

Interfaces funcionais

Facilitando o código da API



DIGITAL
INNOVATION
ONE

Parte 1: Funções de alta ordem

Desenvolvimento avançado em Java

```

FuncaoAltaOrdem.java x
1 package interfaces_funcionais;
2
3 public class FuncaoAltaOrdem {
4     public static void main(String[] args) {
5         Calculo soma = (a, b) -> a+b;
6         System.out.println(executarOperacao(soma, a: 1, b: 3));
7     }
8     @ public static int executarOperacao(Calculo calculo, int a, int b){
9         return calculo.calcular(a,b);
10    }
11    @FunctionalInterface
12    interface Calculo{
13        public int calcular(int a, int b);
14    }
15    //POR PARAMETRO RECEBE OUTRA FUNÇÃO
16    // OU QUE ELA RETORNA UMA FUNÇÃO
17 }
18

```

```
> Task :FuncaoAltaOrdem.main()
```

```
4
```



DIGITAL
INNOVATION
ONE

Funções de alta ordem

É uma função que retorna uma função ou que recebe uma função como parâmetro.

```

FuncaoAltaOrdem.java x
1  package interfaces_funcionais;
2
3  public class FuncaoAltaOrdem {
4      public static void main(String[] args) {
5          Calculo soma = (a, b) -> a+b;
6          Calculo subtracao = (a, b) -> a-b;
7          Calculo divisao = (a, b) -> a/b;
8          Calculo multi = (a, b) -> a*b;
9
10         System.out.println(executarOperacao(soma, a: 1, b: 3));
11         System.out.println(executarOperacao(subtracao, a: 4, b: 3));
12         System.out.println(executarOperacao(divisao, a: 4, b: 2));
13         System.out.println(executarOperacao(multi, a: 7, b: 3));
14
15     }
16     public static int executarOperacao(Calculo calculo, int a, int b){
17         return calculo.calcular(a,b);
18     }
19     @FunctionalInterface
20     interface Calculo{
21         public int calcular(int a, int b);
22     }
23     //POR PARAMETRO RECEBE OUTRA FUNÇÃO
24     // OU QUE ELA RETORNA UMA FUNÇÃO
25 }
26

```

```

> Task :FuncaoAltaOrdem.main()
4
1
2
21

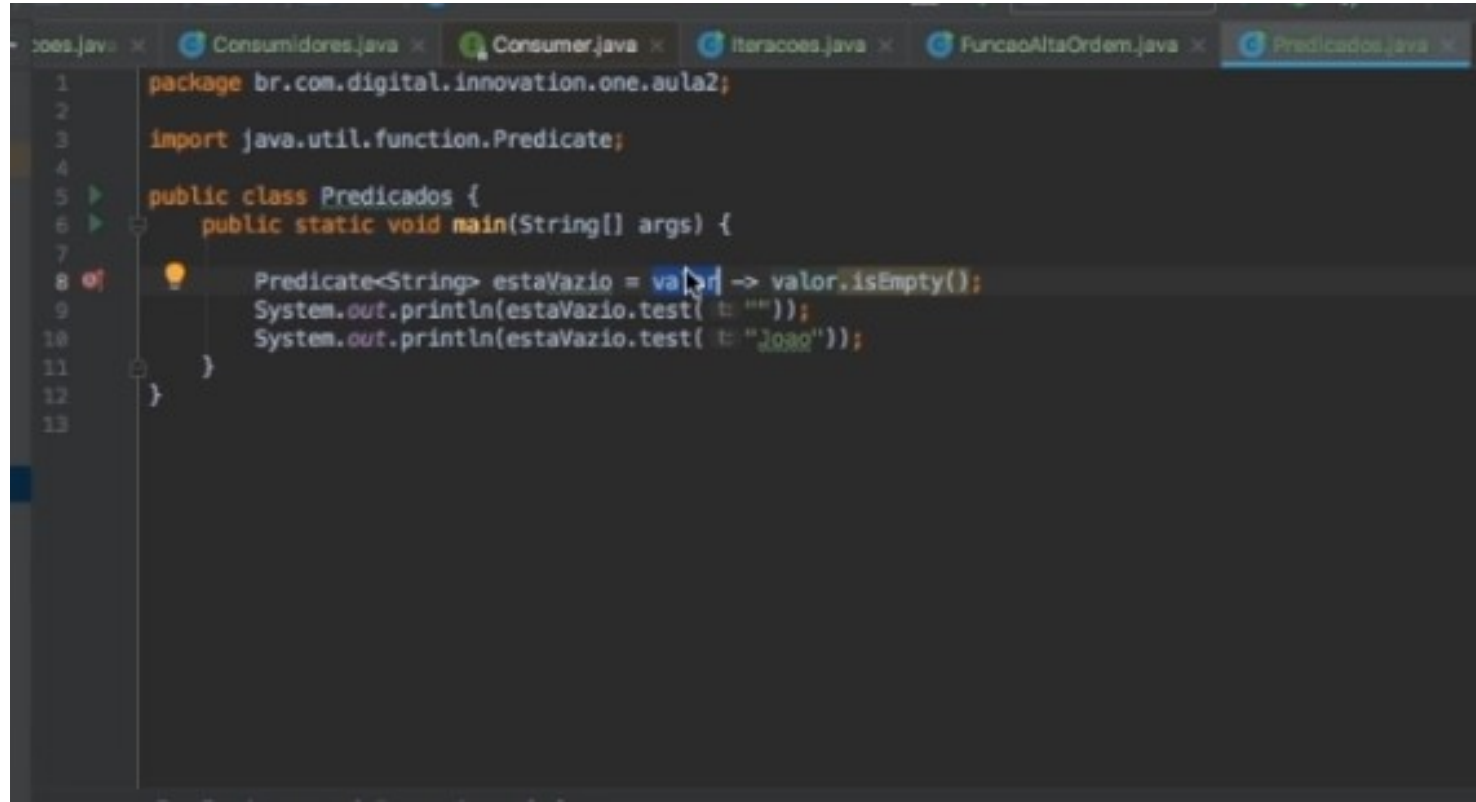
```

CONSUMIDORES

```
Consumidores.java x
1  package interfaces_funcionais;
2
3  import java.util.function.Consumer;
4
5  public class Consumidores {
6      public static void main(String[] args) {
7          // - Apenas
8          // - UTILIZAR o parametro da forma que ele foi recebido;
9
10         Consumer<String> imprimeUmaFrase = frase -> System.out.println(frase);
11         imprimeUmaFrase.accept(t: "Hello World!");
12     }
13 }
14
```

```
> Task :Consumidores.main()
Hello World!
```

PREDICADOS

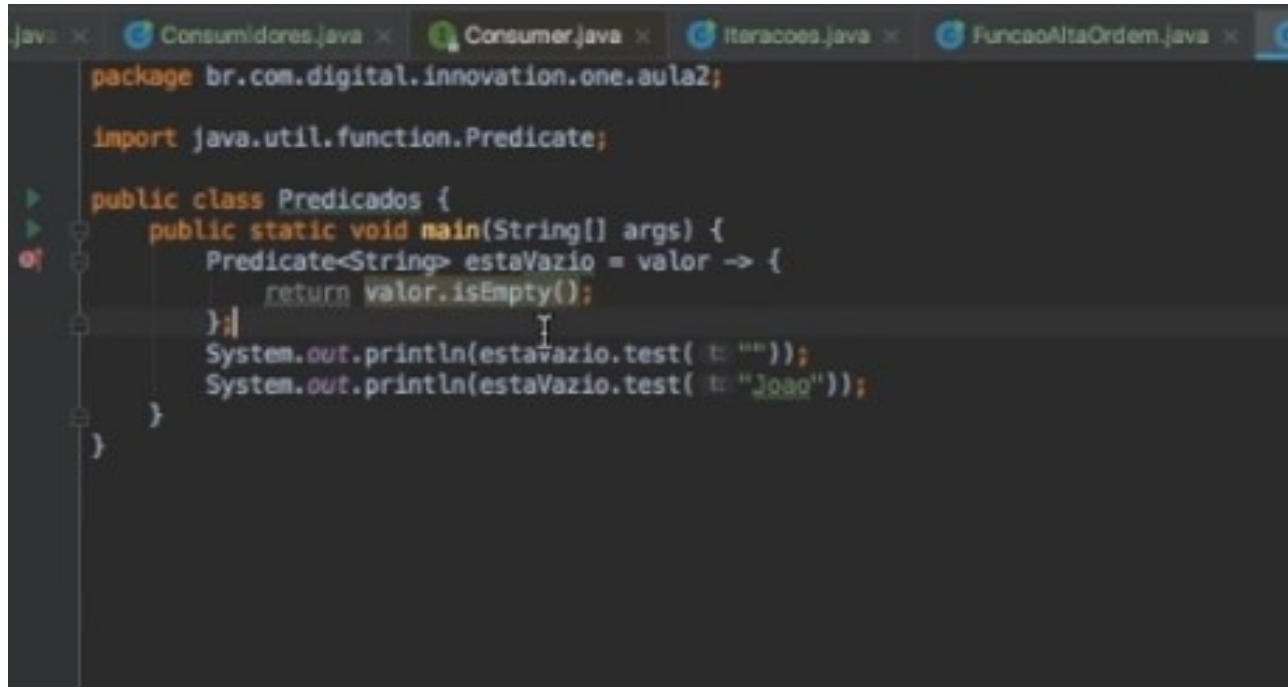


The screenshot shows an IDE with several tabs open: `boes.java`, `Consumidores.java`, `Consumer.java`, `Iteracoes.java`, `FuncaoAltaOrdem.java`, and `Predicados.java`. The `Predicados.java` tab is active, displaying the following code:

```
1 package br.com.digital.innovation.one.aula2;
2
3 import java.util.function.Predicate;
4
5 public class Predicados {
6     public static void main(String[] args) {
7
8         Predicate<String> estaVazio = valor -> valor.isEmpty();
9         System.out.println(estaVazio.test(""));
10        System.out.println(estaVazio.test("Joao"));
11    }
12 }
13
```

A lightbulb icon is visible next to line 8, indicating a suggestion or warning. The code defines a `Predicate<String>` named `estaVazio` that tests if a string is empty, and then uses it to test an empty string and the string "Joao".

PREDICADOS



The screenshot shows an IDE with several tabs open: 'Consumidores.java', 'Consumer.java', 'Iteracoes.java', 'FuncaoAltaOrdem.java', and 'Predicados.java'. The 'Predicados.java' tab is active, displaying the following Java code:

```
package br.com.digital.innovation.one.aula2;

import java.util.function.Predicate;

public class Predicados {
    public static void main(String[] args) {
        Predicate<String> estaVazio = valor -> {
            return valor.isEmpty();
        };
        System.out.println(estaVazio.test( t: ""));
        System.out.println(estaVazio.test( t: "Joao"));
    }
}
```


SUPRIDORES

Supplier – não recebe nenhum parâmetro e retorna um valor.

STREAM

Stream – não tem método construtor.

Stream recebe um array de valores.

O método `filter()` recebe um “predicate”.

EXERCÍCIOS

Interfaces funcionais

Qual a classe do Java 8 que dá suporte a listas, conjuntos, mapas e arrays?

Set.

Object.

Stream. ✓

List.

Map.

PRÓXIMA PERGUNTA

EXERCÍCIOS

Interfaces funcionais

São interfaces funcionais as que possuem:

2 métodos estáticos.

Um método default.

Um método abstrato. ✓

2 métodos abstratos.

Um método estático.

PRÓXIMA PERGUNTA

EXERCÍCIOS

Interfaces funcionais

Qual o método utilizado para transformar dados em um stream?

filter



toString

forEach

calculate

map



PRÓXIMA PERGUNTA

EXERCÍCIOS

Interfaces funcionais

Marque a alternativa que apresenta a assinatura do método abstrato da Interface Consumer:

`T apply(R r);`



`void get(T t);`

`R apply(T t);`

`void accept(T t);`



`void apply(R r);`

PRÓXIMA PERGUNTA

EXERCÍCIOS

Interfaces funcionais

Qual o nome método utilizado para iterar dados em um stream que substitui o "for"?

forEach



calculate

filte"

map

toString

PRÓXIMA PERGUNTA

EXERCÍCIOS

Interfaces funcionais

Marque a alternativa que apresenta a assinatura do método abstrato da Interface Supplier:

`T get()`



`T get(T t)`

`T get(R r)`

`void get(T t)`

`T apply()`

PRÓXIMA PERGUNTA

EXERCÍCIOS

Interfaces funcionais

Qual o nome do método utilizado para filtrar dados em um stream?

forEach

calculate

filter



toString

map

PRÓXIMA PERGUNTA

EXERCÍCIOS

Interfaces funcionais

O que é uma função de alta ordem ?

É uma função que retorna uma função ou que recebe uma função como parâmetro.



É uma interface funcional.

É uma lambda.

A primeira função a ser invocada em um código.

É uma função que possui recursividade.

PRÓXIMA PERGUNTA

EXERCÍCIOS

Interfaces funcionais

Marque a alternativa que apresenta a assinatura do método abstrato da Interface Predicate:

`boolean test(T t);`



`T apply(R r);`

`void apply(R r);`

`T test(boolean test);`

`boolean test();`

PRÓXIMA PERGUNTA

EXERCÍCIOS

Interfaces funcionais

Marque a alternativa que apresenta a assinatura do método abstrato da Interface Function:

`void apply(R r);`

`void get(T t);`

`T apply(R r);`

`R apply(T t);` ✓

`T get(R r);`

FINALIZAR