Operators in java

- Arithmetic Operators
- Increment Decrement Operators
- Relational Operators
- Bitwise Operators
- Logical Operators
- Assignment Operators
- Misc (Special)Operators

Arithmetic Operators

Operator	Example
+	A + B
-	A - B
*	A * B
/	B / A
%	B % A

Increment/Decrement Operators

Operator	Description
++	Increment - Increase the value of operand by 1
--	Decrement - Decrease the value of operand by 1

```
int a = 10;
int d = 25;
System.out.println("a++ = " + (a++) );
System.out.println("b-- = " + (a--) );
// Check the difference in d++ and ++d
System.out.println("++d = " + (++d) );
Output:- a++ = 10
b-- = 11
d++ = 25
++d = 27
```

Relational Operators

Assume variable A holds 10 and variable B holds 20 then:

Operator	Description	Example
==	Checks if the value of two operands are equal or not, if yes then condition becomes true.	(A == B) is not true.
!=	Checks if the value of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true.

Bitwise Operator

- 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 perform bit by bit operation. Assume if

  a = 60;

  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

Bitwise Operators

&	Sets each bit to 1 if both bits are 1
|	Sets each bit to 1 if one of two bits is 1
^	Sets each bit to 1 if only one of two bits is 1
~	Inverts all the bits
<<	Shifts left by pushing zeros in from the right and let the leftmost bits fall off
>>	Shifts right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Operation		Result
5 &1	0101&0001	0001
5 | 1	0101 | 0001	0101
~5	0101	1010
5<<1	0101	1010
5>>2	0101	0001

Logical Operators

Assume boolean variables A holds true and variable B holds false then

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non zero then then condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands are non zero then then condition becomes true.	(A || B) is true.
!	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.	!(A && B) is true.

Assignment Operators

Operator	Example
=	C = A + B will assigne value of A + B into C
+=	C += A is equivalent to C = C + A
-=	C -= A is equivalent to C = C- A
*=	C *= A is equivalent to C = C * A
/=	C /= A is equivalent to C = C/ A

Assignment Operators

Operator	Description	Example
%=	Modulus AND assignment operator, It takes modulus using two operands and assign the result to left operand	C %= A is equivalent to C = C % A
<<=	Left shift AND assignment operator	C <<= 2 is same as C = C << 2
>>=	Right shift AND assignment operator	C >>= 2 is same as C = C >> 2
&=	Bitwise AND assignment operator	C &= 2 is same as C = C & 2
^=	bitwise exclusive OR and assignment operator	C ^= 2 is same as C = C ^ 2
|=	bitwise inclusive OR and assignment operator	C |= 2 is same as C = C | 2

Misc Operators

- **Conditional Operator (? :):**
- Conditional operator is also known as the ternary operator.
- This operator consists of three operands and is used to evaluate boolean expressions.
- variable x = (expression) ? value if true : value if false
- Following is the 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 );
}
}
```

This would produce 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).
- instanceOf operator is wriiten 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.
- Following is the example:
- String name = = 'James';
- boolean result = name instanceOf String;

  // This will return true since name is type of String

instanceOf operator

- This operator will still return true if the object being compared is the assignment compatible with the type on the right.
- Following is one more example:

```
class Vehicle {}
public class Car extends Vehicle
 {
public static void main(String args[])
{
 Vehicle a = new Car();
 boolean result = a instanceof Car;       System.out.println( result);
 }
 }
```

This would produce following result: true