Java Operators and Expressions

- An operator is a function that has a specific symbol
- Java operators can be classified as
  1. Arithmetic operators (+, -, *, /)
  2. relational operators (=<, >=, ==, !=)
  3. logical operators (&&, ll, !)
  4. assignment operator (=)
  5. increment and decrement (++, - -)
  6. conditional operator (:?)
  7. bitwise operator (>>, <<)
  8. special operators

(1) Arithmetic operators

Operator	**Meaning**
+	addition or unary plus
-	subtraction or unary minus
*	multiplication
/	division
%	modulo division

eg:-

x = y/2;
x, y are variables
2 is a literal
=, / are operators

eg:-

```
class Arithmetic{
        public static void main (String args []){
                int x = 17;
                int y = 15;
                System.out.println("x =" + x);
                System.out.println("y =" + y);
                System.out.println("x + y =" + (x+y));
                System.out.println("x * y =" + (x*y));
                System.out.println("x % y =" + (x%y));
                }
}
```

what is the output of this program? try this

(2) Relational operators

- use to compare variables
- returns "TRUE" or "FALSE"

Operator	**Meaning**
<	less than
<=	less than or equal
>	greater than
>=	greater than or equal
==	equal to
!=	not equal

eg:-

10 < 20	TRUE	
x = 10, y = 20	y < x	FALSE
	x < 30	FALSE
	x == y	FALSE
	x != y	TRUE

Note:
When arithmetic expression is used on either sides of the relational operator, the arithmetic expression will be evaluated first.

eg:-

x = 10, y = 20, z = 30;
z < (x + y)

this is how it is going to be evaluated: 30 < (10 + 20) = 30 < 30 and hence FALSE

eg:-

```
class Relational{
        public static void main (String args []){
                float a =  15.0f, b = 20.7f, c = 10.0f;
                System.out.println("a <  b  is" + (a < b));
                System.out.println("a >  b  is" + (a > b));
                }
        }
```

what is the output of this program? try this

(3) Logical Operators

Operator	**Meaning**
&&	logical AND
ll	logical OR
!	logical NOT

eg:-

(a > b) && (x == 10)

if (a > b) is TRUE and (x ==10) is TRUE the whole statement will be TRUE

Truth table:

Operand 1	**Operand 2**	**&&**	**ll**
FALSE	FALSE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE
TRUE	FALSE	FALSE	TRUE
TRUE	TRUE	TRUE	TRUE

eg:-

```
if ((age > 24) && (salary > 1000)){
        …
        }
```

(4) Assignment Operator

used to assign values of an expression to a variable

eg:- n1 = n2 = n3 = n4 = (10+30);
first evaluate the (10+30) expression and the value is assigned to n4, then the value os n4 is assigned to n3, then the value of n3 is assigned to n2 and so on..

Note: Right side of the assignment operator is evaluated and assign the value towards left (right to left)

Shorthand assignment:

eg:- total += num1 + num2;
when expanded: total = total + (num1 + num2)

Advantages of using shorthand form:
1. left hand side need not to be repeated
2. easier to read
3. easier to compile and hence more efficient in code

(5) Increment and Decrement Operator

I have discussed this in the class…..

Try this program, look at the output and understand why it is happening like that

```
public class MyProgram{
	public static void main(String[] args) {
		int m = 10, n =20;
		System.out.println("m = "+m);
		System.out.println("n = "+n);
		System.out.println("++m = "+(++m)); //look at here!
		System.out.println("n++ = "+(n++)); //look at here!
		System.out.println("m = "+m);
		System.out.println("n = "+n);
	}
}
```

(6) Conditional Operator

- This is a ternary (it has three operands) operator, that assigns a value to a variable based on a condition.
- general form is
  var = <expression 1> ? <expression 2> : <expressoin 3>;

- order of execution is
  if <expression 1> is true, then the <expression 2> get executed and become the value of the variable var, otherwise <expression 3> get executed and become the value of the variable var.

eg:-

```
int a = 10, b = 20;
int x = (a>b) ? a : b;
```

this is same as the

```
int a = 10, b = 20, x;
if (a > b){
        x = a;
        }
else
        x = b;
```

eg:-

```
int a = 10, b = 20;
int x = (a>b) ? a + 5 : b - 5;
```

eg:-

```
int a = 10, b = 20;
boolean x = (a>b) ? a + 5 : b - 5;
```

(7) Bitwise Operator

- manipulate date of values of bit level
- this operator can be used to (i) test the bit (ii) shift to left (iii) shift to right

Operator	Meaning
&	bitwise AND
!	bitwise NOT
^	exclusive OR
l	bitwise OR
~	complement
<<	shift left
>>	shift right
>>>	shift right with zero fill

eg:-

```
public class MyProgram{
        public static void main(String[] args) {
        int a = 5, b = 6; //bitwise 101 and 110
        System.out.println("a = "+a);
        System.out.println("b = "+b);
        System.out.println("a & b = "+(a&b)); //101 AND 110 = 100 which is 4
        System.out.println("a l b = "+(alb)); //101 OR 110 = 111 which is 7
        System.out.println("a ^ b = "+(a^b)); //101 XOR 110 = 011 which is 3
        }
}
```

eg:-

```
public class MyProgram{
        public static void main(String[] args) {
        int a = 7; // bitwise 111
```

```
        System.out.println("a = "+a);
        System.out.println("a>>>2 = "+(a>>>2)); //111 becomes 001 that is 1
        System.out.println("a<<1 = "+(a<<1)); //111 becomes 1110 that is 14
        System.out.println("a>>>1 = "+(a>>>1)); //111 becomes 011 that is 3
        }
}
```

(8) Special Operators

1. instanceOf operator

- returns true or false
- if your object on the left is an instance of the class given on the right, then returns true

eg:-

```
class Student;
class Teacher;

Student s = new Student ();
if (s instanceOf Student){
        System.out.print(“True”);
        }
```

- This is used to make sure that the object s is an object of Student class when calling a method.

2. Dot (.) operator

- used to access instance variables, constants and operators

eg:-

```
 class Student{
        int age;
        public int getAge(){
              return this.age; \\ . operator is used to access the age variable of an object of this
(Student) class

                }
        }
```

Student s1 = new Student();
System.out.println("Age is" + s1.getAge()); \\ . operator is used to access the getAge method of the object s1

Expressions

a combination of variables, constants and operators arranged according to the syntax of the language.

General order of evaluation

1. increment, decrement
2. arithmetic
3. comparison
4. logical
5. assignment

Order of persistence

1. ., [], ()
2. ++, - -, !, ~
3. *,/,%
4. +,-
5. >, <. <=, >=
6. == ,!=
7. &&, ll, !
8. &, ^, l
9. =

Note: it is possible to change the order of execution by using parentheses ().

eg:-
result = 3 + 2* 3 which gives result = 9

using parentheses
result = (3+2)*3 which gives result = 15