La clase Scanner

La clase Scanner es utilizada en Java para obtener la entrada del usuario desde la consola. Al trabajar con esta clase, es importante comprender los conceptos de clase e instancia. Una clase se puede visualizar como un plano o una plantilla que define las características y comportamientos de un objeto. Al crear una instancia de una clase, se crea un objeto específico basado en esa plantilla.

> Por ahora, tu principal enfoque debe ser comprender cómo utilizar esta clase para capturar datos ingresados por el usuario. Más adelante, explorarás más a fondo los conceptos de clases y objetos.

Con la ayuda de la clase Scanner vamos a poder solicitar que se ingresen datos por consola y el programa se detenga hasta que se ingrese la información:

La clase Scanner nos permite solicitar datos al usuario y detener la ejecución del programa hasta que se ingrese la información requerida. Para utilizar la clase Scanner, es necesario importarla al comienzo del archivo de Java mediante la instrucción:

```
import java.util.Scanner;
```

Después de importarla, se puede crear una instancia de la clase Scanner para comenzar a obtener la entrada del usuario. Usualmente, se crea una instancia asociada al flujo de entrada estándar, que es el teclado. Por ejemplo:

```
Scanner miScanner = new Scanner(System.in);
```

Una vez que se tiene una instancia de Scanner, se pueden utilizar sus métodos para leer los datos ingresados por el usuario. Algunos de los métodos más comunes son:

- **nextBoolean():** Lee un valor booleano (true o false) desde la entrada.
- **nextInt():** Lee un número entero desde la entrada.

- **nextDouble():** Lee un número de tipo double desde la entrada.
- **nextLine():** Lee una línea completa de texto desde la entrada.

A continuación, se muestra un ejemplo de cómo utilizar la clase Scanner para obtener un número entero ingresado por el usuario:

```
Scanner miScanner = new Scanner(System.in);

System.out.print("Ingresa un número entero: ");
int numero = miScanner.nextInt();

System.out.println("El número ingresado es: " + numero);
```

En este ejemplo, el programa solicita al usuario ingresar un número entero utilizando el método **nextInt()** de la instancia de Scanner. Luego, se asigna ese número a la variable "numero" y se imprime en pantalla.

Es importante tener en cuenta que al utilizar la clase Scanner, es una buena práctica cerrar el objeto Scanner una vez que ya no se necesita. Esto se hace mediante la llamada al método close() del objeto Scanner.

Cuando se cierra un objeto Scanner, se liberan los recursos asociados con él, como los recursos de entrada de la consola. Si no se cierra el objeto Scanner, podría causar una fuga de recursos, lo que eventualmente podría llevar a problemas de rendimiento del programa.

Para cerrar un objeto Scanner, simplemente llamamos al método close() en el objeto Scanner que hemos creado. Por ejemplo:
`miScanner.close();`
La llamada a close() asegura que los recursos asociados con el objeto Scanner sean liberados adecuadamente, lo que contribuye a un mejor manejo de recursos y previene problemas de rendimiento en el programa.

Operadores en Java

Los operadores en Java son símbolos especiales que se utilizan para realizar operaciones en variables y valores. Estas operaciones pueden ser aritméticas, de asignación, de comparación, lógicas, de incremento y decremento, entre otras. Los operadores en Java se clasifican en diferentes categorías según su funcionalidad. Algunos ejemplos de operadores en Java incluyen:

Operadores Aritméticos: Los operadores aritméticos en Java son símbolos especiales que se utilizan para realizar operaciones matemáticas en variables. Estos operadores incluyen:

Operador	Descripción	Ejemplo
Adición (+)	Suma dos valores.	a + b
Sustracción(-)	Resta un valor de otro.	a - b
Multiplicación (*)	Multiplica dos valores.	a * b
División (/)	Divide un valor por otro.	a / b
Módulo (%)	Obtiene el residuo de una división.	a % b

Veamos un ejemplo de implementación:

```
public class App {
    public static void main(String[] args) {
        // Definir variables
        int a = 10;
        int b = 5;
        // Suma
        int suma = a + b;
        System.out.println("Suma: " + suma); // Salida: 15
        // Resta
        int resta = a - b;
        System.out.println("Resta: " + resta); // Salida: 5
        // Multiplicación
        int multiplicacion = a * b;
```

```
        System.out.println("Multiplicación: " + multiplicacion); // Salida: 50
        // División
        int division = a / b;
        System.out.println("División: " + division); // Salida: 2
        // Módulo
        int modulo = a % b;
        System.out.println("Módulo: " + modulo); // Salida: 0
    }
}
```

Operadores de Asignación: Los operadores de asignación en Java se utilizan para asignar un valor a una variable. Estos operadores combinan la asignación con una operación aritmética o de bits. Los operadores de asignación más comunes son:

Operador	Descripción	Ejemplo
Asignación (=)	Asigna un valor a una variable	a = b
Suma y asignación (+=)	Suma y luego asigna el valor	a += b
Resta y asignación (-=)	Resta y luego asigna el valor	a -= b
Multiplicación y asignación (*=)	Multiplica y luego asigna el valor	a *= b
División y asignación (/=)	Divide y luego asigna el valor	a /= b
Módulo y asignación (%=)	Aplica el módulo y luego asigna el valo	a %= b

Operadores de Incremento y Decremento: Los operadores de incremento y decremento en Java son utilizados para aumentar o disminuir el valor de una variable en una unidad. Estos operadores son:

Operador	Descripción	Ejemplo
Incremento (++	Aumenta el valor de una variable en 1	a++ o ++a.
Decremento (--)	Disminuye el valor de una variable en 1	a-- o --a.

Veamos un ejemplo de implementación:

```java
public class App {
    public static void main(String[] args) {
        // Definir una variable
        int numero = 5;

        // Operador de incremento (++): aumenta el valor de la variable en 1
        numero++; // Equivalente a numero = numero + 1;
        System.out.println("Después del incremento: " + numero); // Salida: 6

        // Operador de decremento (--): disminuye el valor de la variable en 1
        numero--; // Equivalente a numero = numero - 1;
        System.out.println("Después del decremento: " + numero); // Salida: 5

        // Usar el operador de incremento/decremento en una expresión
          int resultado = numero++ * 2; // Incrementa 'numero' después de usar su
valor en la expresión
        System.out.println("Resultado: " + resultado); // Salida: 10

        // El valor de 'numero' ha sido incrementado
         System.out.println("Valor de número después de la operación: " + numero);
// Salida: 6
    }
}
```

En este ejemplo, primero se inicializa la variable numero con el valor 5. Luego, se utilizan los operadores de incremento ++ y decremento -- para aumentar y disminuir el valor de numero, respectivamente. También se muestra cómo utilizar estos operadores dentro de una expresión y cómo el operador de incremento ++ incrementa el valor después de usarlo en la expresión.

Operadores Relacionales o de Comparación: Los operadores relacionales o de comparación en Java se utilizan para comparar dos valores y producir un resultado booleano que indica si la comparación es verdadera o falsa. Estos operadores incluyen:

Operador	Descripción	Ejemplo
Igual a (==)	Verifica si dos valores son iguales.	a == b
No igual a (!=)	Verifica si dos valores no son iguales.	a != b
Mayor que (>)	Verifica si un valor es mayor que otro.	a > b

Menor que (<)	Verifica si un valor es menor que otro.	a < b
Mayor o igual que (>=)	Verifica si un valor es mayor o igual que otro.	a >= b
Menor o igual que (<=)	Verifica si un valor es menor o igual que otro.	a <= b

Veamos un ejemplo de implementación:

```
public class App {
    public static void main(String[] args) {
      // Declarar variables
      int a = 5;
      int b = 10;

      // Igual a (==)
      System.out.println("a == b: " + (a == b)); // Salida: false

      // No igual a (!=)
      System.out.println("a != b: " + (a != b)); // Salida: true

      // Mayor que (>)
      System.out.println("a > b: " + (a > b)); // Salida: false

      // Menor que (<)
      System.out.println("a < b: " + (a < b)); // Salida: true

      // Mayor o igual que (>=)
      System.out.println("a >= b: " + (a >= b)); // Salida: false

      // Menor o igual que (<=)
      System.out.println("a <= b: " + (a <= b)); // Salida: true
  }
}
```

Operadores Lógicos: Los operadores lógicos en Java se utilizan para combinar expresiones booleanas y producir un resultado booleano basado en ellas. Los operadores lógicos más comunes son:

Operador	Descripción	Ejemplo
AND lógico (&&)	Devuelve verdadero si ambos operandos son verdaderos.	a && b
OR lógico (||)	Devuelve verdadero si al menos uno de los operadores es verdadero.	a || b.

NOT lógico (!)	Invierte el valor de verdad del operando.	!a

Operador Condicional (Operador Ternario): El operador condicional, también conocido como operador ternario, es una expresión condicional en Java que permite tomar decisiones basadas en una condición. El operador ternario es una forma concisa y eficiente de escribir expresiones condicionales simples en Java. Sin embargo, es importante no abusar de su uso para mantener la legibilidad del código.

Operador	Descripción	Ejemplo
(? :)	a > b ? a : b (si a es mayor que b, devuelve a, sino devuelve b).	a > b ? a : b (si a es mayor que b, devuelve a, sino devuelve b).

Te dejamos aquí unos ejemplos de implementación:

```
public class App {
    Run | Debug
    public static void main(String[] args) {
    int edad = 20;
    // Utilizando el operador ternario para determinar el mensaje basado en la edad
    String mensaje = (edad >= 18) ? "Eres mayor de edad" : "Eres menor de edad";
    System.out.println(mensaje);
}}
```

En este ejemplo, si la edad es mayor o igual a 18, se asigna el texto "Eres mayor de edad" a la variable mensaje, de lo contrario se asigna el texto "Eres menor de edad". Luego, se imprime el mensaje en la consola.

```java
public class App {
    Run | Debug
    public static void main(String[] args) {
        int a = 5;
        int b = 10;
        // Utilizando el operador ternario para determinar el mayor entre a y b
        int mayor = (a > b) ? a : b;
        System.out.println("El número mayor es: " + mayor);
    }
}
```

En este ejemplo, la condición a > b se evalúa primero. Si es verdadera, se asigna el valor de a a la variable mayor, de lo contrario se asigna el valor de b. Finalmente, se imprime en la consola el número mayor entre a y b.

Operadores de Bit: Los operadores de bit en Java son utilizados para realizar operaciones a nivel de bits en los valores enteros. Estos operadores manipulan los bits individuales de los operandos. Los operadores de bits más comunes son:

Operador	Descripción	Ejemplo
AND binario (&)	Realiza una operación AND en cada par de bits.	a & b
OR binario (|)	Realiza una operación OR en cada par de bits	a | b.
XOR binario (^)	Realiza una operación XOR en cada par de bits.	a ^ b
Complemento binario (~)	Invierte todos los bits.	~a
Desplazamiento a la izquierda (<<)	Desplaza los bits a la izquierda, rellena con ceros a la derecha.	a << 2
Desplazamiento a la derecha (>>)	Desplaza los bits a la derecha, rellena con el bit más significativo a la izquierda.	a >> 2
Desplazamiento a la derecha sin signo (>>>)	Desplaza los bits a la derecha, rellena con ceros a la izquierda.	a >>> 2