INTRODUCCIÓN AL API STREAM

API STREAM

- Una de las grandes novedades de Java SE 8, junto a las expresiones lambda.
- Permite realizar operaciones de filtro/mapeo/reducción sobre colecciones de datos.
- Puede trabajar de forma secuencia o paralela.
- Transparente al desarrollador.
- Combinación perfecta para las expresiones lambda.

STREAM

Es una secuencia de elementos que soporta operaciones para procesarlos

- Usando expresiones lambda
- Permitiendo el encadenamiento de operaciones (código legible y conciso).
- De forma secuencial o paralela.

Definido por la interfaz
java.util.stream.Stream<T>

CARACTERÍSTICAS DE UN STREAM

- Las operaciones intermedias retornan un *Stream* (encadenamiento).
- Las operaciones intermedias se encolan, y son invocadas al invocar una operación terminal.
- Solo se puede recorrer una vez.
- Iteración interna vs. externa: nos centramos en qué hacer con los datos, no en como recorrerlos.

SUBTIPOS BÁSICOS

- IntStream
- LongStream
- DoubleStream

FORMAS DE OBTENER UN STREAM

- *Stream.of(…)* ordenado y secuencial de los parámetros que se le pasan
- *Arrays.stream(T[])* secuencial a partir del array proporcionado. Si el array es de tipo básico, se devuelve un subtipo de Stream.
- *Stream.empty()* devuelve un stream secuencial y vacío.

FORMAS DE OBTENER UN STREAM

- *Stream.iterate(T, UnaryOperator<T>)* devuelve un stream infinito, ordenado, y secuencial, a partir de un valor y de aplicar una función a ese valor. Se puede limitar su tamaño con *limit(long)*.
- *Collection<T>.stream()*, *Collection<T>.parallelStream()* devuelve un stream (secuencial o paralelo) a partir de una colección.

FORMAS DE OBTENER UN STREAM

- *Stream.generate(Supplier<T>)* retorna un stream infinito, secuencial y no ordenado a partir de un *Supplier*.

OPERACIONES INTERMEDIAS SOBRE UN STREAM

- Son operaciones que devuelven un stream.
- Nos permiten realizar diversas funciones (filtrado, transformación, …)

OPERACIONES DE FILTRADO

- *filter(Predicate<T>)*: nos permite filtrar utilizando una condición.
- limit(n): nos permite obtener los n primeros elementos.
- skip(m): nos permite *obviar* los primeros m elementos.

OPERACIONES DE MAPEO

- *map(Function<T, R>)*: nos permite transformar los valores de un stream a través de una expresión lambda o instancia de *Function*.
- *mapToInt(…)*, *mapToDouble(…)*, *mapToLong(…)* nos permite transformar los valores en uno de estos tipos básicos, obteniendo un *IntStream*, *DoubleStream*, *LongStream*, respectivamente.

OPERACIONES TERMINALES SOBRE UN STREAM

Provoca que se ejecuten todas las operaciones terminales.
Varios tipos:

- Consumo de elementos: forEach(…)
- Obtener datos del stream
- Recolección de elementos para transformarlos en otro objeto, como una colección.

Los dos últimos tipos los estudiaremos en lecciones posteriores.