

Sistema

FIRJAN



INFORMA, FORMA, TRANSFORMA.

# Programação Desktop

*Fabício Curvello Gomes*

Sistema

FIRJAN



INFORMA, FORMA, TRANSFORMA.



# Interfaces

2

INFORMA. FORMA. TRANSFORMA.

# Interfaces

Pelo fato da Orientação a Objetos não permitir herança múltipla, faz-se necessária a utilização de interfaces.

A interface é uma Classe que permite estabelecer um “contrato” entre Classes, porém não permite a implementação de um método, contendo apenas a especificação deste.

OBS: A interface define os métodos que as Classes que a implementam são obrigadas a conter.

3

INFORMA. FORMA. TRANSFORMA.



## Exemplo *15\_Interface*

```

classDiagram
    class Usuario {
        - login : String
        - senha : String
        - tipo : int
        + mostrar() : void
    }
    class Aluno {
        - matriculaAluno : int
        - nome : String
        - turno : String
        - turma : String
        + mostrar() : void
        + validarLogin() : boolean
    }
    class Professor {
        - matriculaProfessor : int
        - nome : String
        - especialidade : String
        + mostrar() : void
        + validarLogin() : boolean
    }
    Usuario <|-- Aluno
    Usuario <|-- Professor
    
```

Na Interface ***IUsuario*** existe o método ***validarLogin***, o que força uma implementação obrigatória deste método em todas as SubClasses ligadas à interface.


4





INFORMA, FORMA, TRANSFORMA.

## Exemplo *15\_Interface* (Cont.)

1. Copiar o projeto *14\_ClassesAbstratas* e renomear para *15\_Interface*.
2. Na Classe Teste, apagar a linha de declaração de objeto *user* (que está com erro)
3. No pacote *model*, criar a **Interface** *IUsuario*.
4. Na interface, implementar o método `validarLogin`:  

```
boolean validarLogin(String login, String senha);
```


5




INFORMA, FORMA, TRANSFORMA.

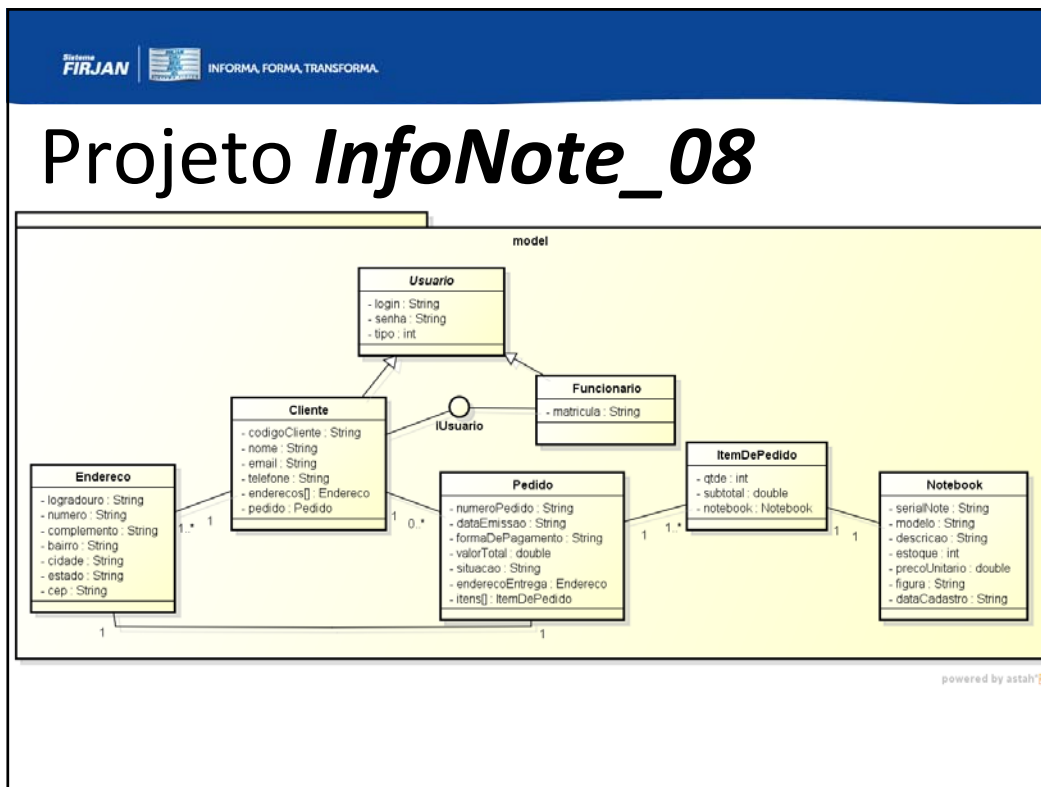
## Exemplo *15\_Interface* (Cont.)

5. Na Classe *Aluno*, fazer a referência à Interface:  

```
public class Aluno extends Usuario implements IUsuario{
```
6. Ainda na Classe *Aluno*, implementar o polimorfismo:  

```
@Override
public boolean validarLogin (String login, String senha){
    if(getLogin().equals(login) && getSenha().equals(senha)){
        return true;
    }
    return false;
}
```
7. Aplicar os itens 5 e 6 também na Classe *Professor*.




6



**Projeto *InfoNote\_08* (Cont.)**

- Copiar o projeto *InfoNote\_07*, e colar renomeando-o.
- Criar a Interface *IUsuario*, que fará a interface entre as SubClasses *Cliente* e *Funcionário*, para validação de login e senha.
- Nas Classes *Cliente* e *Funcionário*, fazer as referências à interface e implementar o polimorfismo.


8



  **FIRJAN** | **INFORMA. FORMA. TRANSFORMA.**

## Projeto *InfoNote\_08* (Cont.)


Na Classe *InfoNote*, que está no pacote *controller*, alterar o método *efetuarLogin()*, de forma que ele deixe de trabalhar com login fixo admin e senha fixa 1234, e passe a interagir com o método *validarLogin()*:

```
public void efetuarLogin() {  
    String login, senha;  
    login = Teclado.lerTexto("Digite o login: ");  
    senha = Teclado.lerTexto("Digite a senha: ");  
    if (cliente != null) {  
        logado = cliente.validarLogin(login, senha);  
        if (logado) {  
            System.out.println("Login efetuado com sucesso!");  
        } else {  
            System.out.println("Usuário ou senha inválido.");  
        }  
    }  
}
```


 9

  **FIRJAN** | **INFORMA. FORMA. TRANSFORMA.**


## Dúvidas?




10




# Bibliografia



Java Como Programar 8ª Edição  
Paul Deitel e Harvey Deitel  
Ed. Pearson



Java 7 Ensino Didático  
Sérgio Furgeri  
Ed. Érica



Fundamentos de Computação e Orientação a Objetos Usando Java  
Francisco A. C. Pinheiro  
Ed. LTC

11