Inatel
Instituto Nacional de Telecomunicações

Programação Orientada a Objetos

Cap.8 - Interfaces

Prof. Me. Renzo P. Mesquita

Objetivos

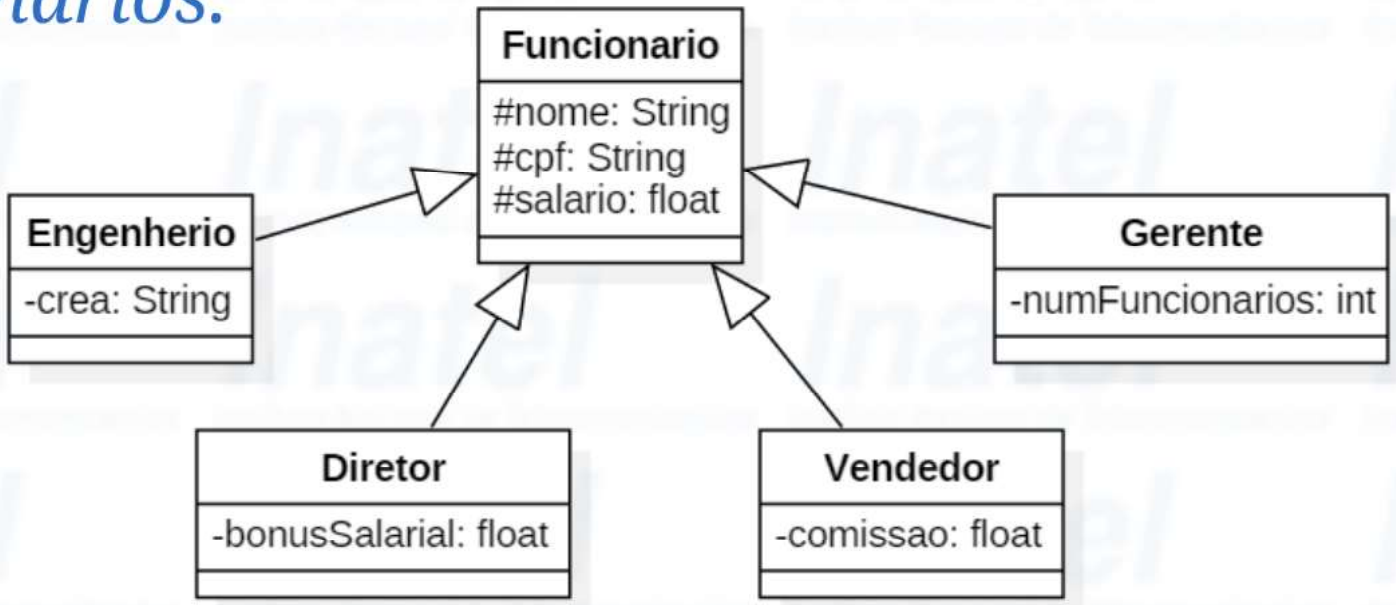
- Compreender o que são Interfaces e a diferença entre herança e implementação;
- Escrever Interfaces em Java;
- Utilizá-las como um poderoso recurso para diminuir o acoplamento entre as Classes;

Capítulo 8
Interfaces

8.1. Introdução;
8.2. Interfaces;
8.3. Interfaces no Diagrama de Classes;
8.4. Classes Abstratas vs. Interfaces;
8.5. Mais exemplos de cases com Interfaces;

8.1. Introdução

Vamos voltar ao nosso exemplo de Herança de Funcionários:



Agora suponha que de todos os Funcionários, precisamos criar um método autentica() para o Gerente e para o Diretor para que eles possam realizar um acesso restrito no sistema.

Baseado nos conceitos que vimos até agora, qual seria a forma mais genérica e possivelmente mais inteligente para resolver este problema?

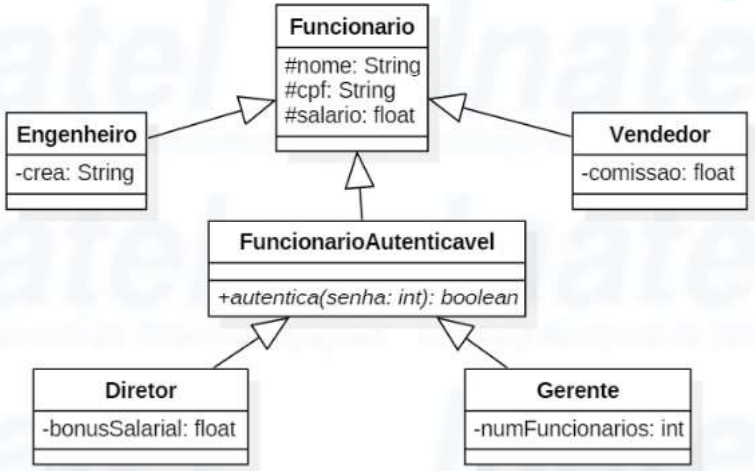
8.1. Introdução

Para o problema anterior, poderíamos apresentar as seguintes soluções:

1) Criar um método autentica() em Funcionário;

Será que ficaria legal? Lembre-se que foi definido que apenas o Gerente e o Diretor poderiam enxergar e sobrescrever este método :(.

2) Criar uma Classe chamada FuncionarioAutenticavel que estende de Funcionario e seja mãe de Gerente e Diretor;



Jóia! Mas e se em determinado momento também quiséssemos que os nossos Clientes também fossem autenticáveis? Clientes são Funcionários? :(

Mas e então? Como resolver esse problema de forma genérica e inteligente? As Interfaces poderão nos dar uma forcinha! ;)

8.2. Interface

Uma Interface em Java permite criar um "CONTRATO" que define tudo que uma Classe deva fazer, independente do seu tipo, caso queira realizar uma ou mais ações específicas.

No nosso exemplo anterior, poderíamos criar um contrato que definisse o seguinte para quem quisesse ser autenticável:

Contrato Autenticavel:

quem quiser ser Autenticavel deverá:
1.realizar uma autenticação com uma senha;
2.devolver um booleano;

E como poderíamos criar este contrato em Java?

```
public interface Autenticavel {

    boolean autentica(int senha);

}
```

8.2. Interface

A Interface é um contrato onde quem assina se responsabiliza por implementar seus métodos (cumprir o contrato).

Uma Interface pode definir uma série de métodos, mas NUNCA conter a implementação deles. Ela só expõe O QUE um objeto deve fazer, e não COMO ele faz.

No nosso caso anterior, como podemos fazer para que uma Classe IMPLEMENTE uma Interface? Basta utilizarmos da palavra chave implements.

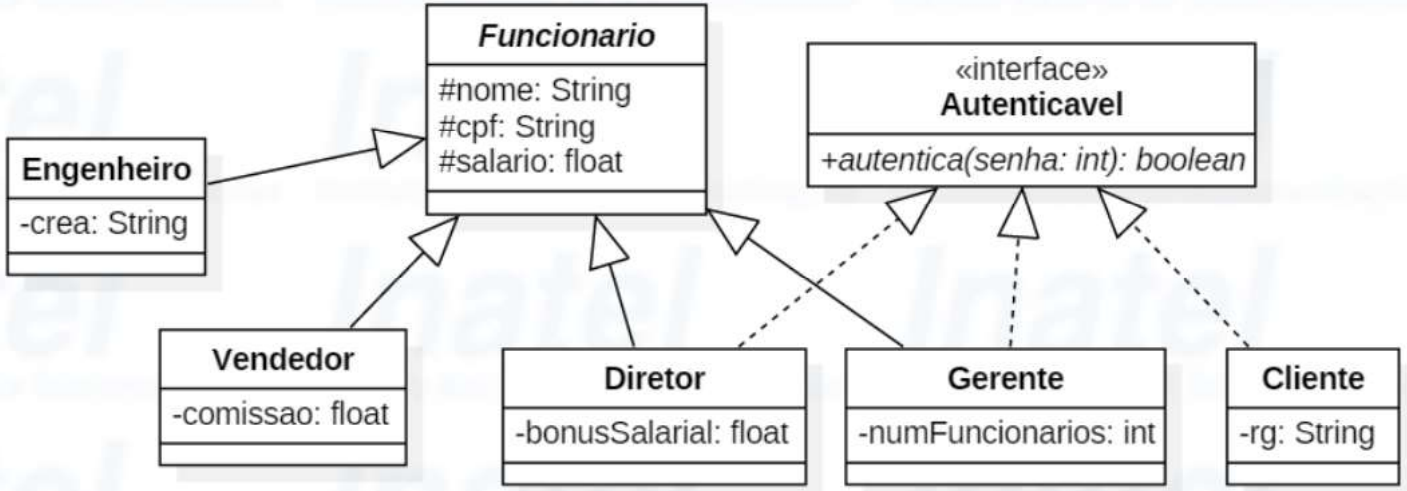
Exemplo:

```
public class Gerente extends Funcionario implements Autenticavel{

    private int senhaEspecial;
    private int numFuncionarios;

    @Override
    public boolean autentica(int senha) {
        // Implementação do Método autentica para o Gerente
        return true;
    }
    //... resto da Classe
}
```

8.3. Interfaces no Diagrama de Classes

Métodos de uma Interface são públicos e abstratos por padrão!

No Diagrama de Classes, podemos representar uma Interface da seguinte maneira:



A partir de agora, no nosso exemplo, podemos transformar qualquer Classe em um autenticável, basta que elas implementem a Classe Autenticavel.
Sem contar que também ganhamos mais Polimorfismo! Do mesmo jeito da herança, podemos referenciar um Diretor, Gerente ou Cliente da seguinte forma:

```
Autenticavel a1 = new Gerente();
Autenticavel a2 = new Diretor();
Autenticavel a3 = new Cliente();
```

8.4. Classes Abstratas vs. Interfaces

Apesar das Classes Abstratas e Interfaces possuírem características muito similares, existem algumas diferenças entre elas:

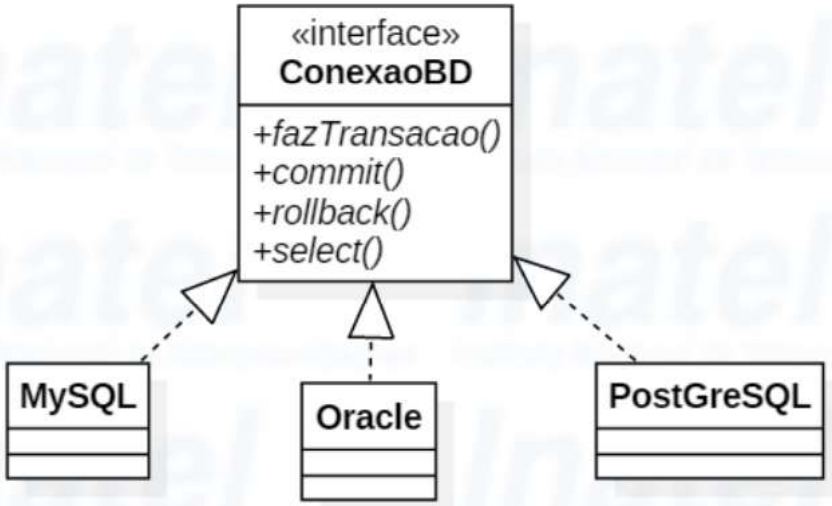
Classes Abstratas **vs.** *Interfaces*

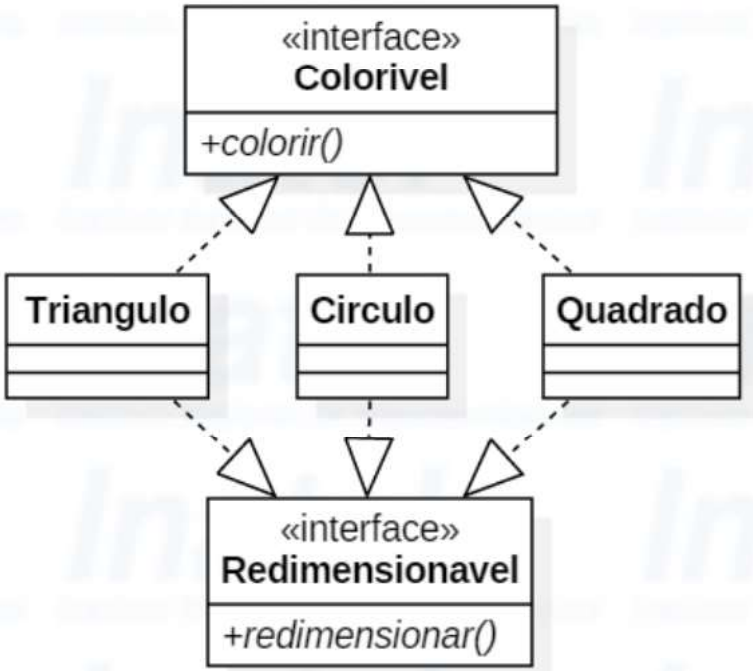
- Pode possuir variáveis comuns e constantes;
- Pode ou não ter implementações de código;
- Pode possuir métodos concretos e abstratos;
- Utiliza apenas de herança simples (por meio da palavra extends);
- Pode possuir Construtores;

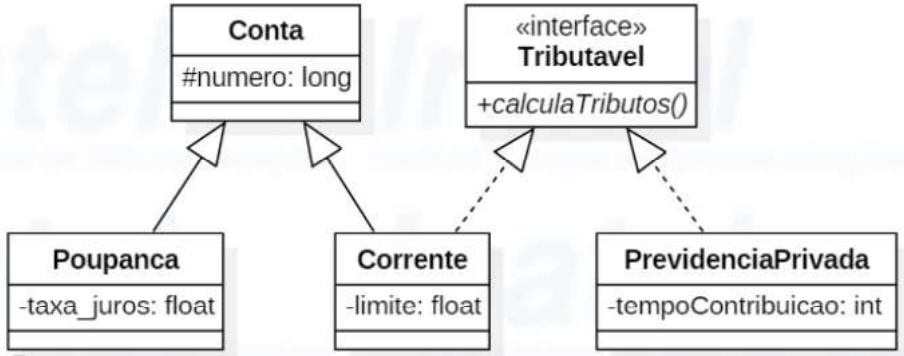
- Suporta apenas constantes;
- Não pode conter qualquer tipo de implementação de código (só fornece cabeçalhos);
- Todos os métodos de uma Interface são por padrão abstratos;
- Permite herança múltipla (por meio da palavra implements);
- Não possui Construtores;

8.5. Mais exemplos de cases com Interfaces

Abaixo alguns outros exemplos de Interfaces e Classes que poderiam utilizar delas:







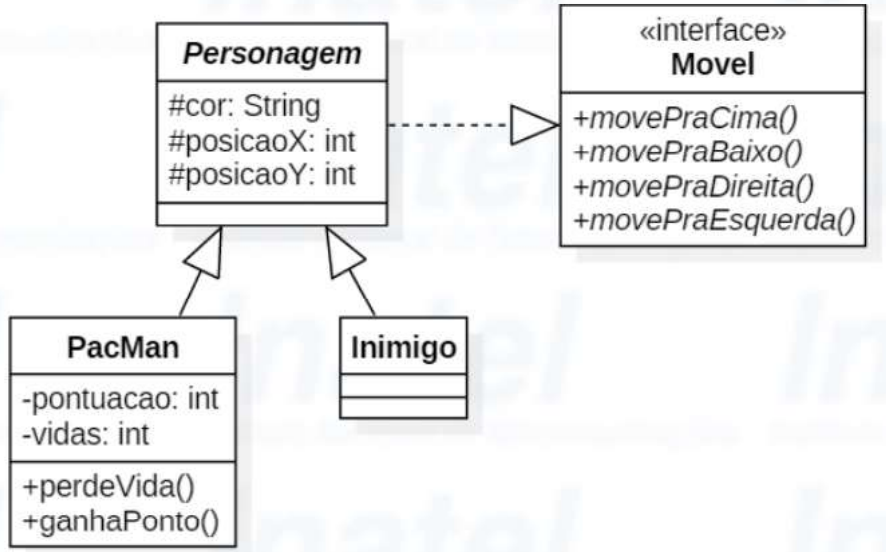
entre infinitas outras..

8.5. Mais exemplos de cases com Interfaces

Exercício 1 - PacManInterface

A fim de programar as Mecânicas do jogo PacMan, crie Classes em Java que atendam às seguintes regras e especificações:

1. Os personagens deverão caminhar em coordenadas que estejam dentro de uma faixa 100x100;
2. Um personagem deve se mover por vez;
3. Os Inimigos deverão começar o jogo nas extremidades, e o PacMan no centro;
4. Utilizando dos métodos da Interface, os Personagens deverão se mover aleatoriamente pelas posições (Deve-se tratar posições não permitidas);
5. O Jogo acaba quando o PacMan se encontra com um inimigo e ele perde todas as suas vidas;
6. A cada posição que o PacMan anda sem encontrar com os seus inimigos ele aumenta sua pontuação;
7. Depois que o PacMan perde todas as vidas, deve-se mostrar a mensagem de "Game Over" e a pontuação final;



FIM
DO
CAPÍTULO 8

Próximo Capítulo

Coleções