d="u3i1s2f5b393d4ff54017b51202306988fd70">

/div>
     </text-note-view><template is="component"
omponent-name="text-note-view"> </template>

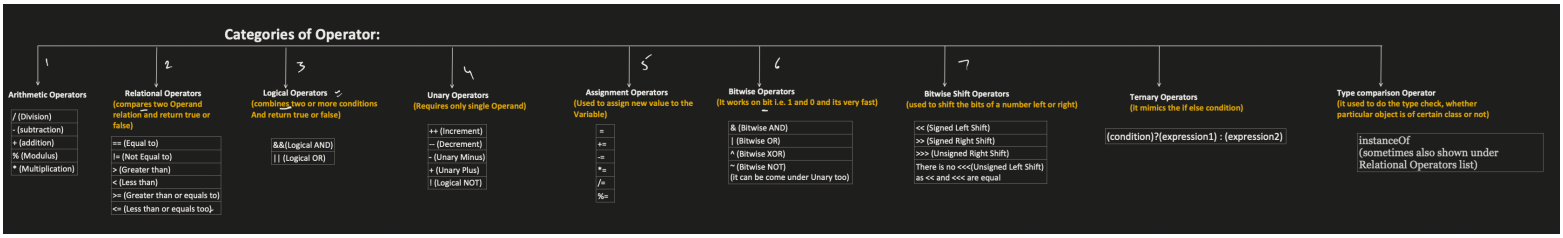
Java: Operators

"Concept && Coding" YT Video Notes

What is Operator: this indicate what actions to perform like addition, subtraction etc.

What is Operand: this indicate the items, on which action has to apply on.

What is Expression: it consist of 1 or more Operand and 0 or more Operators.



Arithmetic Operators:

```java
public class Main {

    public static void main(String[] args) {

        int division = 5 / 2;
        System.out.println(division);

        int mod = 5 % 2;
        System.out.println(mod);

        int sum = 3 + 4;
        System.out.println(sum);

        int subtract = 4-3;
        System.out.println(subtract);

        int multiply = 3 * 4;
        System.out.println(multiply);
    }

}
```

Output:

```
2
1
7
1
12
```

Relational Operators:

```java
public class Main {

    public static void main(String[] args) {

        int a = 4;
        int b = 7;
        System.out.println(a == b);
        System.out.println(a != b);
        System.out.println(a > b);
        System.out.println(a < b);
        System.out.println(a >= b);
        System.out.println(a <= b);
    }
}
```

Output:

```
false
true
false
true
false
true
```

Logical Operators:

```java
public class Main {

    public static void main(String[] args) {

        int a = 4;
        int b = 7;
        //AND operator
        System.out.println(a<3 && a!=b);
        System.out.println(a>3 && a!=b);
        //OR operator
        System.out.println(a<3 || a!=b);
        System.out.println(a>3 || a!=b);
    }
}
```

Output:

```
false
true
true
true
```

Unary Operators:

```java
public class Main {

    public static void main(String[] args) {
        int a = 5;
        boolean flag = true;

        //Increment operator
        System.out.println(a++);
        System.out.println(++a);

        //Decrement operator
        System.out.println(a--);
        System.out.println(--a);

        //Logical NOT operator
        System.out.println(!flag);

        //Unary Minus operator
        System.out.println(-a);

        //Unary Plus operator
        System.out.println(+a);
    }
}
```

Output:

```
5
7
7
5
false
-5
5
```

Assignment Operators:

```java
public class Main {

    public static void main(String[] args) {
        int a = 5;
        int variable;

        variable = a;
        System.out.println(variable);

        variable = 0;
        variable+=a;
        System.out.println(variable);

         variable-=3;
         System.out.println(variable);

         variable*=a;<
         System.out.println(variable);

         variable/=a;
         System.out.println(variable);
    }
}
```

Output:

```
5
5
2
10
2
```

Bitwise Operators:

```java
public class Main {

    public static void main(String[] args) {

        int a=4;
        int b=6;

        //Bitwise AND
        System.out.println(a & b);

        //Bitwise OR
        System.out.println(a | b);

        //Bitwise XOR
        System.out.println(a ^ b);

        //Bitwise NOT,  bitwise complement of any integer n is  -(n + 1)
        System.out.println(~a);
    }
}
```

Output:

```
4
6
2
-5
```

How does BITWISE NOT works?

int a = 4

0100 (remember its not equal to 100 as int is a signed as it represent both positive and negative value and in Java there is nothing like unsinged integer like in C++)

0100 ------> ~0100 = 1011
Now what is 1011?

1*2^-3 + 0*2^2 + 1*2^1 + 1*2^0 = -8+0+2+1 = -5

which is equivalent to -(N+1) = -(4+1) = -5

How can we confirm if -5 is 1011?

5 = 0101

To get the -5, we know that we have to find its 2nd Complement

0101 =>
1st Complement = 1010
2nd Complement = 1010 + 1 = 1011

Bitwise Shift Operators:

>> : its signed right shift
it fills the most significant bit with the sign of the number :
example 11000110 its >>
11100011 (add 1 in MSB, i.e. MSB of the original number)

01000110 its >>
00100011 (add 0 in MSB, MSB of the original number)

>>> : its Unsigned right shift
it fills the most significant bit with the 0 :
example 11000110 its >>>
01100011

01000110 its >>>
00100011

```java
public class Main {

    public static void main(String[] args) {

        int a=4;

        //left shift
        System.out.println(a<<1);
        System.out.println(a<<2);

        //right shift
        System.out.println(a>>1);
        System.out.println(a>>2);
    }
}
```

Output:

```
8
16
2
1
```

Ternary Operators:

```java
public class Main {

    public static void main(String[] args) {

        int a=4;
        int b=5;

        int maxValue = (a>b) ? a : b;
        System.out.println(maxValue);
    }
}
```

Output:

```
5
```

Type comparison (*instanceOf*) Operator:

```java
public class ParentClass {
}
```

```java
public class ChildClass1 extends ParentClass{
}
```

```java
public class ChildClass2 extends ParentClass{
}
```

```java
public class Main {

    public static void main(String[] args) {

        ParentClass obj = new ChildClass2();
        System.out.println( obj instanceof ChildClass2);
        System.out.println(obj instanceof ChildClass1);

        ChildClass1 childObj = new ChildClass1();
        System.out.println( childObj instanceof ParentClass);

        String val = "hello";
        System.out.println( val instanceof String);

        Object unknownObject = new RandomClass();
        System.out.println( unknownObject instanceof ChildClass2);
    }
}
```

Output:

```
true
false
true
true
false
```

Operator Precedence:

Associativity: if 2 operators have the same precedence, then its evaluated based on its Associativity (Left to Right or Right to Left)

Operators	Precedence	Associativity
Parentheses	(), []	Left to right
Unary: Postfix	expr++ , expr--	Left to right
Unary: Prefix	++expr, --expr, +expr, -expr, ~, !	Right to Left
Multiplicative	*, /, %	Left to right
Additive	+, -	Left to right
Bitwise Shift	<<, >>, >>>	Left to right
Relational	<, >, <=, >=, instanceOf	Left to right
equality	==, !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Ternary	?:	Right to Left
Assignment	=, +=, -=, *=, /=, %=, &=, ^=, |=, <<=, >>=, >>>=	Right to Left