EXPRESIONES ARITMETICAS

Las expresiones aritméticas son utilizadas para realizar cálculos matemáticos o lógicos y obtener un resultado que es almacenado en una variable que fue previamente declarada.

Para realizar cálculos matemáticos tenemos los operadores aritméticos:

- Suma: (+) Este operador se utiliza para sumar dos valores.
- Resta: (–) Este operador se utiliza para restar dos valores.
- Multiplicación: (*) Este operador se utiliza para multiplicar dos valores.
- División: (/) Este operador se utiliza para dividir dos valores.
- Módulo: (%) Este operador se utiliza para obtener el resto de la división de dos valores.
- Incremento: (+ +) Este operador se utiliza para aumentar en 1 el valor de una variable.
- Decremento: (- -) Este operador se utiliza para disminuir en 1 el valor de una variable.

NOTA: Recuerde que los valores pueden estar en variables, constantes o métodos que regresan un valor.

OPERADORES BASICO + , - , * , /

Son los que básicamente conocemos desde la primaria.

Ejemplo 01:

```
public class Ejer0211001 {

    public static void main(String[] args) {
        // Uso de los operadores aritmeticos
        int v1 = 10;
        int v2 = 15;
        int v3 = 20;

        // Suma
        int suma = v1 + v2;
        System.out.printf ("%n %n La suma de %d + %d = %d %n",
            v1,v2,suma);

        int resta = v2 - v1;
        System.out.printf ("%n %n La resta de %d - %d = %d %n",
            v2,v1,resta);

        int multi = v1 * v2;
        System.out.printf ("%n %n La multiplicacion de %d * %d = %d %n",
            v1,v2,multi);

        int division = v3 / v1;
        System.out.printf ("%n %n La division de %d / %d = %d %n",
            v1,v2,division);
    }
}
```

Resultado

```
General Output
2
3
4   La suma de 10 + 15 = 25
5
6
7   La resta de 15 - 10 = 5
8
9
10  La multiplicacion de 10 * 15 = 150
11
12
13  La division de 10 / 15 = 2
14
15 Process completed.
16
```

OPERADORES UNITARIOS + + , - -

Los operadores unitarios incrementan el valor de una forma diferente, ya que si esta a la derecha de la variable, primero se asigna el valor y luego se incrementa, pero si está a la izquierda primero se incrementa y luego se asigna.

Ejemplo 02 % + + , - - (a la derecha)

```
public class Ejer0211002 {

    public static void main(String[] args) {
        // Operadores Unitarios
        int valor = 10;
        int valorAuxiliar = 0;

        // Primero asigna y luego se incrementa (derecha)
        valorAuxiliar = valor++;

        System.out.printf ("Valor Auxiliar = %d, valor = %d%n",
            valorAuxiliar, valor );
    }
}
```

Resultado

```
General Output
--------------------Configuration:
Valor Auxiliar = 10, valor = 11

Process completed.
```

NOTA: Primero asigna el valor de 10 y luego se autoincrementa la variable valor.

Ejemplo 03: % + + , - - (a la izquierda)

```
public class Ejer0211003 {


    public static void main(String[] args) {
        // Operadores Unitarios
        int valor = 10;
        int valorAuxiliar = 0;

        // Primero incrementa y luego asigna (izquierda)
        valorAuxiliar = ++valor;

        System.out.printf ("Valor Auxiliar = %d, valor = %d%n",
            valorAuxiliar, valor );

    }
}
```

Resultado

```
General Output
--------------------Configuration:
Valor Auxiliar = 11, valor = 11

Process completed.
```

NOTA: Primero el valor de 10 se incrementa y luego se asigna el valor a la variable.

NOTA: Igual sucede para el operador de autodecremento - -

CONCLUSION

Operador de Incremento (++):

- Se utiliza para incrementar el valor de la variable en 1.
- Puede ser pre-incremento (++variable) o post-incremento (variable++).

Operador de Decremento (--):

- Se utiliza para decrementar el valor de la variable en 1.
- Puede ser pre-decremento (--variable) o post-decremento (variable--).

USO DE PARENTESIS Y JERARQUIA DE OPERADORES

Los operadores tienen una jerarquía que JAVA utiliza para evaluar una expresión (formula) cuando ejecuta el programa. La siguiente lista muestra la jerarquía

- operadores unarios ++, --
- multiplicación y división *, /, %
- suma y resta + , -
- asignación += ,-= , *= , /=

El uso de paréntesis para agrupar la jerarquía de evaluación sigue las mismas reglas que la aritmética convencional. Veamos un problema:

Tenemos tres variables que deseamos sumar las primeras y luego multiplicarla por la tercera: 4 + 3 = 7 y luego 7 x 2 = 14. Veamos el código:

```
public class Ejer0211004 {
    public static void main(String[] args) {
        // Jerarquia de operadores
        int v1 = 4;
        int v2 = 3;
        int v3 = 2;

        // Si deseamos sumar 4 + 3 * 2 para que
        // de 14. ¿Será 14 el resultado?

        int suma = v1 + v2 * v3;
        System.out.println ("La suma es: " + suma);

    }
}
```

Resultado

```
General Output
--------------------
La suma es: 10

Process completed.
```

Nada que ver, para solucionar este inconveniente debido a la jerarquía de operadores, debemos de utilizar paréntesis para agrupar el orden de la evaluación dentro de la expresión.

SOLUCION

```
public class Ejer0211005 {

    public static void main(String[] args) {
        // Jerarquia de operadores
        int v1 = 4;
        int v2 = 3;
        int v3 = 2;

        // Si deseamos sumar 4 + 3 * 2 para que
        // de 14. ¿Será 14 el resultado?

        int suma = ( v1 + v2 ) * v3; // Agrupamos la suma
        System.out.println ("La suma es: " + suma);

    }

}
```

Resultado

```
General Output
--------------------
La suma es: 14

Process completed.
```