value on the right, then the condition becomes true.	(A > B)
<	Two values are checked and if the value on the left is less than the value on the right, then the condition becomes true	(A < B)
>=	Two values are checked and if the value on the left is greater than equal to the value on the right, then the condition becomes true	(A >= B)
<=	Two values are checked and if the value on the left is less than equal to the value on the right, then the condition becomes true	(A <= B)

Comparing Strings

For comparing Strings instead of using == operator, use equals Method

```
String s1=new String("hai");
String s2="hai";
String s3="hai";

System.out.println(s2==s3);
System.out.println(s2.equals(s3));
System.out.println(s1==s2);
System.out.println(s1.equals(s2));
```

Relational Operators - Example

```
/* Example to understand      Relational operator */
class Sample
{
   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) );
   }
}
```

Logical Operators

- The following table lists the logical operators

Operator	Description	Example
&&	This is known as Logical AND & it combines two variables or expressions and if and only if both the operands are true, then it will return true	(A && B) is false
||	This is known as Logical OR & it combines two variables or expressions and if either one is true or both the operands are true, then it will return true	(A || B) is true
!	Called Logical NOT Operator. It reverses the value of a Boolean expression	!(A && B) is true

Logical Operators - Example

```
/* Example to understand logical operator */

class Sample
{
    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) );
    }
}
```

Simple Assignment Operator

= Simple assignment operator

Which assigns right hand side value to left hand side variable

Ex:

```
int a;
a = 10;
```

Shift Operators << and >>

The shift operators(<< and >>) shift the bits of a number to the left or right, resulting in a new number.

They are used only on integral numbers(and not
on floating point numbers, i.e. decimals).

The right shift operator(>>) is used to divide a number in the multiples of 2, while the left shift operator(<<) is used to multiply a number in the multiples of 2.

Right Shift Operator >>

Let us understand the use of right shift operator with the following example :

```
int x = 16;
x = x >> 3;
```

When we apply the right shift operator >>, the value gets divided by 2 to the power of number specified after the operator. In this case, we have 3 as the value after the right shift operator. So, 16 will be divided by the value 2 to the power of 3, which is 8.

The result is 2.

Right Shift Operator >>

When we represent 16 in binary form, we will get the following binary value :

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

When we apply >> which is the right shift operator, the bit represented by 1 moves by 3 positions to the right(represented by the number after the right shift operator).

After shifting the binary digit 1, we will get :

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

x = 2

Right Shift Operator >> - Demo

```
class ShiftExample1
{
   public static void main(String[] args)
   {
        int x = 16;
        System.out.println("The original value of x is "+x);
        x = x >> 3;
        System.out.println("After using >> 3, the new value is "+x);
   }
}
```

Left Shift Operator <<

Let us understand the use of left shift operator with the following example :

```
int x = 8;
x = x << 4;
```

When we apply the left shift operator <<, the value gets multiplied by 2 to the power of number specified after the operator. In this case, we have 4 as the value after the left shift operator. So, 8 will be multiplied by the value 2 to the power of 4, which is 16.

The result is 128.

Left Shift Operator <<

When we represent 8 in binary form, we will get the following binary value :

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

When we apply << which is the left shift operator, the bit represented by 1 moves by 4 positions to the left (represented by the number after the right shift operator).

After shifting the binary digit 1, we will get :

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

X=128

Left Shift Operator << - Demo

```
class ShiftExample2
{
   public static void main(String[] args)
   {
        int x =8;
        System.out.println("The original value of x is "+x);
        x = x << 4;
        System.out.println("After using << 4, the new value is "+x);
   }
}
```

Bitwise operators

The bitwise operators take two bit numbers, use OR/AND to determine the result on a bit by bit basis.

The 3 bitwise operators are :

- & (which is the bitwise AND)
- | (which is the bitwise inclusive OR)
- ^ (which is the bitwise exclusive OR)

Bitwise Operator Demo

```
class BitwiseExample1
{
public static void main(String[] args)
{
int x = 7;
int y = 9;
System.out.println(x & y);
System.out.println(x | y);
System.out.println(x ^ y);
}
}
```

Output:

1

15

14

Reason:

7 = 0 1 1 1	7 = 0 1 1 1	7 = 0 1 1 1
9 = 1 0 0 1	9 = 1 0 0 1	9 = 1 0 0 1
& = 0 0 0 1	| = 1 1 1 1	^ = 1 1 1 0

Quiz

What will be the result, if we try to compile and execute the following code?

```
public class Sample
{
    public static void main()
    {
        int i_val = 10,
        j_val = 20;
        boolean chk;
        chk = i_val < j_val;
        System.out.println("chk value: "+chk);
    }
}
```

Summary

We have discussed about

- Operators in Java