



 **VEM SER**
DBC



Paradigma Funcional



Sumário

- Paradigma Funcional no Java
- Lambda
- Interfaces Funcionais
- Funções
- Stream API
- Generics

Paradigma Funcional no Java

<https://www.baeldung.com/java-functional-programming>

Paradigma Funcional no Java

- Programação funcional é um estilo de escrever programas de computador que trata os cálculos como avaliações de funções matemáticas.

Paradigma Funcional no Java

- Programação funcional é um estilo de escrever programas de computador que trata os cálculos como avaliações de funções matemáticas.
- Uma função é uma expressão que relaciona um conjunto de entrada a um conjunto de saída.

Paradigma Funcional no Java

- Programação funcional é um estilo de escrever programas de computador que trata os cálculos como avaliações de funções matemáticas.
- Uma função é uma expressão que relaciona um conjunto de entrada a um conjunto de saída.
- É importante ressaltar que a saída de uma função depende apenas de sua entrada. Mais interessante, podemos compor duas ou mais funções juntas para obter uma nova função.

Lambda

- Na década de 1930, o matemático Alonzo Church desenvolveu **um sistema formal para expressar cálculos baseados na abstração de funções**.

<https://www.baeldung.com/java-functional-programming>

Lambda

- Na década de 1930, o matemático Alonzo Church desenvolveu **um sistema formal para expressar cálculos baseados na abstração de funções** .
- Este modelo universal de computação veio a ser conhecido como [Cálculo Lambda](#) .

Lambda

- Na década de 1930, o matemático Alonzo Church desenvolveu **um sistema formal para expressar cálculos baseados na abstração de funções**.
- Este modelo universal de computação veio a ser conhecido como [Cálculo Lambda](#).
- O cálculo lambda teve um tremendo impacto no desenvolvimento da teoria das linguagens de programação

<https://www.baeldung.com/java-functional-programming>

Lambda

- Na década de 1930, o matemático Alonzo Church desenvolveu **um sistema formal para expressar cálculos baseados na abstração de funções**.
- Este modelo universal de computação veio a ser conhecido como [Cálculo Lambda](#).
- O cálculo lambda teve um tremendo impacto no desenvolvimento da teoria das linguagens de programação
- Como o cálculo lambda se concentra na composição de funções, as linguagens de programação funcional fornecem maneiras expressivas de compor software na composição de funções.

<https://www.baeldung.com/java-functional-programming>

Interfaces Funcionais

```

@FunctionalInterface
public interface Funcao {
    String gerar(String valor);
}

Funcao colocarPrefixoSenhor = new Funcao() {
    @Override
    public String gerar(String valor) {
        return "Sr. " + valor;
    }
};

System.out.println(colocarPrefixoSenhor.gerar("Maicon"));
  
```

Interfaces Funcionais

@FunctionalInterface

```
public interface Funcao {
    String gerar(String valor);
}
```

```
Funcao funcao = valor -> "Sr. " + valor;
```

```
System.out.println(funcao.gerar("Maicon"));
```

Let's practice;

Exercício #1

- Criar uma interface funcional **Calculo** que retorne um número inteiro e receba dois parâmetros.
- Criar uma nova classe **Main** para testar onde:
 - Deve conter a implementação da interface funcional Calculo com a operação de **soma**.
 - Deve conter a implementação da interface funcional Calculo com a operação de **multiplicação**.

Técnicas de Programação Funcional

- Funções
- Streams

Composição de Função

```
Function<Double, Double> log = (value) -> Math.log(value);
Function<Double, Double> sqrt = (value) -> Math.sqrt(value);
Function<Double, Double> logThenSqrt = sqrt.compose(log);
System.out.println(logThenSqrt.apply(3.14));
// Output: 1.06
Function<Double, Double> sqrtThenLog = sqrt.andThen(log);
System.out.println(String.valueOf(sqrtThenLog.apply(3.14)));
// Output: 0.57
```


Let's practice;

Exercício #2

- Utilize a interface **Function do java** faça:
 - Criar uma função que realize a multiplicação por 10 de algum número.
 - Criar outra função que realize a raiz quadrada.
- Crie uma outra função e junte as duas operações (andThen).
- Realize o cálculo e teste com a operação.

Stream API

- Manipulação de coleções de dados

Stream API

- Manipulação de coleções de dados
- Lançada à partir do Java 8

Stream API

- Manipulação de coleções de dados
- Lançada à partir do Java 8
- Iterar sobre essas coleções de objetos e, a cada elemento, realizar alguma ação, seja ela de filtragem, mapeamento, transformação, etc.

Stream API

```

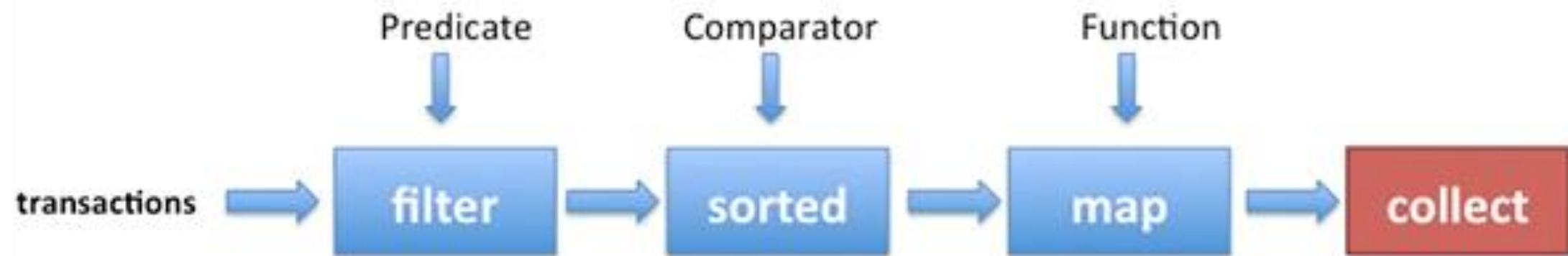
List<Transaction> transactionList = new ArrayList<>();
for (Transaction t : transactions) {
    if (t.getType() == Transaction.GROCERY) {
        transactionList.add(t);
    }
}
Collections.sort(transactionList, new Comparator() {
    public int compare(Transaction t1, Transaction t2) {
        return t2.getValue().compareTo(t1.getValue());
    }
});
List<Integer> transactionIds = new ArrayList<>();
for (Transaction t : transactionList) {
    transactionIds.add(t.getId());
}
  
```

Stream API

```
List<Transaction> transactionList = new ArrayList<>();
for (Transaction t : transactions) {
    if (t.getType() == Transaction.GROCERY) {
        transactionList.add(t);
    }
}
Collections.sort(transactionList, new Comparator() {
    public int compare(Transaction t1, Transaction t2) {
        return t2.getValue().compareTo(t1.getValue());
    }
});
List<Integer> transactionIds = new ArrayList<>();
for (Transaction t : transactionList) {
    transactionIds.add(t.getId());
}
```

```
List<Integer> transactionIds = transactions.stream()
    .filter(t -> t.getType() == Transaction.GROCERY)
    .sorted(comparing(Transaction::getValue).reversed())
    .map(Transaction::getId)
    .collect(toList());
```

Fluxo



<https://www.oracle.com/br/technical-resources/articles/java/processing-streams-java-se-8.html>

Let's practice;

STREAMS

- **Optional<T>** => classe responsável por tratar os valores nulos
- **.findFirst()** => retorna o primeiro elemento
- **.peek()** => executa o comando passado na expressão
- **.map(obj => outroObj)** => transforma um objeto em outro

Exercício

- Utilizando a StreamAPI, baixe o código Stream.java que será disponibilizado e faça as operações descritas nos comentários do arquivo.
- Para cada comentário, imprimir a lista com o resultado da operação

GENERICs (o famoso <T>)

- Responsável por generalizar códigos redundantes na nossa app
- Como é o caso de casting excessivo
- Este foi introduzido desde o Java SE 5.0

Let's practice;

#Exercicio

- Na pasta “modulo1” criar um projeto chamado “sistema-aluno”
- Elaborar um programa que cadastre alunos com os seguintes dados:
 - Identificador, nome, idade, cidade, nota1, nota2 e nota3
- Crie uma lista com esses alunos cadastrados
- (para facilitar, deixe alguns previamente cadastrados nessa lista)
- Com essa lista faça alguns relatórios que o usuário poderá solicitar pelo menu interativo
 - Relatório com todos os alunos que moram no Acre
 - Relatório com alunos com mais de 18 anos
 - Média das notas $[(n1 + n2 + n3) / 3]$ ordenada da maior para a menor média. Além disso, trazer todos os dados do aluno ao lado da média (usar a função map (com um objeto com os dados media e aluno))
 - Identificadores de cidades