Operators in Java

Module-1

Dr.M. Sivakumar
Professor & Trainer / CSE
Saveetha School of Engineering, SIMATS, Chennai-602 105
sivakumarm.sse@saveetha.com
+91-9500600868

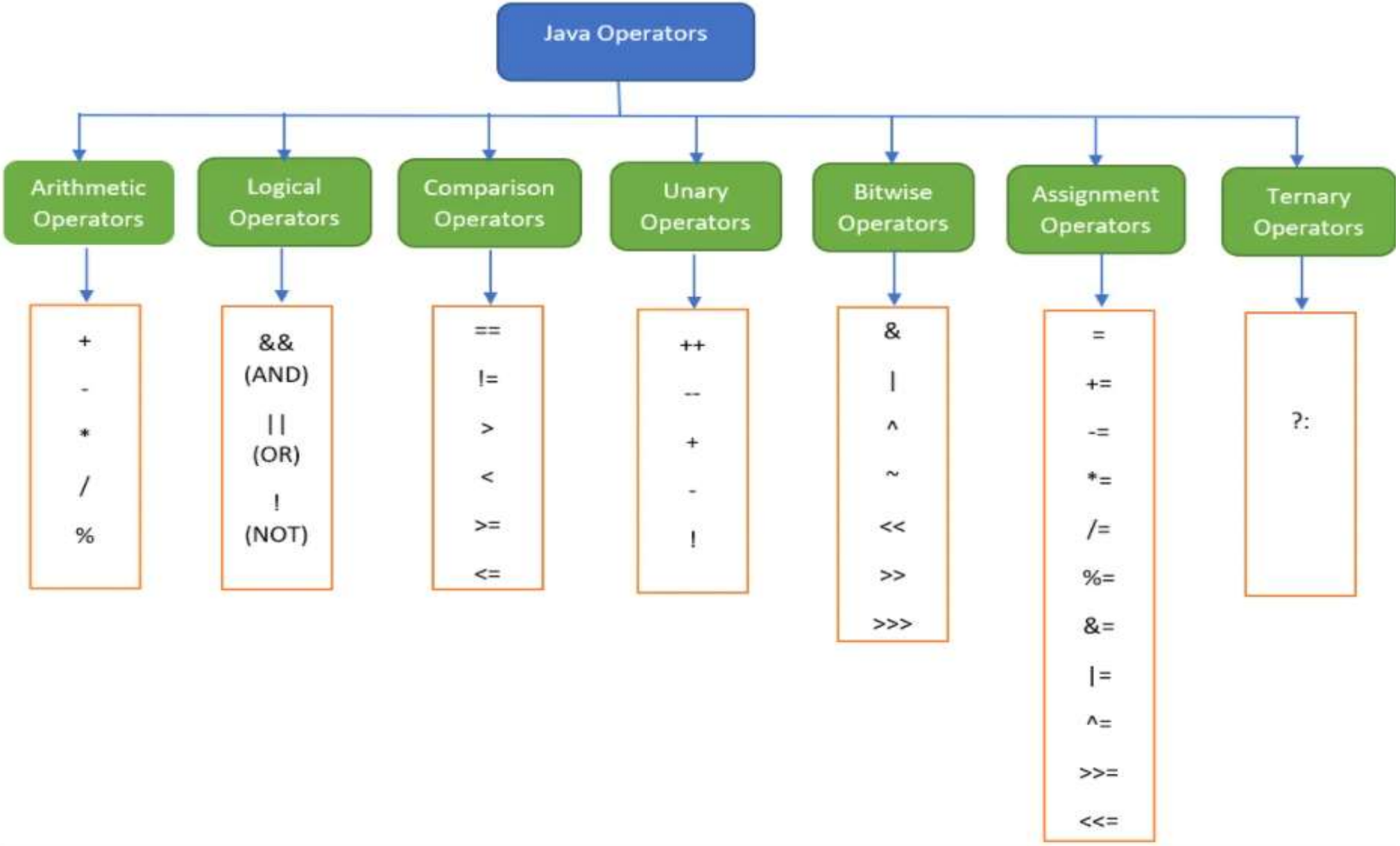
Operators

- **Operator** is a symbol that tells the computer to perform some action on constants and variables called operands

```
int a = 100, b = 50;
int c = a + b;    // a, b are operands
```

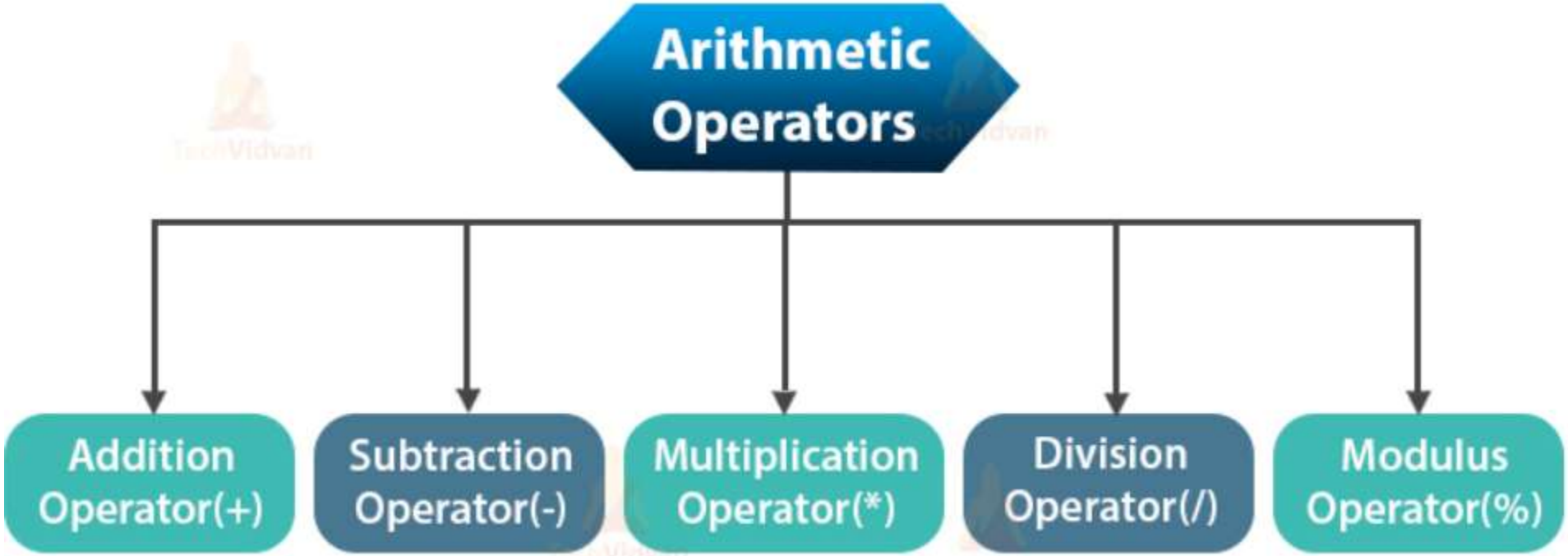
- It follows priority for evaluation

Operators


Java Operators
Arithmetic Operators
+
-
*
/
%
Logical Operators
&&
(AND)
||
(OR)
!
(NOT)
Comparison Operators
==
!=
>
<
>=
<=
Unary Operators
++
--
+
-
!
Bitwise Operators
&
|
^
~
<<
>>
>>>
Assignment Operators
=
+=
-=
*=
/=
%=
&=
|=
^=
>>=
<<=
Ternary Operators
?:

1.Arithmetic Operators

- All the arithmetic operators requires two operands to perform any operation



integer type (int, short, byte, long, double) or float type (float and double).

1.Arithmetic Operators

For division(/) operator

If we divide two integers, the result will be an integer.

5/2=2 (Not 2.5)

To get 2.5, at least one of the numerator or denominator must have a decimal(float) value.

5.0/2=2.5 or 5/2.0=2.5 or 5.0/2.0=2.5

For modulo(%) operator

If we apply two integers, the result will be a remainder.

5%2=1 2%5 = 2 [If Numerator is smaller]

To get signature in output, modulo takes sign of numerator

-5%2=-1 or 5%-2= 1 or -5%-2=-1

Modulo will not take floating-point value as an input operand

1.Arithmetic Operators

Test yourself

Example -1

```
int a= +10;
System.out.println(a);
```

Output

```
Output: 10
```

Example -2

```
int a=10;
-a;
System.out.println(a);
```

Output

```
Output: 10
```

Example -3

```
int a=10;
System.out.println(a+10);
System.out.println(a);
```

Output

```
Output: 20
        10
```

Example -4

```
int a = 15, b= -2;
System.out.println(a/b);
```

Output

```
Output: -7
```

Example -5

```
int a = 15, b= 2;
System.out.println(b%a);
```

Output

```
Output: 2
```

Example -6

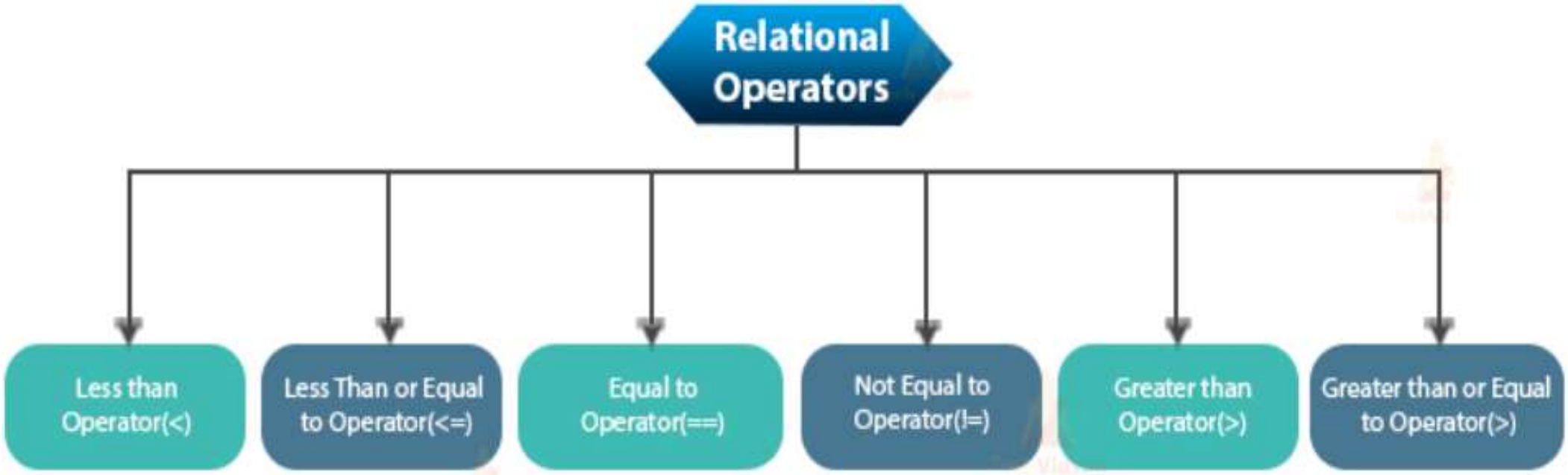
```
int a = -15,b=2;
System.out.println(a%b);
```

Output

```
Output: 0
```

1.Relational Operators

- The relational operators determine the relation among the operands.
- Java provides 6 relational operators for comparing numbers and characters. { true / false }



Integer, Character

2. Relational Operators

Test yourself

Example -1

```
int a = 5, b=10;
System.out.println(a<b<2);
```

Output

```
Output: Error
```

Example -2

```
float a = 5.6;
System.out.println(a<5.6);
```

Output

```
Output: Error
```

Example -3

```
char a = ‘A’,b=‘a’;
System.out.println(a<b);
```

Output

```
Output: true
```

Example -4

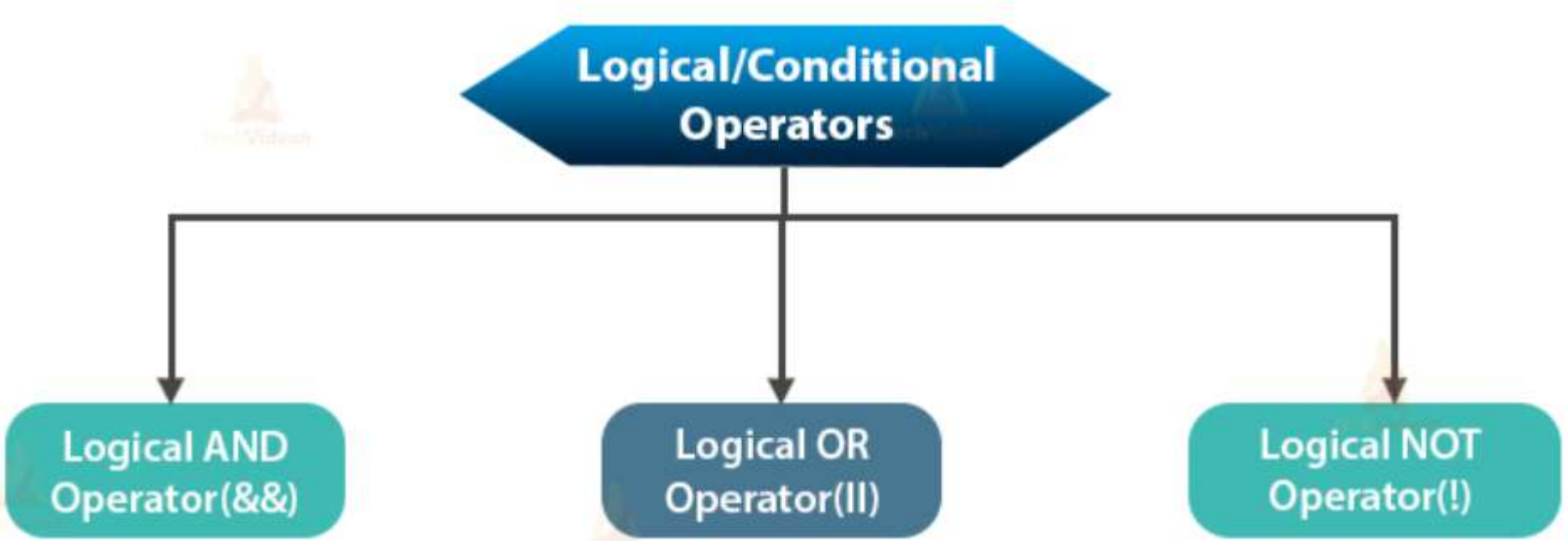
```
float a=5.6f;
System.out.println(5.6>a);
```

Output

```
Output: true
```

3.Logical Operators

- These operators are used for evaluating one or more boolean expression, for complex decision-making.{ true / false }



3.Logical Operators

Operator	Meaning	Example	Result
&&	Logical AND	(5<2)&&(5>3)	False
||	Logical OR	(5<2)||(5>3)	True
!	Logical NOT	!(5<2)	True

- It requires boolean input for evaluation rather than a value

```
System.out.println(5 && 10); // Error
```

- For logical OR, if first condition is true then it will not execute second portion.
- For logical AND, if first condition is false then it will not execute second portion.

3.Logical Operators

Test yourself

Example -1

```
int a= 10;
System.out.println(a<20 && true);
```

Output

```
Output: true
```

Example -2

```
int a=5,b=6,c=10;
System.out.println(a<b||++a<c);
System.out.println(a);
```

Output

```
Output: true
        5
```

Example -3

```
int a= 10;
System.out.println(“skcet" && true);
```

Output

```
Output: Error
```

Example -4

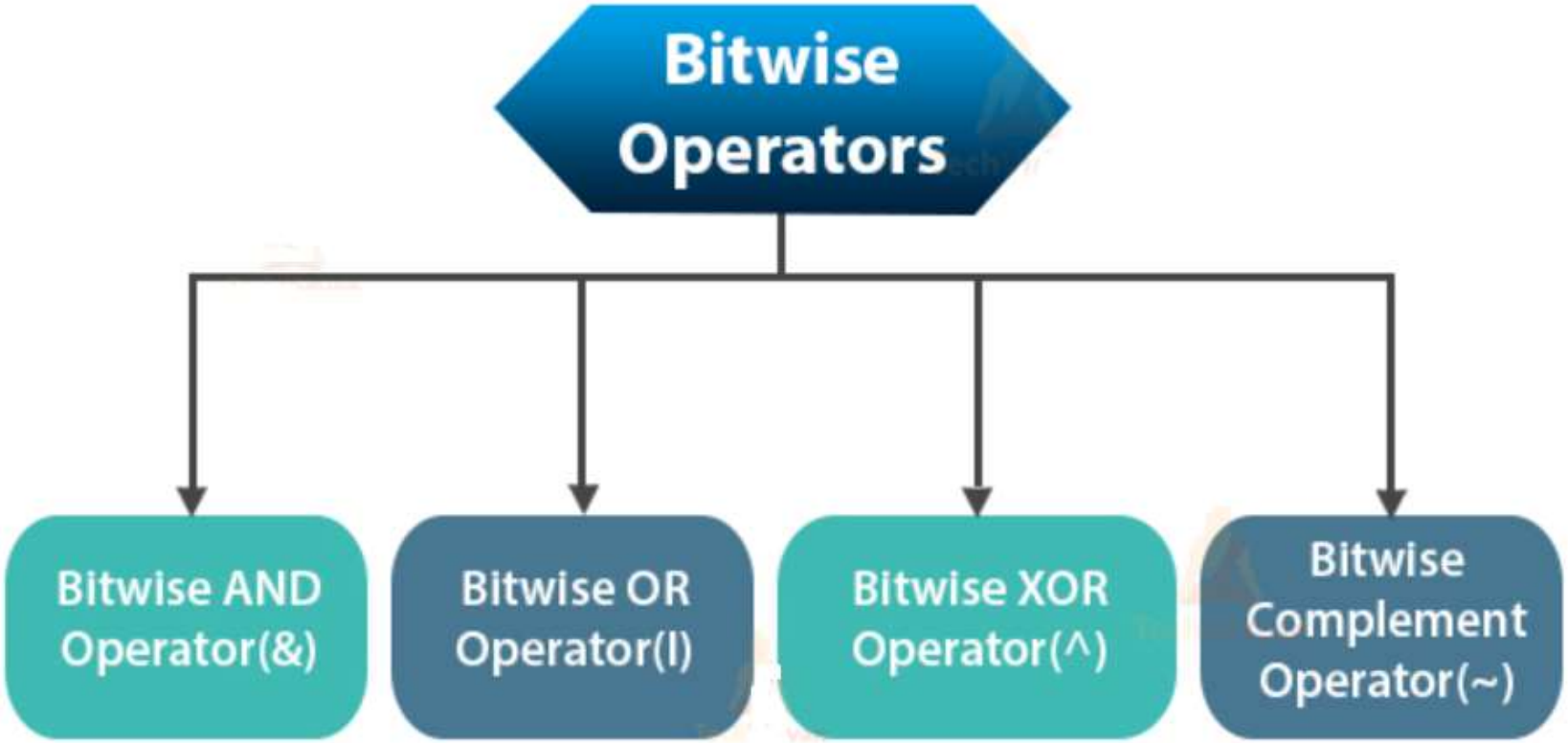
```
int a= 10;
System.out.println(!(!a));
```

Output

```
Output: 2
```

4.Bitwise Operators

- The Bitwise operators manipulate the individual bits of a number.
- It works with the integer types that is, *byte, short, int, and long types.*



4.Bitwise Operators

int a = 10, b = 2 for all examples below

Operator	Meaning	Example	Result
~	Bitwise unary NOT	~a	-11
&	Bitwise AND	a&b	2
|	Bitwise OR	a|b	10
^	Bitwise Ex-OR	a^b	8
>>	Shift right	a>>1	5
>>>	Shift right zero fill	a>>>1	5
<<	Shift left	a<<1	20
&=	Bitwise AND assignment	a &= b	2
|=	Bitwise OR assignment	a |= b	10
^=	Bitwise Ex-OR assignment	a ^= b	8
>>=	Shift right assignment	a >>= 1	5
>>>=	Shift right zero fill assignment	a >>>=1	5
<<=	Shift left assignment	a <<= 1	20

Bitwise AND (&):

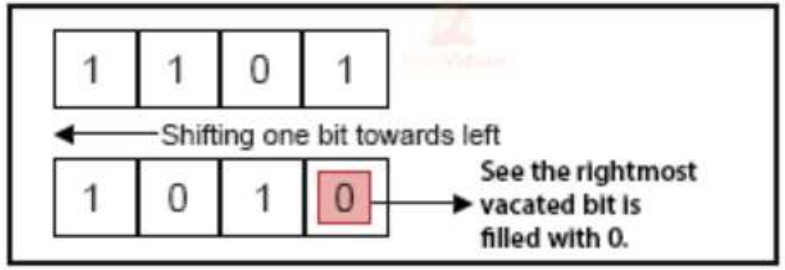
A	B	A&B
1	1	1
1	0	0
0	1	0
0	0	0

Bitwise OR (|)

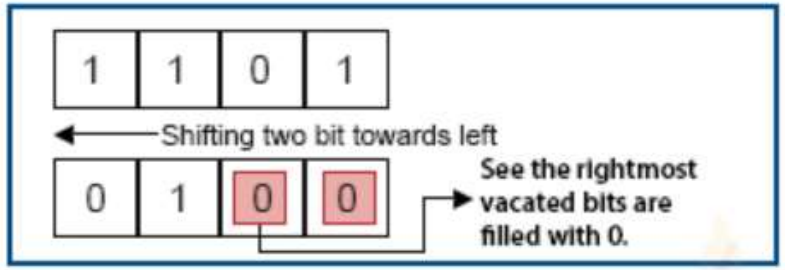
x	y	x | y
0	0	0
0	1	1
1	0	1
1	1	1

4.Bitwise Operators

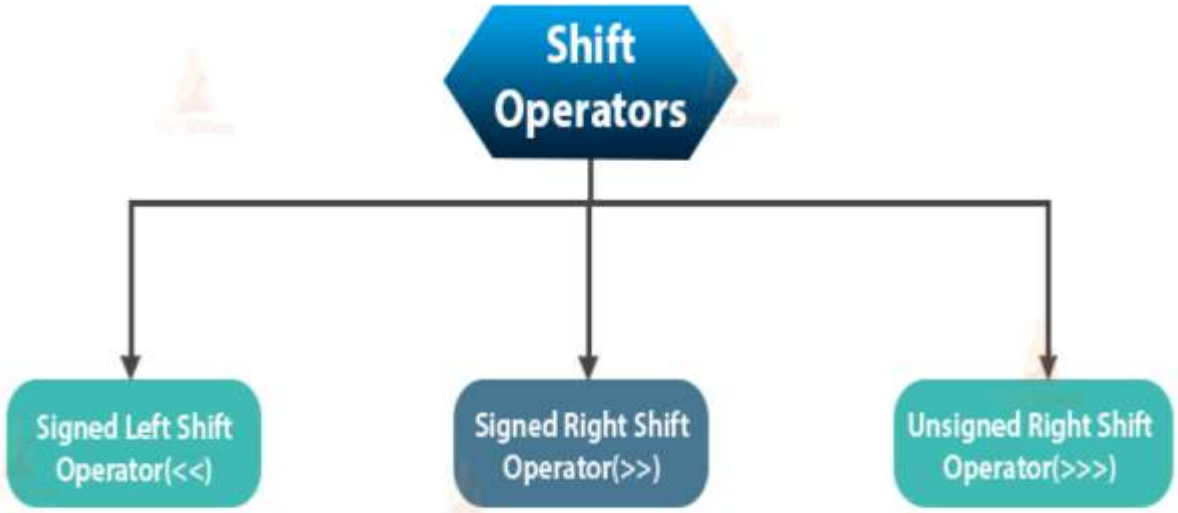
1101 << 1 becomes 1010
Left Shift



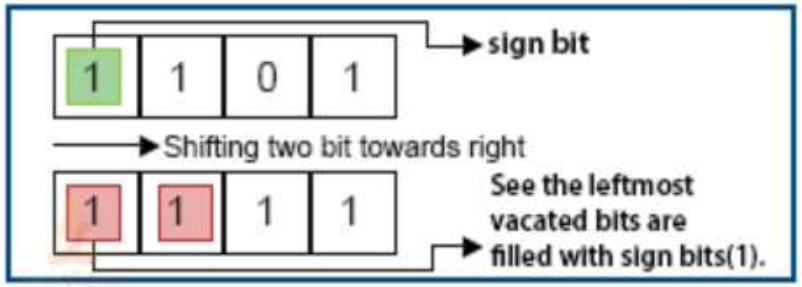
1101 << 2 becomes 0100
Left Shift



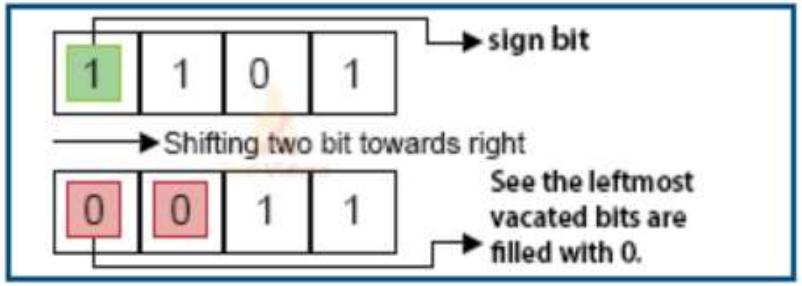
Shift Operators



1101 >> 2 becomes 1111
Signed Right Shift



1101 >>> 2 becomes 0011
Unsigned Right Shift



4.Bitwise Operators

Test yourself

Example -1

```
int a= -10;
System.out.println(~a);
```

Output

```
Output: 9
```

Example -2

```
int a=5;
System.out.println(a<<2);
System.out.println(a>>2);
```

Output

```
Output: 20
        1
```

Example -3

```
int a= 10;
System.out.println(a>>>2);
```

Output

```
Output: 2
```

Example -4

```
int a=10,b=5;
int c=a^b | ~b;
System.out.println(c);
```

Output

```
Output: -1
```

5.Additional Operators

Assignment Operators

- It is used to assign the result of an expression to a variable
- It follows the right to left associativity

short hand assignment

Operator	Example	Equivalent Expression
=	a =10	-
+=	a+=2	a= a + 2
-=	a-=2	a = a -2
*=	a*=2	a = a *2
/=	a/=2	a = a/2
%=	a%=2	a = a%2
&=	a&=2	a = a&2
|=	a|=2	a = a|2
^=	a^=2	a = a^2
<<=	a<<=2	a = a<<2
>>=	a>>=2	a = a>>2

10 = a	Not Valid
Lvalue = Rvalue	
Lvalue must be meaningful operand	
Rvalue can be anything	

5.Additional Operators

Increment /Decrement

- ++ { can be pre-increment or post-increment }
- -- { can be pre-decrement or post-decrement }

Example -1

```
int a = 5, b=10;
System.out.println(a++);
System.out.println(++b);
```

Example -2

```
int a = 5, b=10;
int c = a++ + a++;
int d = ++b + ++b;
System.out.println(c);
System.out.println(d);
```

Example -3

```
int a = 5;
int c = a++ + ++a;
System.out.println(c);
```

Example -4

```
int a = 5;
int b = a++;
System.out.println(a);
System.out.println(b);
```

5.Additional Operators

Conditional Operator

- A ternary operator is known as conditional operator
- Syntax: *exp1 ? exp2 : exp3*

Working of the ? : Operator

exp1 is evaluated first
if exp1 is true(nonzero) then
- exp2 is evaluated and its value becomes the value of the expression

If exp1 is false(zero) then
- exp3 is evaluated and its value becomes the value of the expression

Example

```
m=2, n=3;
r=(m>n) ? m : n;
```

Explanation

Value of r will be 3

Example

```
m=2, n=3;
r=(m<n) ? m : n;
```

Explanation

Value of r will be 2

5.Additional Operators

Instanceof Operator

- This is a type-check operator.
- It checks whether a particular object is the instance of a certain class or not.
- It returns true if the object is a member of the class and false if not.

```
class Simple1{
 public static void main(String args[]){
 Simple1 s=new Simple1();
 System.out.println(s instanceof Simple1); //true
 }
}
```

Practice Problems

- Program to find pass percentage of a student [Use 5 subject marks]
- Program to print value in decimal, octal and hexadecimal
- Program to get and display the mobile number of a person
- Program to check the input number positive or negative or zero using ternary operator