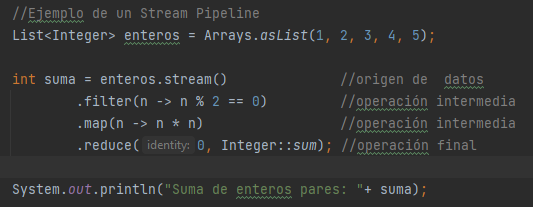
**Iteración externa**: es la forma tradicional en la que se trabaja con colecciones, usando ciclos for, while, etc. Sólo se puede ejecutar en un solo hilo o de forma secuencial.

**Iteración interna**: es la que usa stream y sus diferentes tipos de operaciones, para iterar sobre colecciones de datos, tiene la ventana de ser más entendible con un, código más limpio y además se pueden ejecutar en procesamiento paralelo (multithread o multi núcleo), usando **parallelStream**.

**Stream Pipeline**:

* Comienza con un origen de datos.
* Realiza una o más operaciones intermedias.
* Finaliza con una operación terminal.
* Se forma mediante el encadenamiento de llamadas.

Ejemplo:



**Operaciones intermedias**:

* Especifica las tareas a realizar sobre los elementos del Stream (operaciones no definitivas).
* Siempre producen un nuevo Stream.
* Son perezosas, no se ejecutan sino hasta que se ejecuta una operación terminal.

Algunos ejemplos de **operaciones intermedias** son:

* **filter**: produce un stream que contiene solo elementos que satisfacen una condición.
* **distinct**: produce un stream que contiene solo elementos únicos.
* **limit**: produce un stream con el numero especificado de elementos a partir del inicial del stream inicial.
* **map**: produce un stream, en el que cada elemento de stream, está asociado a un nuevo valor.
* **sorted**: produce un stream en el que los elementos están ordenados.

**Operaciones terminales**:

* Esta operación, inicial el procesamiento de las operaciones terminales de un pipeline.
* Se dice que son ansiosas, se ejecutan en cuanto son invocadas.
* Siempre producen un resultado.

Algunos ejemplos de **operaciones finales** son:

* + **forEach**: realiza un procesamiento sobre cada elemento en un stream (por ejemplo, mostrar cada elemento en pantalla).
  + **average:** calcula el promedio de los elementos en un stream numérico.
  + **count**: devuelve el número de elementos en el stream.
  + **max**: localiza el valor más grande en un stream numérico.
  + **min**: localiza y devuelve el valor más pequeño de un flujo numérico.
  + **reduce**: reduce los elementos de una colección a un solo valor mediante el uso de una función de acumulación.
  + **collect:** crea una colección de elementos que contiene los resultados de las operaciones anteriores del stream.
  + **toArray:**  crea un array de elementos que contiene los resultados de las operaciones anteriores del stream.
  + **findFirst:** encuentra el primer elemento del stream, con base en las operaciones intermedias.
  + **findAny:** encuentra cualquier elemento del stream, con base en las operaciones intermedias anteriores.
  + **anyMatch:** determina si alguno de los elementos del stream coincide con alguna condición especificada.
  + **allMatch:** determina si todos los elementos del stream, coinciden con una condición especificada.