El futuro digital es de todos | MinTIC

UNIVERSIDAD EL BOSQUE | MisiónTIC 2022

Ciclo 2

Semana 1

Introducción a la Programación Orientada a Objetos (POO)

Lectura 3 - Expresiones y operadores relacionales, lógico, aritméticos y de asignación

Convenio de asociación no. 777 de 2021 suscrito entre el Fondo Único de Tecnologías de la Información y las Comunicaciones y la Universidad el Bosque.

Expresiones y operadores relacionales, lógico, aritméticos y de asignación

Expresiones

Una expresión se le conoce a la forma en que se expresa en un lenguaje de programación algo del estado de un objeto, o en otras palabras como se dice algo sobre el mundo del problema. Estas se plasman en el cuerpo del método y se conforman de operandos (atributos, parámetros, literales, constantes o llamadas de métodos) y operadores que indican como calcular el valor de la expresión.

Los operadores usados en una expresión dependen del tipo de dato del operando

Operadores

1. Operadores relacionales

Los lenguajes de programación disponen de operadores relacionales, que establecen un valor de verdad ya sea falso o verdadero y se usan para comparar dos valores o expresiones compatibles, devolviendo un resultado de tipo lógico.

Estos operadores de relación se aplican a cualquiera de los tipos de datos como enteros, reales, lógico, carácter, etc.

Las que se aplican entre números reales y enteros están establecidas por reglas matemáticas, en las expresiones lógicas el valor falso se considera menor que el valor verdadero. Las de caracteres y cadenas se basan en el código ASCII de cada carácter.

Los operadores relacionales son seis como se muestran a continuación:

Significado	Símbolo	Ejemplo
Es igual que	`==`	`valorUnitario == 55.75`
Es diferente de	`!=`	`tipo != PAPELERIA`
Es menor que	`<`	`cantidadBodega < 120`
Es mayor que	`>`	`cantidadBodega > cantidadMinima`
Es menor o igual que	`<=`	`valorUnitario <= 100.0`
Es mayor o igual que	`>=`	`valorUnitario >= 100.0`

2. Operadores lógicos

Los operadores lógicos describen escenarios más complicadas, con la unión de expresiones relacionales o de atributos de tipo booleano. Los operadores lógicos son tres:

&& (y) , || (o), ! (no)

En donde:

- operando1 && operando2 es verdadero, si ambos son verdaderos.
- operando1 || operando2 es verdadero, si cualquiera de los dos es verdadero.
- !operando es verdadero, si el operando es falso.

Los operadores && y || se evalúan de izquierda a derecha hasta que se determine si es verdadera o falsa.

Los operadores sobre cadenas de caracteres se implementan mediante clases especiales y en algunos casos, para invocar sus operaciones se utiliza la sintaxis de llamada de métodos. Existen muchas operaciones sobre cadenas de caracteres entre la que está la concatenación (+), en el de comparación (equals) y el de extracción de un carácter (charAt).

3. Operadores aritméticos

Para el manejo de expresiones aritméticas se usan operadores aritméticos, que combinados constituyen las expresiones. Los operadores aritméticos son la suma (+), la resta (-), la multiplicación (*) y la división (/).

Estas pueden usarse solas o combinadas con varios operadores con el uso de paréntesis, y al resolverse se tiene en cuenta la prioridad, donde primero se realizan los paréntesis, luego las potencias, las multiplicaciones y divisiones y finalmente las sumas y restas. Si hay dos operaciones con igual prioridad, se evalúan de izquierda a derecha.

4. Operadores de asignación

Los operadores de asignación permiten cambiar el valor de un atributo de un objeto, reflejando un cambio en el mundo del problema.

- Operador ++ . Para atributos de tipo entero, para incrementarlo en 1, y es de asignación pues modifica el valor del operando al sumarle el valor 1.
- Operador -- . Para atributos de tipo entero, para disminuirlo en 1. Se utiliza de manera equivalente al de incremento.
- Operador += . Incrementa un atributo en cualquier valor.
- Operador -= . Disminuye un atributo en cualquier valor Se utiliza de manera equivalente al operador += .

Interacción usuario-computador: la consola

Entrada y salida por consola en java

Con frecuencia, cada programa que se programa se necesita comunicar con un usuario. Los usuarios son las personas que interactúan con los programas.

La comunicación con el usuario es en dos (2) sentidos:

- Entrada (input): El usuario ingresa la información al programa desarrollado.
- Salida (output): el programa da información al usuario.

A continuación, se va a ver la forma de realizar esto en una aplicación de línea de comandos.

1. Salida por consola

En Java existen funcionalidades por defecto. Una de estas funcionalidades es la denominada System.out. Este objeto dispone de un método llamado **println** que nos permite imprimir algo por pantalla en una ventana de consola.

La sintaxis de este método es:

```
System.out.println("Hola mundo");
```

En este espacio se puede ingresar el mensaje que se quiere mostrar por consola

Si se requiere mostrar en consola un valor de una variable se puede hacer de la siguiente

```
int valor = 120 ;
System.out.println("El valor es igual a " + valor);
```

manera:

2. Entrada por consola

De la misma manera que la salida por consola, Java cuenta con un objeto y un método que permite leer un dato ingresado por un usuario por consola.

Para utilizar el método de entrada, se debe adicionar la siguiente importación:

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

Después se debe crear el objeto de la clase Scanner de la siguiente forma:

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

Después de esto ya es posible capturar información dada por el usuario, por ejemplo, se puede capturar información y verificarla de la siguiente forma:

```
System.out.println("Ingrese primer nombre de la persona: ");
String nombre = leer.next() ;
System.out.println("El  nombre ingresado fue " + nombre);
```

El anterior ejemplo, permite leer un texto hasta el primer espacio en blanco que encuentre, pero si se quiere leer el nombre completo debe ser de la siguiente forma:

```
System.out.println("Ingrese nombre completo de la persona: ");
String nombre_completo = leer.nextLine() ;
System.out.println("El  nombre ingresado fue " + nombre_completo);
```

Si se requiere leer un dato entero, un ejemplo es:

```
System.out.println("Ingrese edad (años): ");
int edad = leer.nextInt() ;
System.out.println("La edad ingresada es " + edad);
```

Para leer un dato real, un ejemplo es:

```
System.out.println("Ingrese estatura (mts): ");
double altura = leer.nextDouble() ;
System.out.println("La estatura ingresada es " + altura);
```

En algunas ocasiones, la consola de java se salta alguna lectura, esto se debe a que los métodos de Scanner no consumen el salto de línea, para esto se aconseja limpiar el buffer, por tanto, puede probar el siguiente ejemplo para verificar su funcionamiento. Es muy importante siempre cerrar la instancia de leer.

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

System.out.println("Ingrese primer nombre de la persona: ");
String nombre = leer.next();
leer.nextLine();

System.out.println("Ingrese nombre completo de la persona: ");
String nombre_completo = leer.nextLine() ;

System.out.println("Ingrese edad (años): ");
int edad = leer.nextInt();
leer.nextLine();

System.out.println("Ingrese estatura (mts): ");
double altura = leer.nextDouble();
leer.nextLine();

System.out.println("El  nombre ingresado fue " + nombre);
System.out.println("El  nombre ingresado fue " + nombre_completo);
System.out.println("La edad ingresada es " + edad);
System.out.println("La estatura ingresada es " + altura);

leer.close();
```