Clase 3: Operadores y Expresiones

3.1. Operadores

3.1.1. ¿Qué es un operador?

Los operadores se utilizan para realizar operaciones entre variables y valores.

Son símbolos especiales que se utilizan en expresiones.

3.1.2. Tipos de Operadores

Operadores Aritméticos

Operador	Nombre	Significado	Ejemplo
+	Suma	Suma dos números.	`10 + 15 = 25`
-	Resta	Resta dos números.	`100 – 90 = 10`
*	Multiplicación	Multiplica dos números.	`12 * 3 = 36`
/	División	Divide dos números.	`15 / 2 = 7`
%	Módulo	Retorna el residuo de la división de dos números. Sumamente útil para saber si un número es múltiplo de otro número.	`15 % 2 = 1`

Operadores de Asignación

Operador	Nombre	Significado	Ejemplo
=	Asignación simple	Asigna un valor a una variable	`int x = 5;`
+=	Suma y Asignación	Asigna el valor a la izquierda de la expresión a sí misma y le suma el valor a la derecha de la expresión.	`x += 3;` `(x = x + 3)`
-=	Resta y Asignación	Asigna el valor a la izquierda de la expresión a sí misma y le resta el valor a la derecha de la expresión.	`x -= 3;` `(x = x – 3)`
*=	Multiplicación y Asignación	Asigna el valor a la izquierda de la expresión a sí misma y le multiplica el valor a la derecha de la expresión.	`x *= 3;` `(x = x * 3)`
/=	División y Asignación	Asigna el valor a la izquierda de la expresión a sí misma y lo divide entre el valor a la derecha de la expresión.	`x /= 3;` `(x = x / 3)`
%=	Módulo y Asignación	Asigna el valor a la izquierda de la expresión a sí misma y retorna el residuo de la división entre el valor a la derecha de la expresión.	`x %= 3;` `(x = x % 3)`

Operadores de Comparación o Relacionales

Operador	Nombre	Significado	Ejemplo
==	Igual a	Si ambos valores son iguales retorna true. Si no son iguales retorna false.	`5 == 5; (true)`
			`5 == 1; (false)`
!=	Diferente de		`5 != 5; (false)`

		Si ambos valores son distintos (no son iguales) retorna true. Si no son distintos (son iguales) retorna false.	`5 != 1; (true)`

<	Menor que	Si el valor a la izquierda del operador es menor que el valor a la derecha del operador retorna true. De lo contrario retorna false.	`5 < 10; (true)`
			`5 < 5; (false)`
			`5 < 1; (false)`

>	Mayor que	Si el valor a la izquierda del operador es mayor que el valor a la derecha del operador retorna true. De lo contrario retorna false.	`5 > 10; (false)`
			`5 > 5; (false)`
			`5 > 1; (true)`

<=	Menor o igual a	Si el valor a la izquierda del operador es menor o igual que el valor a la derecha del operador retorna true. De lo contrario retorna false.	`5 <= 10; (true)`
			`5 <= 5; (true)`
			`5 <= 1; (false)`

>=	Mayor o igual a	Si el valor a la izquierda del operador es mayor o igual que el valor a la derecha del operador retorna true. De lo contrario retorna false.	`5 >= 10; (false)`
			`5 >= 5; (true)`
			`5 >= 1; (true)`

Operadores Lógicos

Operador	Nombre	Significado	Ejemplo

&&	AND lógico	Evalúa dos expresiones a la misma vez. Su valor de retorno será true solo si ambas expresiones son true.	`5 == 5 && 10 == 10; (true)`
			`(true AND true = true)`
			`5 == 5 && 10 == 6; (false)`
			`(true AND false = false)`
			`5 == 3 && 10 == 6; (false)`
			`(false AND false = false)`

||	OR lógico	Evalúa dos expresiones a la misma vez. Su valor de retorno será true si:
		Ambas expresiones son true.
		Si solo (1) expresión es true.
			`5 == 5 || 10 == 10; (true)`
			`(true OR true = true)`
			`5 == 5 || 10 == 6; (true)`
			`(true OR false = true)`
			`5 == 3 || 10 == 6; (false)`
			`(false OR false = false)`

!	NOT lógico	Invierte el valor de una expresión booleana.
		Si la expresión es true, devolverá false.
		Si la expresión es false, devolverá true.
			`condicion = true;`
			`!condicion; (false)`
			`Condición = false;`
			`!condicion; (true)`

Operadores de Incremento y Decremento

Operador	Nombre	Significado	Ejemplo

++	Incremento	Incrementa en 1 (+1) el valor de la variable. Hay dos maneras de utilizarlo:
		
		Postincremento: Primero se utiliza el valor actual de la variable en la expresión y al finalizar se incrementa en 1.
		
		Preincremento: Primero se incrementa en 1 el valor actual de la variable y luego se utiliza el nuevo valor en la expresión.
			Postincremento
			`int a = 5;`
			`int resultado = a++;`
			`(resultado = 5)`
			`(a = 6)`
			
			Preincremento
			`int a = 5;`
			`int resultado = ++a;`
			`(resultado = 6)`
			`(a = 6)`

--	Decremento	Decrementa en 1 (-1) el valor de la variable. Hay dos maneras de utilizarlo:
		
		Postdecremento: Primero se utiliza el valor actual de la variable en la expresión y al finalizar se decrementa en 1.
		
		Predecremento: Primero se decrementa en 1 el valor actual de la variable y luego se utiliza el nuevo valor en la expresión.
			Postdecremento
			`int a = 5;`
			`int resultado = a--;`
			`(resultado = 5)`
			`(a = 4)`
			
			Predecremento
			`int a = 5;`
			`int resultado = --a;`
			`(resultado = 4)`
			`(a = 4)`

Operadores de concatenación

Operador	Nombre	Significado	Ejemplo

+	Concatenación	Cuando el operador "+" se encuentre utilizado con valores literales char o String se utilizará como operador de concatenación.	`String nombre = "Omar";`
			`String apellido = "Montoya";`
			`String nombreCompleto = nombre + apellido;`
			`(nombreCompleto = "OmarMontoya")`

3.2. Expresiones

3.2.1. ¿Qué es una expresión?

Una expresión es una combinación de variables, operadores y valores literales que se evalúan para producir un resultado. Las expresiones pueden ser simples o más complejas, y se utiliza en diferentes contextos dentro del código Java.

3.2.2. Tipos de Expresiones

Expresiones Aritméticas

Son expresiones que utilizan operadores aritméticos. Se emplean para formular procesos matemáticos.

```
1  //Expresión aritmética
2  int resultado = 6 + 4 * (10 - 8);
```

Expresiones de Asignación

Son expresiones que utilizan operadores de asignación. Su finalidad de asignarle un valor a una variable.

```
1  //Expresión de asignación
2  int x = 20;
3  int y = x * 2;
4  String z = "Hola Mundo";
```

Expresiones de Comparación

Son expresiones que utilizan operadores de comparación y/o operadores lógicos. El resultado de la expresión siempre será true o false, sin importar cuan compleja sea la expresión.

```
1  //Expresión de comparación
2  int x = 10;
3  int y = 100;
4  boolean esMayor = (x > y); //Devuelve false
5  boolean esIgual = (x == y); //Devuelve false
6  boolean esDiferente = (x != y) //Devuelve true
7  boolean noEsDiferente = !(x != y) //Devuelve false
```

Expresiones de Concatenación

Son expresiones que utilizan el operador "+" con valores literales o variables de tipo carácter o cadena. Su finalidad es unir cadenas.

```
1  //Expresión de concatenación
2  String cadena1 = "Hola";
3  String cadena2 = "Mundo";
4  String saludo = cadena1 + " " + cadena2; //Devuelve "Hola Mundo"
```