Problema exemplo

Fazer um programa que, a partir de uma lista de produtos, calcule a soma dos preços somente dos produtos cujo nome começa com "T".

```
List<Product> list = new ArrayList<>();

list.add(new Product("Tv", 900.00));
list.add(new Product("Mouse", 50.00));
list.add(new Product("Tablet", 350.50));
list.add(new Product("HD Case", 80.90));
```

1250.50

https://github.com/acenelio/lambda5-java

Stream

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Stream

- É uma sequencia de elementos advinda de uma fonte de dados que oferece suporte a "operações agregadas".
  - Fonte de dados: coleção, array, função de iteração, recurso de E/S
- Sugestão de leitura:
  http://www.oracle.com/technetwork/pt/articles/java/streams-api-java-8-3410098-ptb.html

Características

- Stream é uma solução para processar sequências de dados de forma:
  - Declarativa (iteração interna: escondida do programador)
  - Parallel-friendly (imutável -> thread safe)
  - Sem efeitos colaterais
  - Sob demanda (lazy evaluation)
- Acesso sequencial (não há índices)
- Single-use: só pode ser "usada" uma vez
- **Pipeline**: operações em streams retornam novas streams. Então é possível criar uma cadeia de operações (fluxo de processamento).

Operações intermediárias e terminais

- O pipeline é composto por zero ou mais operações intermediárias e uma terminal.
- Operação intermediária:
  - Produz uma nova streams (encadeamento)
  - Só executa quando uma operação terminal é invocada (lazy evaluation)
- Operação terminal:
  - Produz um objeto não-stream (coleção ou outro)
  - Determina o fim do processamento da stream

Operações intermediárias

- filter
- map
- flatmap
- peek
- distinct
- sorted
- skip
- limit (*)

** short-circuit*

Operações terminais

- forEach
- forEachOrdered
- toArray
- reduce
- collect
- min
- max
- count
- anyMatch (*)
- allMatch (*)
- noneMatch (*)
- findFirst (*)
- findAny (*)

** short-circuit*

Criar uma stream

- Basta chamar o método `stream()` ou `parallelStream()` a partir de qualquer objeto Collection.
  https://docs.oracle.com/javase/10/docs/api/java/util/Collection.html

- Outras formas de se criar uma stream incluem:
  - Stream.of
  - Stream.ofNullable
  - Stream.iterate

Demo - criação de streams

```
List<Integer> list = Arrays.asList(3, 4, 5, 10, 7);
Stream<Integer> st1 = list.stream();
System.out.println(Arrays.toString(st1.toArray()));


Stream<String> st2 = Stream.of("Maria", "Alex", "Bob");
System.out.println(Arrays.toString(st2.toArray()));


Stream<Integer> st3 = Stream.iterate(0, x -> x + 2);
System.out.println(Arrays.toString(st3.limit(10).toArray()));


Stream<Long> st4 = Stream.iterate(new long[]{ 0L, 1L }, p->new long[]{ p[1], p[0]+p[1] }).map(p -> p[0]);
System.out.println(Arrays.toString(st4.limit(10).toArray()));
```

Pipeline (demo)

http://educandoweb.com.br

Prof. Dr. Nelio Alves