



High performance. Delivered.

# Application Delivery Fundamentals: Java

Module 12: Programação Funcional (Lambda e Streams)

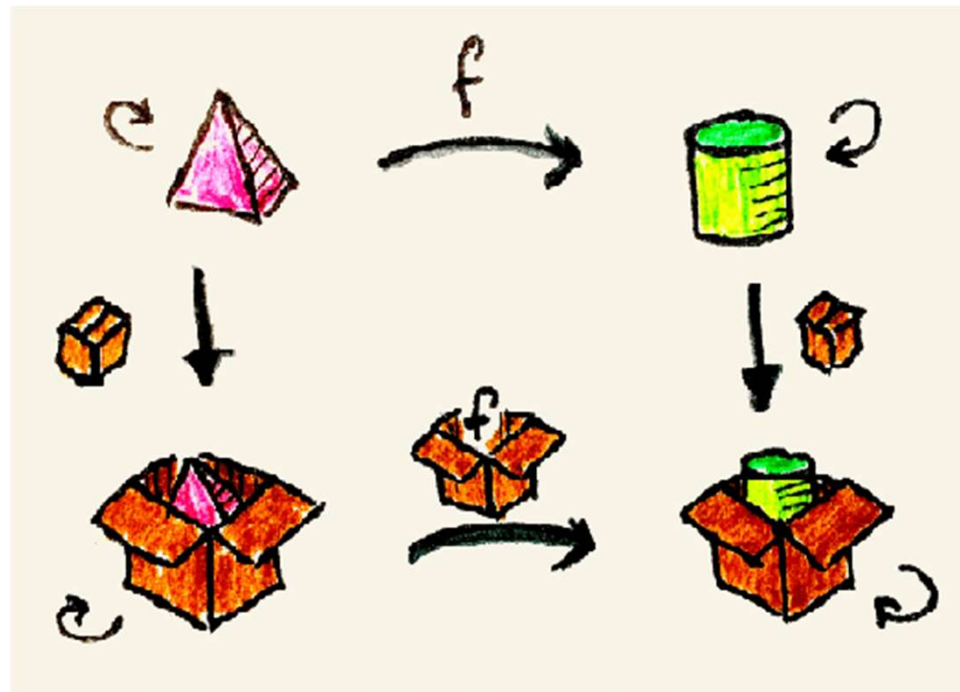
# Programação Funcional

**“PROGRAMAÇÃO FUNCIONAL”**



# Programação Funcional: É um paradigma de programação;

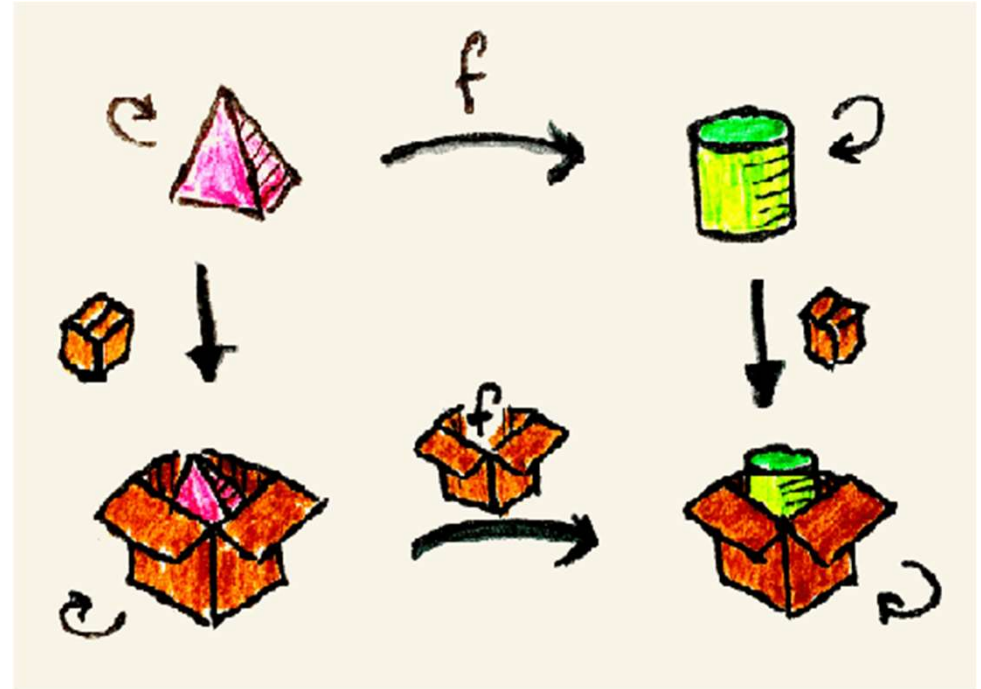
- No paradigma funcional eu não digo ao meu código o que ele deve fazer, quando e como. Não irei desenvolvê-lo passo a passo. Eu penso meu código como uma sequência de funções e/ou passos, as quais de maneira composta irão resolver meu problema.



[illegible]

# Programação Funcional: É um paradigma de programação;

- Pense da seguinte forma: eu tenho um dado de entrada e preciso transformá-lo em um dado de saída.
- Usando PF eu vou abstrair as lógicas de transformações do meu código em funções, e usá-las no momento oportuno para transformar este meu dado.



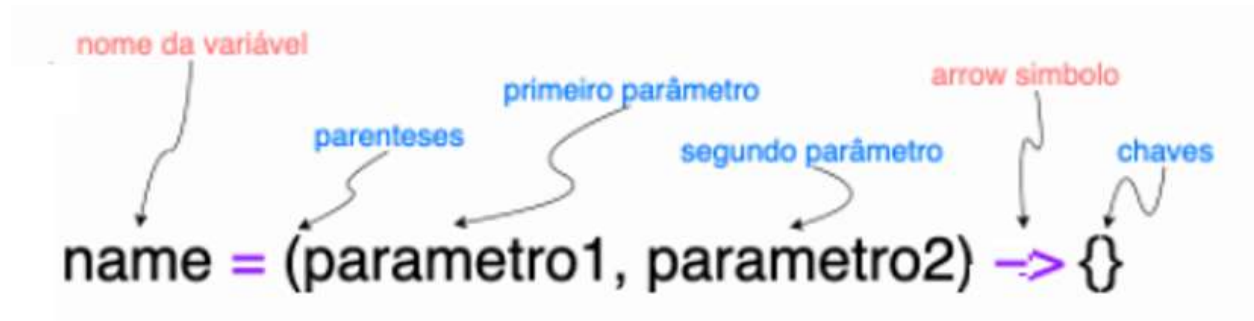
[illegible]

[illegible]

- A expressão lambda é tratada como uma função, portanto, o compilador não cria o arquivo .class.
- Habilite para tratar a funcionalidade como um argumento de método ou o código como dados.
- Uma função que pode ser criada sem pertencer a nenhuma classe.
- Uma expressão lambda pode ser transmitida como se fosse um objeto e executada sob demanda;

- Para fornecer a implementação da interface funcional.
- Menos codificação.

# Syntaxe



- Zero parameter:

```
() -> System.out.println("Zero parameter lambda");
```

- **One parameter:-**

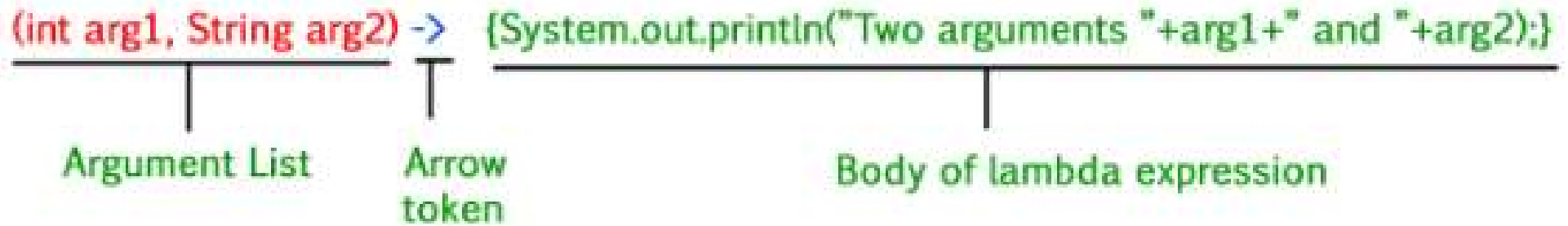
```
(p) -> System.out.println("One parameter: " + p);
```

- **Multiple parameters :**

```
(p1, p2) -> System.out.println("Multiple parameters: " + p1 + ", " + p2)
```

[illegible]

# Syntaxe



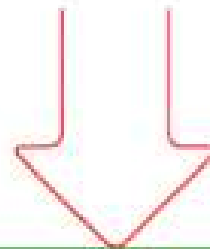


# LAMBDA

2f48

196768207

```
Arrays.sort(dogArray, new Comparator<Dog>() {  
    @Override  
    public int compare(Dog o1, Dog o2) {  
        return o1.getWeight() - o2.getWeight();  
    }  
});
```



Lambda Expression

```
Arrays.sort(dogArray, (m, n) -> m.getWeight() - n.getWeight());
```

1. Uma lista separada por vírgula de parâmetros formais entre parênteses. Nesse caso, é (Dog m, Dog n)
2. O token de seta ->
3. Um corpo, que consiste em uma única expressão ou em um bloco de instruções.

Nesse caso, é uma expressão única - Integer.compare (m.getWeight (), n.getWeight ())

[illegible]

# Syntaxe

```
List<Integer> intSeq = Arrays.asList(1,2,3);

intSeq.forEach(x -> {
    int y = x * 2;
    System.out.println(y);
});
```

[illegible]

Copyright © 2023 Accenture All Rights Reserved. 11

Um **closure** nos dá acesso à função externa de uma função interna.

[illegible]

Copyright © 2023 Accenture All Rights Reserved.



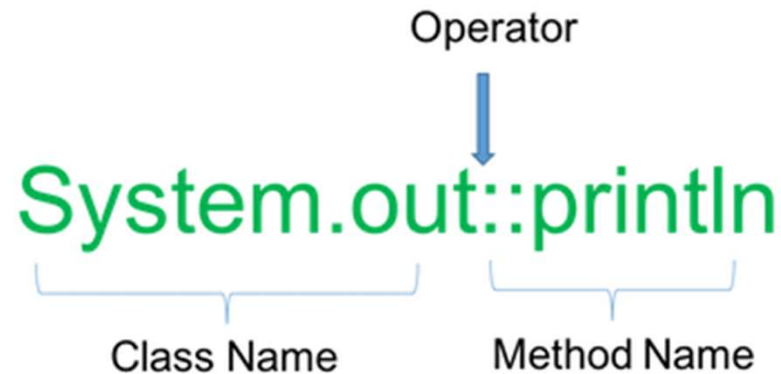
[illegible]

20 }

[illegible]

Copyright © 2023 Accenture All Rights Reserved. 15

## Sintaxe da chamada do método por referência



```
array.forEach(System.out::println);
```

MethodReferencesExamples1,2 e 3.java

## Lambda Expression vs Method Reference

Lambda Expression	Method Reference
<code>s -&gt; s.toString()</code>	<code>String :: toString</code>
<code>s -&gt; s.toLowerCase()</code>	<code>String :: toLowerCase</code>
<code>s -&gt; s.length()</code>	<code>String :: length</code>
<code>(i1,i2) -&gt; i1.compareTo(i2)</code>	<code>Integer :: compareTo</code>
<code>(s1,s2) -&gt; s1.compareTo(s2)</code>	<code>String :: compareTo</code>

LambdaVSmethod.java  
MapCollectStream.java

[illegible]

# “STREAM”



[illegible]

214869676820706572666f726d616e63652e2044656c6976657265642e2f486967682070657266f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820706572666f726d616e63652e2044656c6976657265642e2f4869676820

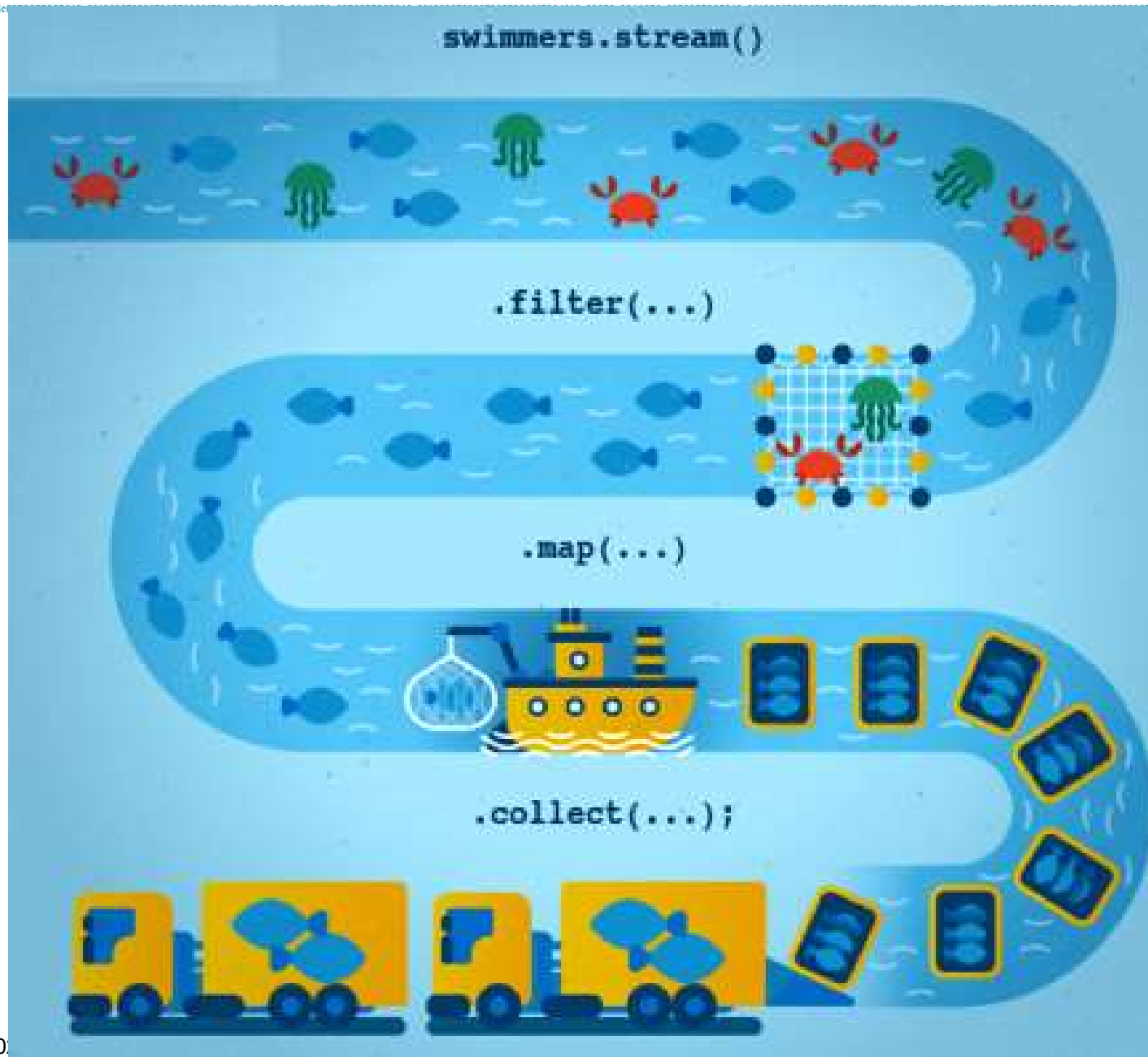
[illegible]

# O que é Stream Processing?

- Sequência de elementos de uma fonte de dados que suporta operações de agregação;
- **Sequência de elementos:** Uma stream oferece uma interface para um conjunto de valores sequenciais de um tipo de elemento particular. Apesar disso, as streams não armazenam elementos; estes são calculados sob demanda.
- **Fonte de dados:** As streams tomam seu input de uma fonte de dados, como coleções, matrizes ou recursos de E/S.
- **Operações de agregação:** As streams suportam operações do tipo SQL e operações comuns à maioria das linguagens de programação funcionais, como filter, map, reduce, find, match e sorted, entre outras.



# STREAM



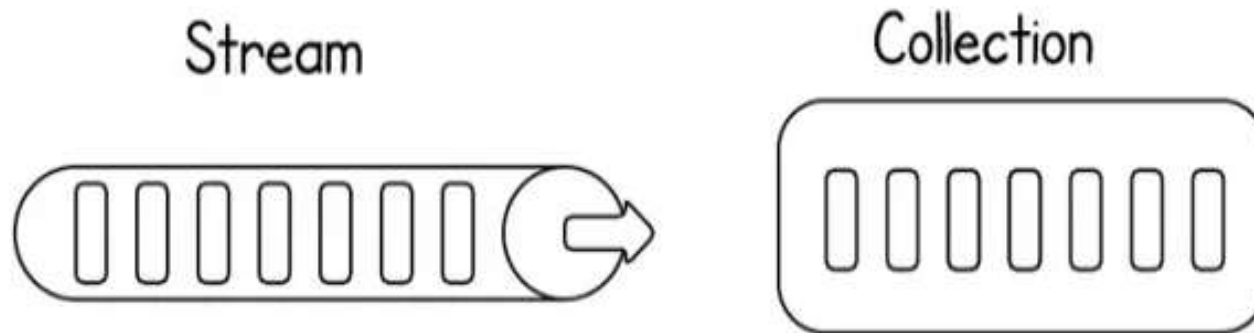


[illegible]

## Intermediate Operations

# STREAM vs Collection

- As **collection** se referem a dados enquanto as **stream** se referem a cálculos;
- **java.util.stream.Stream**



[illegible]

- C



[illegible]

- **Filteragem:**

- **filter(Predicate):** Retorna uma stream incluindo todos os elementos que coincidem com o predicado indicado.
- **distinct:** Retorna uma stream com elementos únicos;
- **limit(n):** Retorna uma stream cuja tamanho máximo é **n**.
- **skip(n):** Retorna uma stream em que se descartaram os primeiros **n** elementos.

[illegible]

- Copyright © 2023 Accenture All Rights Reserved.



[illegible]

- 

[illegible]

- **Mapeamento**

- Projeta os elementos da **stream** em outro formato;
- A função é aplicada a cada elemento, que é "mapeado".

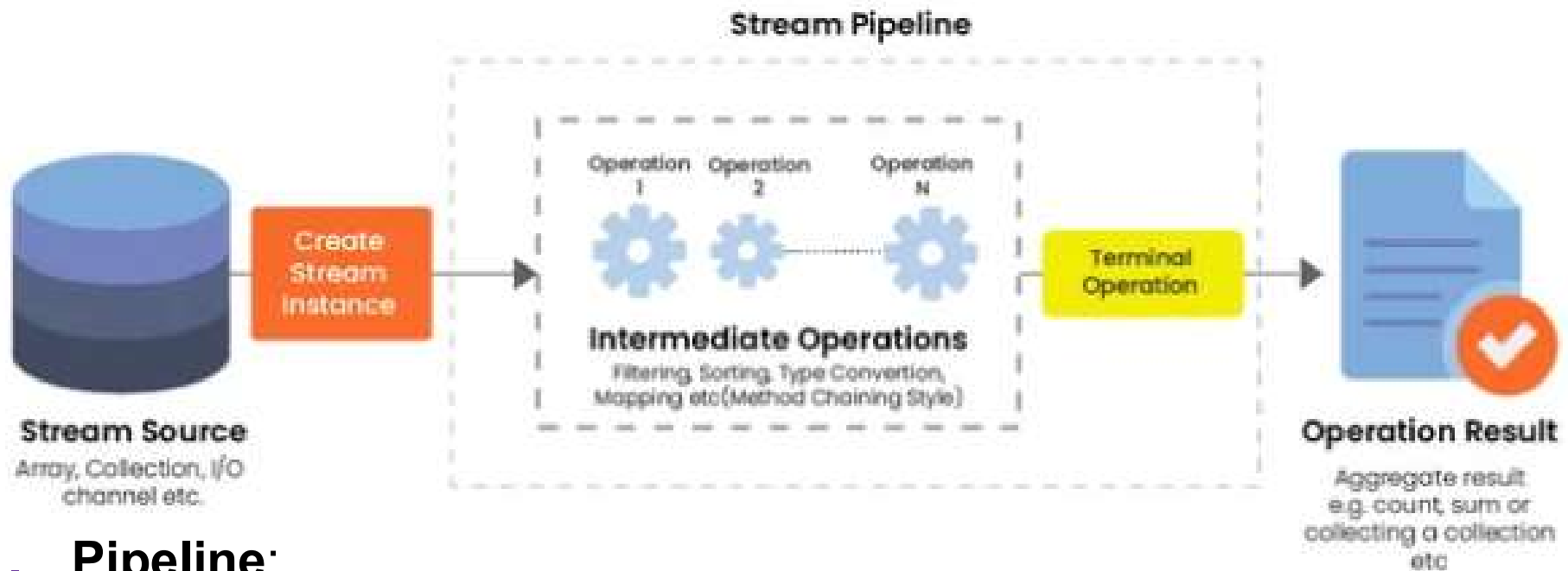
```
List<String> words = Arrays.asList("Oracle", "Java", "Magazine");
    List<Integer> wordLengths = words.stream()
        .map(String::length)
        .collect(toList());
```

[illegible]

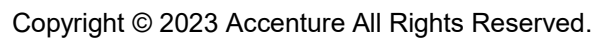
- **Redução**

- Usamos **collect** para combinar todo o conjunto de elementos de uma Stream em um objeto List.
- (ex. StreamCollect.java)

```
List<String> palavra = Arrays.asList("Oracle", "Java", "Magazine");  
List<Integer> tamanhos = palavra.stream()  
    .map(String::length)  
    .collect(Collectors.toList());
```



1. **Pipeline:**
2. É o processo pelo qual uma instrução de processamento é subdividido em etapas;
3. É utilizada para acelerar a velocidade de operação da CPU;



-

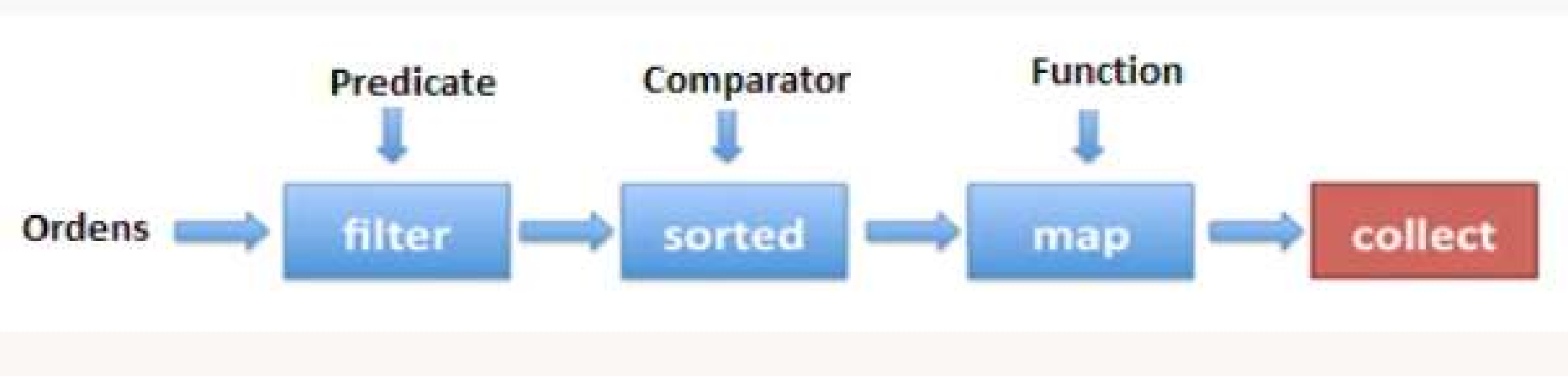


[illegible]

Terminal operation

[illegible]

```
List<Integer> ordensIDs =      Ordem.stream()
    .filter(o -> o.getType() == Ordem.ATIVACAO)
    .sorted(comparing(Ordem::getValue).reversed())
    .map(Ordem::getId)
    .collect(toList());
```



[illegible]



## Programar baseado em funções;

**Uma função seta que pode ser criada sem pertencer a nenhuma classe;**

# Um stream é a maneira de abstrair e especificar como processar uma agregação

[illegible]

- [illegible]

