Programación en Java – Clase 2

Contenido

Operadores matemáticos

Los operadores aritméticos son símbolos matemáticos utilizados para realizar cálculos en expresiones aritméticas.

Los operadores aritméticos disponibles en Java son:

- Suma (+): Este operador se utiliza para sumar dos valores.

- Resta (-): Este operador se utiliza para restar un valor de otro.

- Multiplicación (*): Este operador se utiliza para multiplicar dos valores.

- División (/): Este operador se utiliza para dividir un valor por otro.

- Módulo (%): Este operador se utiliza para obtener el resto de la división de un valor por otro.

  El operador módulo se representa con el símbolo %. Su función es calcular el resto de una división entera entre dos números. Por ejemplo, si tenemos dos números enteros a y b, el operador módulo a % b calcula el resto de dividir a por b.

```
int a = 10;
int b = 3;
int resultado = a % b;
System.out.println(resultado); // Imprime 1
```

En este ejemplo, la variable “a” tiene el valor de 10 y la variable “b” tiene el valor de 3. Cuando se calcula a % b, el resultado es 1, ya que 10 se puede dividir entre 3 dos veces, quedando un resto de 1.

El operador módulo puede ser útil en diversas situaciones, como por ejemplo para verificar si un número es par o impar (si el resto de la división por 2 es 0, entonces es par. Ya veremos esto la próxima clase!!

Veamos un ejemplo de uso de los operadores:

```
class Main {
    public static void main(String[] args) {
        int x = 10;
        int y = 5;

        // Suma
        int suma = x + y;
        System.out.println("La suma de " + x + " y " + y + " es " + suma);

        // Resta
        int resta = x - y;
        System.out.println("La resta de " + x + " y " + y + " es " + resta);

        // Multiplicación
        int multiplicacion = x * y;
        System.out.println("La multiplicación de " + x + " y " + y + " es " +
        multiplicacion);
```

```
            // División
            int division = x / y;
            System.out.println("La división de " + x + " y " + y + " es " +
            division);

            // Módulo
            int modulo = x % y;
            System.out.println("El módulo de " + x + " y " + y + " es " +
            modulo);
        }
}
```

En primer lugar, se declara una clase llamada " Main " con el modificador public. Dentro de la clase, se define el método main() con el modificador public static. Este es el punto de entrada del programa y se ejecuta cuando se inicia el programa.

Dentro del método main(), se declaran dos variables enteras x e y con valores iniciales de 10 y 5, respectivamente.

Se utilizan los operadores aritméticos +, -, *, / y % para realizar diferentes cálculos con las variables x e y. Los resultados de estos cálculos se almacenan en variables separadas llamadas suma, resta, multiplicación, división y modulo.

Se utiliza el método println() para imprimir los resultados de los cálculos. Cada línea de código utiliza el operador + para concatenar los valores de las variables con texto explicativo.

El programa finaliza cuando el método main() llega a su final.

Manejo de cadenas de caracteres

La clase String es una de las clases más importantes en Java y se utiliza para representar y manipular cadenas de caracteres. Por ahora, es un concepto en el que no profundizaremos hasta dentro de unas clases.

Te acercamos un resumen de algunas de las características más importantes de la clase String:

La clase String proporciona una gran cantidad de métodos para manipular cadenas de caracteres, como length(), charAt() , entre otros. Estos métodos permiten realizar operaciones como obtener la longitud de una cadena, obtener un carácter específico en una posición dada entre otras acciones que veremos pronto.

En Java, las cadenas de caracteres se pueden concatenar utilizando el operador +. Por ejemplo, si tienes dos cadenas s1 y s2, puedes concatenarlas usando el siguiente código:

```
String s3 = s1 + s2;
```

Conversiones entre datos

A continuación, veremos una serie de herramientas que nos permitirán realizar conversiones entre diferentes tipos de datos:

Conversión de entero a String

```
int numeroEntero = 42;
String numeroString = Integer.toString(numeroEntero);
```

Conversión de String a entero

```
String numeroString = "42";
int numeroEntero = Integer.parseInt(numeroString);
```

Conversión de String a float

```
String numeroString = "3.14";
float numeroFloat = Float.parseFloat(numeroString);
```

Conversión de float a String

```
float numeroFloat = 3.14f;
String numeroString = Float.toString(numeroFloat);
```

Tengamos en cuenta que la conversión de un tipo de dato a otro puede generar errores si los datos no son compatibles. Por ejemplo, si intentamos convertir una cadena de caracteres que no contiene un número válido en un entero, se producirá una excepción en tiempo de ejecución.

Veamos un ejemplo completo:

```
class Main {
        public static void main(String[] args) {
                //Conversión de entero a String
                int numeroEntero = 42;
                String numeroString = Integer.toString(numeroEntero);
                System.out.println(numeroString);

                //Conversión de String a entero
                String StringNumero = "42";
                numeroEntero =  Integer.parseInt(StringNumero);
                System.out.println(numeroEntero);

                //Conversión de String a float
```

```
            String StringFloat = "3.14";
            float numeroFloat = Float.parseFloat(StringFloat);
            System.out.println(StringFloat);
            //Conversión de float a String
            String floatString = Float.toString(numeroFloat);
            System.out.println(floatString);
        }
}
```