Java avanzado

Introducción

La versión 8 de Java: Introdujo a los desarrolladores a la programación funcional con expresiones lambda. Esta versión de Java notificó efectivamente a los desarrolladores que ya no es suficiente pensar en la programación solo desde la perspectiva imperativa y orientada a objetos. Un desarrollador de Java también debe poder pensar y codificar utilizando el paradigma funcional declarativo.

El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) incluyó los lenguajes de programación funcional en sus 25 lenguajes de programación principales para 2018, y Google
Trends actualmente clasifica la programación funcional como más popular que la programación
orientada a objetos.

Que es la Programación Funcional?

Las computadoras generalmente implementan la arquitectura de Von Neumann. Esta arquitectura está sesgada hacia la programación imperativa, que es un paradigma de programación que usa declaraciones para cambiar el estado de un programa.

La programación funcional es un estilo de programación en el que los cálculos se codifican como funciones de programación funcional. Estas son construcciones similares a funciones matemáticas (por ejemplo, funciones lambda) que se evalúan en contextos de expresión.

Filter y map

Los lenguajes de programación funcionales suelen proporcionar varias funciones útiles de orden superior. Dos ejemplos comunes son el filtro y el mapa.

- Un filtro procesa una lista en algún orden para producir una nueva lista que contenga exactamente aquellos elementos de la lista original para los cuales un predicado dado (piense en una expresión booleana) devuelve verdadero.
- Un mapa aplica una función dada a cada elemento de una lista, devolviendo una lista de resultados en el mismo orden.

Reduce

Otra función común de orden superior es reducir, que es más comúnmente conocida como pliegue. Esta función reduce una lista a un solo valor

Que es una expresión Lambda ?

Una expresión lambda representa una función anónima. Se compone de un conjunto de parámetros, un operador lambda (->) y un cuerpo de función. Los siguientes son ejemplos de expresiones Java de Lambda:

```
1.- n -> n % 2 != 0;
2.- (char c) -> c == 'y';
3.- (x, y) -> x + y;
4.- (int a, int b) -> a * a + b * b;
5.- () -> 42
6.- () -> { return 3.14 };
7.- (String s) -> { System.out.println(s); };
8.- () -> { System.out.println("Hello World!"); };
```

Interpretación:

1. Dado un número **n** devuelve un valor booleano que indica si es impar.
2. Dado un carácter, **c** devuelve un valor booleano que indica si es igual a **"y"**.
3. Dados dos números **x** e y devuelve otro número con su suma.
4. Dados dos enteros **a** y **b** devuelve otro entero con la suma de sus cuadrados.
5. Dado que ningún parámetro devuelve el entero 42.
6. Dado que no hay parámetros se devuelve el doble 3.14.
7. Dada una cadena **s** imprime la cadena en la salida principal y devuelve vacío.
8. Dale sin parámetros de impresión **Hello World!** A la salida principal y devuelve void.

Los parámetros

1. Una expresión lambda puede recibir cero, uno o más parámetros.
2. El tipo de los parámetros se puede declarar explícitamente o se puede inferir del contexto.
3. Los parámetros están entre paréntesis y separados por comas.
4. Los paréntesis vacíos se utilizan para representar un conjunto vacío de parámetros.
5. Cuando hay un solo parámetro, si se infiere su tipo, no es obligatorio usar paréntesis.

El cuerpo

1. El cuerpo de la expresión lambda puede contener cero, una o más declaraciones.
2. Cuando hay una sola frase, los paréntesis no son obligatorios y el tipo de retorno de la función anónima es el mismo que el de la expresión del cuerpo.
3. Cuando hay más de una declaración, éstas deben incluirse entre paréntesis (un bloque de código) y el tipo de retorno de la función anónima es el mismo que el tipo del valor devuelto dentro del bloque de código, o anulado si no se devuelve nada.