



POO

Programação Orientada a
Objetos



Professor Rômulo C. Silvestre

SENAI TAGUATINGA



AGENDA

- Pacote
- Classe
- Atributo
- Get e set
- Workers
- Construtores
- Sobrecarga
- POO avançado (abstrato)



PACOTE

package

CLASSE



CECIL LIMA
Presidente



YARA LIMA
CEO



GABRIELLE GONÇALVES
COO



HENRIQUE CASTRO
Vice-presidente de
Marketing

OBJETOS



DOMINIQUE OLIVEIRA
Vice-presidente de Produto



MICA PEREIRA
Estrategista de SEO



CRIS BARROS
Designer de Produto



ÁLVARO ALMARAZ
Desenvolvedor
de Conteúdo



ATRIBUTOS

10 anos	20 anos
Cecil Lima	Gabrielle Gonçalves
Lara Cardoso	Henrique Castro

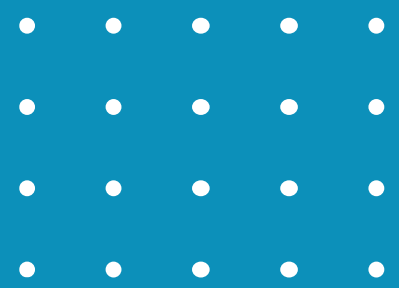




Atributo idade

inteiro

MÉTODOS



GET E SET

métodos

Método GET

Método SET

O que é void?

MÉTODOS WORKERS

BO

- BUSINESS OBJECT

DAO

- Data Access Object

DTO

- Data Transfer Object



CONTRUTORES

SOBRECARGA

construtor	método
overload	overload
Não Muda nome	Muda tipo e não muda o nome
Muda parâmetros	Muda parâmetros

POO ADVANCED

- Generalização
- Especialização
- Herança
- Herança simples x Herança múltipla
- Abstração
- Classe Abstrata
- Método Abstrato
- Interfaces
- Implementação múltipla de interface
- Interface a partir do Java 8
- Interface aplicada ao SWING
- Lambda e Eventos no Swing



GENERALIZAÇÃO

ESPECIALIZAÇÃO

Abstração

- O que realmente é importante

Abstrata

- Não pode instanciar

Final

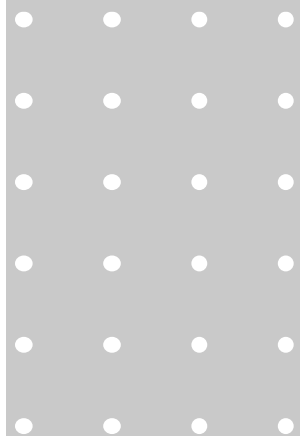
- Não pode ter filhos





Contoso

HERANÇA₃




HERANÇA


Simples ou múltipla
















 Package Explorer ×

▼  revisaofinalpoo

>  JRE System Library [JavaSE-1.8]

 src

Projeto

- ▼  revisaofinalpoo
 - >  JRE System Library [JavaSE-1.8]
 - ▼  src
 -  br.com.contoso.bo  Negócio
 -  br.com.contoso.dao  Banco de Dados
 -  br.com.contoso.dto  Dados
 -  br.com.contoso.projetofinal  Principal
 -  br.com.contoso.view  Telas

Pacotes

```
1 package br.com.contoso.projetoFinal;  
2  
3 public class Principal {  
4  
5     public static void main(String[] args) {  
6         // TODO Auto-generated method stub  
7         System.out.println("Alô Mundo!");  
8     }  
9  
10 }  
11
```

Console ×
<terminated> Principal (8) [Java Application] C:\Users\romul\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.5.v20221102-0933\jre\bin\javaw.exe (2 de f

Alô Mundo!

Alô Mundo (static void main())

```
1 package br.com.contoso.dto;
2
3 public class Funcionario {
4
5     private String nome;
6     private String cargo;
7     private int idade;
8
9
10 }
```

Classe

```
private String nome;  
private String cargo;  
private int idade;
```

Atributos

28 → VALOR

```
public Funcionario(String nome, String cargo, int idade) {  
    this.nome = nome;  
    this.cargo = cargo;  
    this.idade = idade.;  
}
```

Porque o "." não funcionou?

Porque int idade é um variável de valor.

Atributo de valor


```
public Funcionario(String nome, String cargo) {  
    this.nome = nome;  
    this.cargo = cargo;  
    this.idade = idade;  
}
```

- concat(String str) : String - String
- formatted(Object... args) : String - String
- indent(int n) : String - String
- intern() : String - String
- repeat(int count) : String - String
- replace(char oldChar, char newChar) : String - String
- replace(CharSequence target, CharSequence replacement) : String - String
- replaceAll(String regex, String replacement) : String - String
- replaceFirst(String regex, String replacement) : String - String
- resolveConstantDesc(Lookup lookup) : String - String
- strip() : String - String

Press 'Ctrl+Space' to show Template Proposals

ATRIBUTO É UMA REFERÊNCIA

```
public Funcionario(String nome, String cargo, int idade) {  
    this.nome = nome.toUpperCase();  
    this.cargo = cargo;  
    this.idade = idade;  
}
```

Atributo de Referência

```
1 package br.com.contoso.projetofinal;
2 import br.com.contoso.dto.*;
3
4 public class Principal {
5
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8         System.out.println("Sistema Contoso!")
9
10    }
11
12 }
13
```

Importar um pacote

```
public static void main(String[] args) {  
    // TODO Auto-generated method stub  
    System.out.println("Sistema Contoso!");  
    Funcionario funcionario = new Funcionario();  
}
```

OBJETO

É uma instância de uma classe.

Public
Private
Protected
package

Nível de Encapsulamento

```
5 public String nome;  
6 public String cargo;  
7 public int idade;  
8
```

public

Funcionario é um **objeto**

"" É o acesso

nome é atributo

"=" atribuição

"Rômulo Cesar Silvestre" é um string

```
funcionario.nome = "Rômulo Cesar Silvestre";  
funcionario.cargo = "Desenvolvedor Java";  
funcionario.idade = 43;
```

ATRIBUTO

```

1 package br.com.contoso.projetoFinal;
2 import br.com.contoso.dto.*;
3
4 public class Principal {
5
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8         System.out.println("Sistema Contoso!");
9         Funcionario funcionario = new Funcionario();
10        //chamar e definir valores para os meus atributos
11        funcionario.nome = "Rômulo Cesar Silvestre";
12        funcionario.cargo = "Desenvolvedor Java";
13        funcionario.idade = 43;
14
15        System.out.println("Nome:"+funcionario.nome);
16        System.out.println("Cargo:"+funcionario.cargo);
17        System.out.println("Idade:"+funcionario.idade);
18
19    }
20
21 }
22

```

Console ×

```

<terminated> Principal (8) [Java Application] C:\Users\romul\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.f
Sistema Contoso!
Nome:Rômulo Cesar Silvestre
Cargo:Desenvolvedor Java
Idade:43

```

Atribuiu com “=”


```
private String nome;  
private String cargo;  
private int idade;
```

ENCAPSULAMENTO

```

1 package br.com.contoso.dto;
2
3 public class Funcionario {
4
5     private
6     private
7     private
8
9     public
10
11 }
12 public
13
14 }
15
16 }
17
18
19
20 }
21
22
23

```

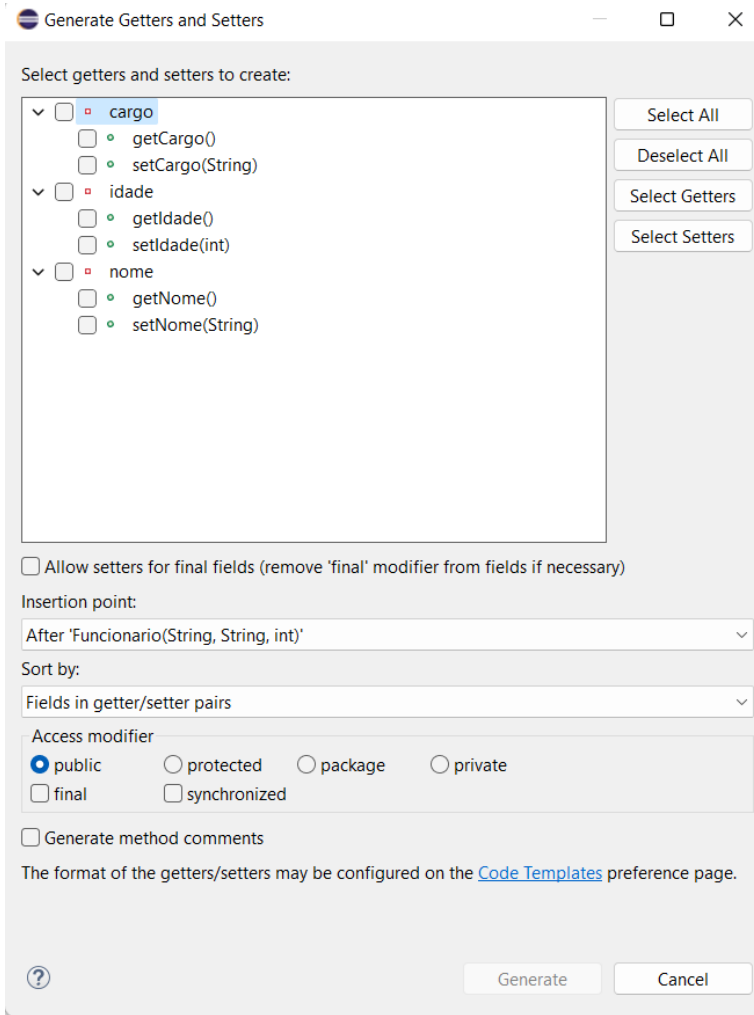
Undo Typing	Ctrl+Z
Revert File	
Save	Ctrl+S
Open Declaration	F3
Open Type Hierarchy	F4
Open Call Hierarchy	Ctrl+Alt+H
Show in Breadcrumb	Alt+Shift+B
Quick Outline	Ctrl+O
Quick Type Hierarchy	Ctrl+T
Open With	>
Show In	Alt+Shift+W >
Cut	Ctrl+X
Copy	Ctrl+C
Copy Qualified Name	
Paste	Ctrl+V
Raw Paste	
Quick Fix	Ctrl+1
Source	Alt+Shift+S >
Refactor	Alt+Shift+T >

Toggle Comment	Ctrl+7
Remove Block Comment	Ctrl+Shift+\
Generate Element Comment	Alt+Shift+J
Correct Indentation	Ctrl+I
Format	Ctrl+Shift+F
Format Element	
Add Import	Ctrl+Shift+M
Organize Imports	Ctrl+Shift+O
Sort Members...	
Clean Up...	
Override/Implement Methods...	
Generate Getters and Setters...	
Generate Delegate Methods...	
Generate hashCode() and equals()...	
Generate toString()...	
Generate Constructor using Fields...	
Generate Constructors from Superclass...	
Externalize Strings...	

Generate Getter and Setter Methods for the Type's Fields

Utilize o Eclipse para
gerar esses métodos

GET E SET



```
public String getName() {  
    return nome;  
}  
  
public void setName(String nome) {  
    this.nome = nome;  
}  
  
public String getCargo() {  
    return cargo;  
}  
  
public void setCargo(String cargo) {  
    this.cargo = cargo;  
}  
  
public int getIdade() {  
    return idade;  
}  
  
public void setIdade(int idade) {  
    this.idade = idade;  
}
```

GET E O SET

```
public Funcionario() {  
    }  
public Funcionario(String nome, String cargo, int idade) {  
    this.nome = nome.toUpperCase();  
    this.cargo = cargo;  
    this.idade = idade;  
}  
public String getNome() {  
    return nome;  
}
```

1

2

CONSTRUTOR

Crie o objeto ou
inicializa

```
public Funcionario() {  
  
}  
public Funcionario(String nome,String cargo,int idade) {  
    this.nome = nome.toUpperCase();  
    this.cargo = cargo;  
    this.idade = idade;  
}  
public Funcionario(String nome,String cargo) {  
    this.nome = nome.toUpperCase();  
    this.cargo = cargo;  
  
}  
public Funcionario(int idade) {  
  
    this.idade = idade;  
}
```

sobrecarga

```
1 package br.com.contoso.bo;
2
3 public class FuncionarioBO {
4     public String validarCpf() {
5         return "CPF válido";
6     }
7 }
8
```

Método worker (Business Object)

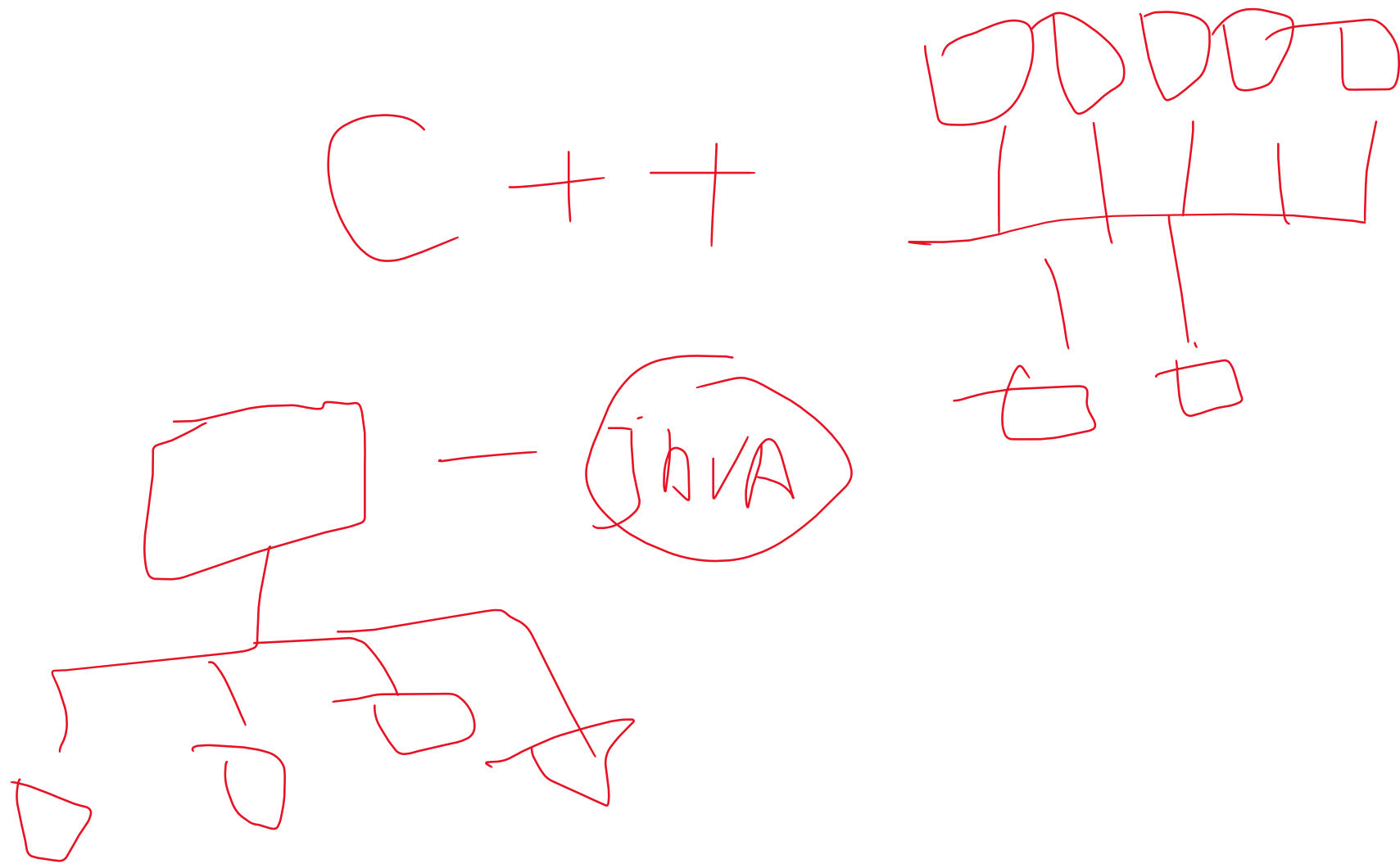
O CONCEITO DE MÉTODO WORKER FOI CITADO NO LIVRO DOMINANDO JAVA DE OZIEL MOREIRA NETO

```
public String validarRegistro() {  
    return "CPF válido";  
}  
public String validarRegistro(int codigo) {  
    return "CPF válido";  
}
```

sobrecarga

```
1 package br.com.contoso.dto;|
2 public class FuncionarioHorista extends Funcionario {
3     private int qtdeHoras;
4
5     public FuncionarioHorista(int qtdeHoras) {
6         super();
7         this.qtdeHoras = qtdeHoras;
8     }
9     public int getQtdeHoras() {
10         return qtdeHoras;
11     }
12     public void setQtdeHoras(int qtdeHoras) {
13         this.qtdeHoras = qtdeHoras;
14     }
15 }
```

Herança simples



Herança múltipla

Dê um comando para a sua mente separar apenas o que é realmente importante.

abstração

```
1 package br.com.contoso.dto;  
2  
3 public abstract class Funcionario {  
4  
5     private String nome;  
6     private String cargo;
```

Classe Abstrata

```
public abstract void verificaEntradaFunc();|
```

Método Abstrato

FuncionarioHorista

- Add unimplemented methods
- 📄 Create new JUnit test case for 'FuncionarioHorista.java'
- Make type 'FuncionarioHorista' abstract
- 🔗 Rename in file (Ctrl+2, R)
- 🔗 Rename in workspace

CONTRATO

```
@Override  
public void verificaEntradaFunc() {  
    // TODO Auto-generated method stub  
  
}
```

@OVERRIDE

Polimorfismo verdadeiro!

```
1 package br.com.contoso.dao;
2
3 public interface IFuncionarioCrud {
4     public void cadastrar();
5     public void pesquisar();
6     public void alterar();
7     public void excluir();
8 }
9
```

Interfaces

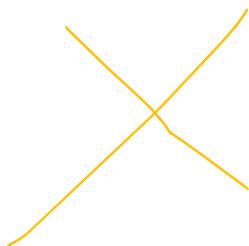
```

1 package br.com.contoso.dao;
2
3 public class FuncionarioDAO implements IFuncionarioCrud {
4
5     @Override
6     public void cadastrar() {
7         // TODO Auto-generated method stub
8     }
9
10    @Override
11    public void pesquisar() {
12        // TODO Auto-generated method stub
13    }
14
15    @Override
16    public void alterar() {
17        // TODO Auto-generated method stub
18    }
19
20    @Override
21    public void excluir() {
22        // TODO Auto-generated method stub
23    }
24 }
25
26 }
27

```

Polimorfismo


```
public Funcionario(String nome,String cargo,int idade) {  
    this.nome = nome.toUpperCase();  
    this.cargo = cargo;  
    this.idade = idade;  
}  
public Funcionario(String nome,String cargo) {  
    this.nome = nome.toUpperCase();  
    this.cargo = cargo;  
}  
public Funcionario(int idade) {  
    this.idade = idade;  
}
```



```
@Override  
public void cadastrar() {  
    // TODO Auto-generated method stub  
}  
  
@Override  
public void pesquisar() {  
    // TODO Auto-generated method stub  
}
```

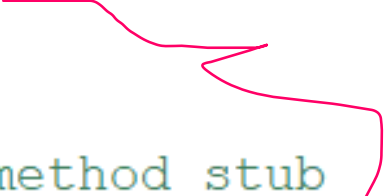
Sobrecarga x Sobrescrita

```
1 package br.com.contoso.dao;
2
3 public interface IFuncionarioArquivo {
4
5     public void abrirArquivo();
6     public void fecharArquivo();
7     public void pesquisarArquivo();
8
9
10 }
11
```

```
package br.com.contoso.dao;

public class FuncionarioDAO implements IFuncionarioCrud ,IFuncionarioArquivo{

    @Override
    public void cadastrar() {
        // TODO Auto-generated method stub
    }
}
```



Interface Múltipla

```
12 public class TelaAposentadoria extends JFrame implements ActionListener{ //herança
13     //Definindo um componente swing título
14     JLabel titulo;
15     //Definindo componentes swing -botões
16     JButton btnGravar,btnLer,btnLimpar;
17     //Definindo um caixa de texto para strings longas.
18     JTextArea txtArea;
```

Interface no Swing

```

11 //Herança a classe JFrame
12 public class TelaAposentadoria extends JFrame implements ActionListener { //herança
13     //Definindo um componente swing título
14     JLabel titulo;
15     //Definindo componentes swing -botões
16     JButton btnGravar, btnLer, btnLimpar;
17     //Definindo um caixa de texto para strings longas.
18     JTextArea txtArea;
19
20     //construtor
21     public TelaAposentadoria() {
22         //referenciando o pai (superclasse)

```

Implementando Interface no Swing

```
@Override
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    if(e.getSource() == btnGravar) {
        JOptionPane.showMessageDialog(null, "Gravando Arquivo...");
    } else if(e.getSource() == btnLer) {
        JOptionPane.showMessageDialog(null, "Lendo Arquivo...");
    } else {
        JOptionPane.showMessageDialog(null, "Limpendo entrada de dados...");
        txtArea.setText("");
    }
}
```

Implementando Interface no Swing

```
11 //Herando a classe JFrame
12 public class TelaAposentadoria extends JFrame implements ActionListener { //herança
13     //Definindo um componente swing título
14     JLabel titulo;
15     //Definindo componentes swing -botões
16     JButton btnGravar, btnLer, btnLimpar;
17     //Definindo um caixa de texto para strings longas.
18     JTextArea txtArea;
19
20     //construtor
21     public TelaAposentadoria() {
22         //referenciando o pai (superclasse)
```

Implementando Interface no Swing

R

Gere uma interface Java com um método abstrato e um método default no Java 8



Aqui está um exemplo de como você pode criar uma interface Java com um método abstrato e um método default no Java 8:



csharp

Copy code

```
interface MinhaInterface {  
    void metodoAbstrato();  
    default void metodoPadrao() {  
        System.out.println("Este é um método padrão");  
    }  
}
```

O método abstrato é um método que não tem uma implementação e é obrigatório ser implementado por qualquer classe que implemente a interface. Já o método default é um método que já tem uma implementação e pode ser sobrescrito ou não pela classe que o implementa.

Inteface Java 8

DEFAULT VOID

Brigar com C# e outras linguagens funcionais - Lambda.

```
1 package br.com.contoso.bo;
2
3 interface MinhaInterface {
4     void metodoAbstrato();
5     default void metodoPadrao() {
6         System.out.println("Este é um método padrão");
7     }
8 }
9
```

Interface e método default (Java 8)


```
1 package br.com.contoso.bo;
2
3 public class FuncionarioBO implements MinhaInterface{
4     public String validarRegistro() {
5         return "CPF válido";
6     }
7     public String validarRegistro(int codigo) {
8         return "CPF válido";
9     }
10    @Override
11    public void metodoAbstrato() {
12        // TODO Auto-generated method stub
13
14    }
15 }
```

E o Default?

```
public int somar(int x, int y ){  
    return x+y;  
}
```

```
soma = (x, y) -> x + y;
```

Programação Funcional

Programação funcional é um paradigma de programação que se baseia em conceitos matemáticos de funções, em que o resultado de uma função é determinado apenas pelos seus argumentos e não por estados ou dados externos. Nesta abordagem, as funções são tratadas como valores imutáveis e as operações são compostas de funções simples para criar funções mais complexas. Esta abordagem é conhecida por sua clareza, expressividade e facilidade de manutenção do código. Linguagens de programação como Haskell, Lisp, e F# são exemplos de linguagens de programação funcional.

Bater um papo com o GPT



Aqui está uma função lambda no Java para tratar um evento ActionListener no framework de interface gráfica de usuário Swing:

```
java Copy code

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;

public class Main {
    public static void main(String[] args) {
        JButton button = new JButton("Clique aqui");
        button.addActionListener((ActionEvent e) -> {
            System.out.println("Botão clicado!");
        });
    }
}
```

Bater um papo com o GPT

```
btnLimpar.addActionListener((ActionEvent e) -> {  
    txtArea.setText("");  
});
```

Botão no Swing com Lambda

Neste exemplo, criamos um botão JButton e adicionamos um ouvinte de ação a ele. A função lambda é usada como o ouvinte de ação, que imprime "Botão clicado!" na saída quando o botão é clicado.

Bater um papo com o GPT

Uso de Lambda

- Swing
- Cálculos (BO - Business Object)



O filé do uso de Lambda é quando você consegue Aplicar no uso de:

Vetor, Array, Collections

Interface + Lambda + Swing + BO + Collections