



Lesson 4

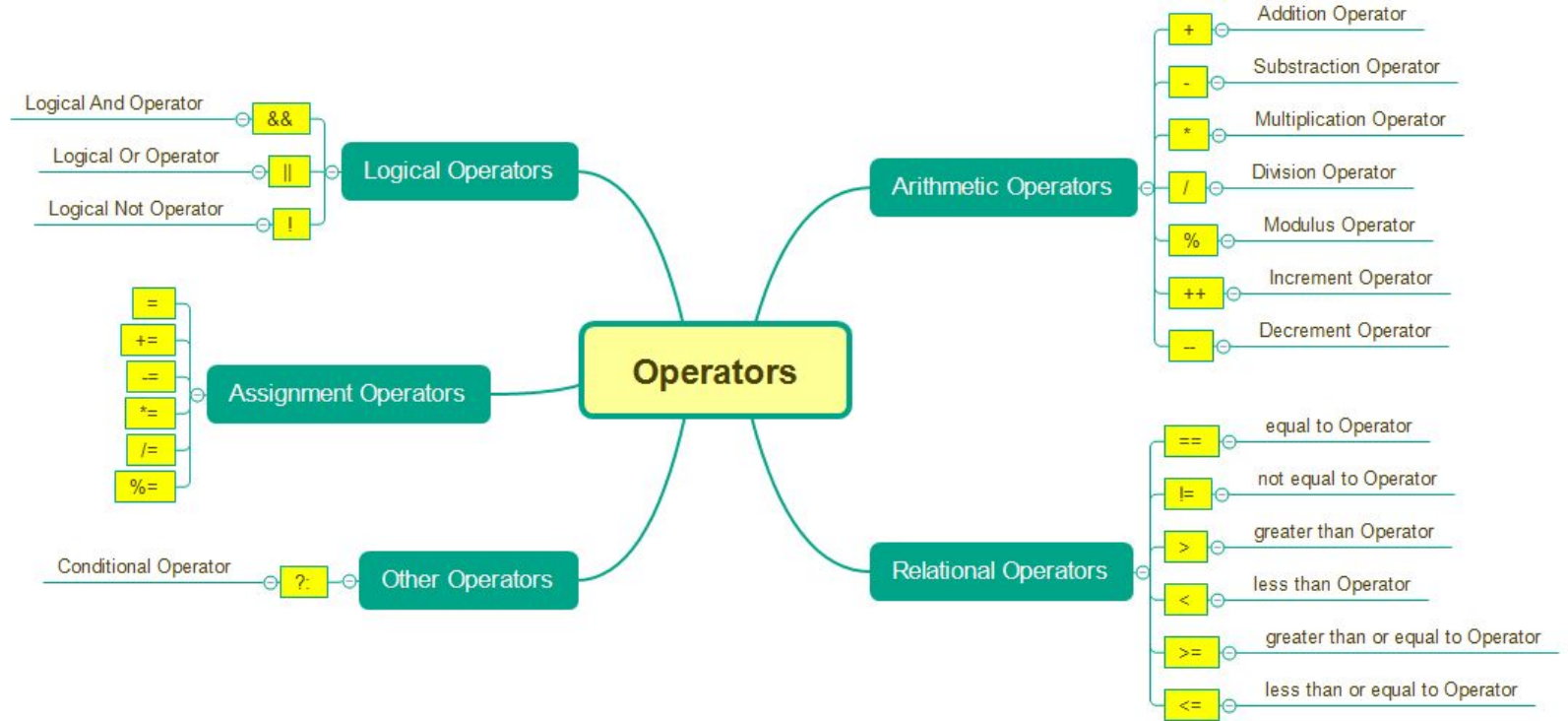
03.02.2025

```
public class Task1 {  
    public static void main(String[] args) {  
        int a = 256;  
        byte b = a;  
        System.out.println(b);  
    }  
}
```

```
public class Task2 {  
    public static void main(String[] args) {  
        double a = 100.0;  
        double b = 100.5;  
        System.out.println((int) a == (int) b);  
    }  
}
```

```
public class Task3 {  
    public static void main(String[] args) {  
        Double a = new Double(1.0);  
        Double b = new Double(1.0);  
        System.out.println(a == b);  
    }  
}
```

```
public class Task4 {  
    public static void main(String[] args) {  
        long bigNumber = 2_147_483_648;  
        System.out.println(bigNumber);  
    }  
}
```





Інкремент та декремент

У програмуванні дуже часто доводиться виконувати операції, коли:

змінна має збільшитися на одиницю $\rightarrow +1$

змінна має зменшитися на одиницю $\rightarrow -1$

Тому вигадали окремі операції зі змінними, які називаються інкремент та декремент.


Інкремент – відповідає за збільшення змінної на одиницю. Позначається як $++$. Наприклад, якщо у нас є змінна i і ми до неї застосуємо інкремент, тоді це буде записано як $i++$. А це означає, що значення змінної i має бути збільшено на 1

Декремент – відповідає за зменшення змінної на одиницю. Позначається як $--$. Наприклад, якщо у нас є змінна n і ми до неї застосуємо декремент, тоді це буде записано як $n--$. А це означає, що значення змінної n має бути зменшено на 1.

Тож у чому ж відмінність між постфіксною та префіксною формами?

У постфікській формі: спочатку використовується старе значення у обчисленнях далі у наступних обчисленнях використовується вже нове значення

У префікській формі: відразу використовується нове значення у обчисленнях



Operator	Result
&	Logical AND
	Logical OR
^	Logical XOR (exclusive OR)
	Short-circuit OR
&&	Short-circuit AND
!	Logical unary NOT
&=	AND assignment
=	OR assignment
^=	XOR assignment
==	Equal to
!=	Not equal to

Parenthesis

Postfix

Prefix

Multiplicative

Additive

Relational

Logical



Java

Operator
Precedence

Operators	Notation	Precedence/Priority
Postfix	expr++, expr--	1
Unary	++expr, --expr, +expr -expr, ~, !	2
Multiplicative	*, /, %	3
Additive	+, -	4
Shift	<<, >>, >>>	5
Relational	<, >, <=, >=, instanceof	6
Equality	==, !=	7
Bitwise AND	&	8
Bitwise Exclusive OR	^	9
Bitwise Inclusive OR		10
Logical AND	&&	11
Logical OR		12
Ternary	? :	13
Assignment	=, +=, -=, *=, /=, %=, &=, ^=, = , <<=, >>=, >>>=	14

Java's Math class

Method name	Description
<code>Math.abs(<i>value</i>)</code>	absolute value
<code>Math.ceil(<i>value</i>)</code>	rounds up
<code>Math.floor(<i>value</i>)</code>	rounds down
<code>Math.log10(<i>value</i>)</code>	logarithm, base 10
<code>Math.max(<i>value1</i>, <i>value2</i>)</code>	larger of two values
<code>Math.min(<i>value1</i>, <i>value2</i>)</code>	smaller of two values
<code>Math.pow(<i>base</i>, <i>exp</i>)</code>	<i>base</i> to the <i>exp</i> power
<code>Math.random()</code>	random double between 0 and 1
<code>Math.round(<i>value</i>)</code>	nearest whole number
<code>Math.sqrt(<i>value</i>)</code>	square root
<code>Math.sin(<i>value</i>)</code> <code>Math.cos(<i>value</i>)</code> <code>Math.tan(<i>value</i>)</code>	sine/cosine/tangent of an angle in radians
<code>Math.toDegrees(<i>value</i>)</code> <code>Math.toRadians(<i>value</i>)</code>	convert degrees to radians and back

Constant	Description
<code>Math.E</code>	2.7182818...
<code>Math.PI</code>	3.1415926...