Programação Orientada a Objetos

Aula 08 – Classes Abstratas e Interface

Prof. Luiz Mário Lustosa Pascoal

Classes Abstratas

- Usamos a palavra chave abstract para impedir a instanciação de uma classe.
- Classes abstratas normalmente possuem um ou mais métodos abstratos.
- Um método abstrato não possui corpo, apenas define a assinatura (protótipo) do método
- Classes abstratas são úteis quando uma classe define métodos a serem implementados apenas pelas suas subclasses

Classes Abstratas

- **Regras sobre as classes abstratas e seus métodos:**
  - Toda classe que possui métodos abstratos é automaticamente abstrata e deve ser declarada como tal.
  - Uma classe pode ser declarada abstrata mesmo se ela não possuir métodos abstratos.
  - Uma classe abstrata não pode ser instanciada.
  - Uma subclasse de uma classe abstrata pode ser instanciada se:
    - Sobrescreve cada um dos métodos abstratos de sua superclasse
    - Fornece uma implementação para cada um deles

Classes Abstratas

- Se uma subclasse de uma classe abstrata não implementa todos os métodos abstratos que ela herda, então a subclasse também é abstrata
- Para declarar um método abstrato coloque a palavra-chave **abstract** no início do protótipo do método.
- Para declarar uma classe abstrata coloque a palavra-chave **abstract** no início da definição da classe

Classes Abstratas - Exemplo

```
public abstract class Conta {
   protected double saldo;

   public void retira(double valor) {
      this.saldo -= valor;
   }

   public void deposita(double valor) {
      this.saldo += valor;
   }

   public double getSaldo() {
      return this.saldo;
   }

   public abstract void atualiza();
}
```

Porém...

- **Problema:**
  - Suponha que você define as classes abstratas **Forma** e **Desenhavel** e a classe **Circulo** (que estende **Forma**)
  - Como criar a classe **CirculoDesenhavel** que é ao mesmo tempo **Circulo** e **Desenhavel**?

```
public class CirculoDesenhavel
        extends Circulo, Desenhavel { ... }
```

  - **Mas nem tudo são flores... Pois o código acima está ERRADO!**
    - Porque o JAVA, diferentemente do C++, não suporta herança múltipla!

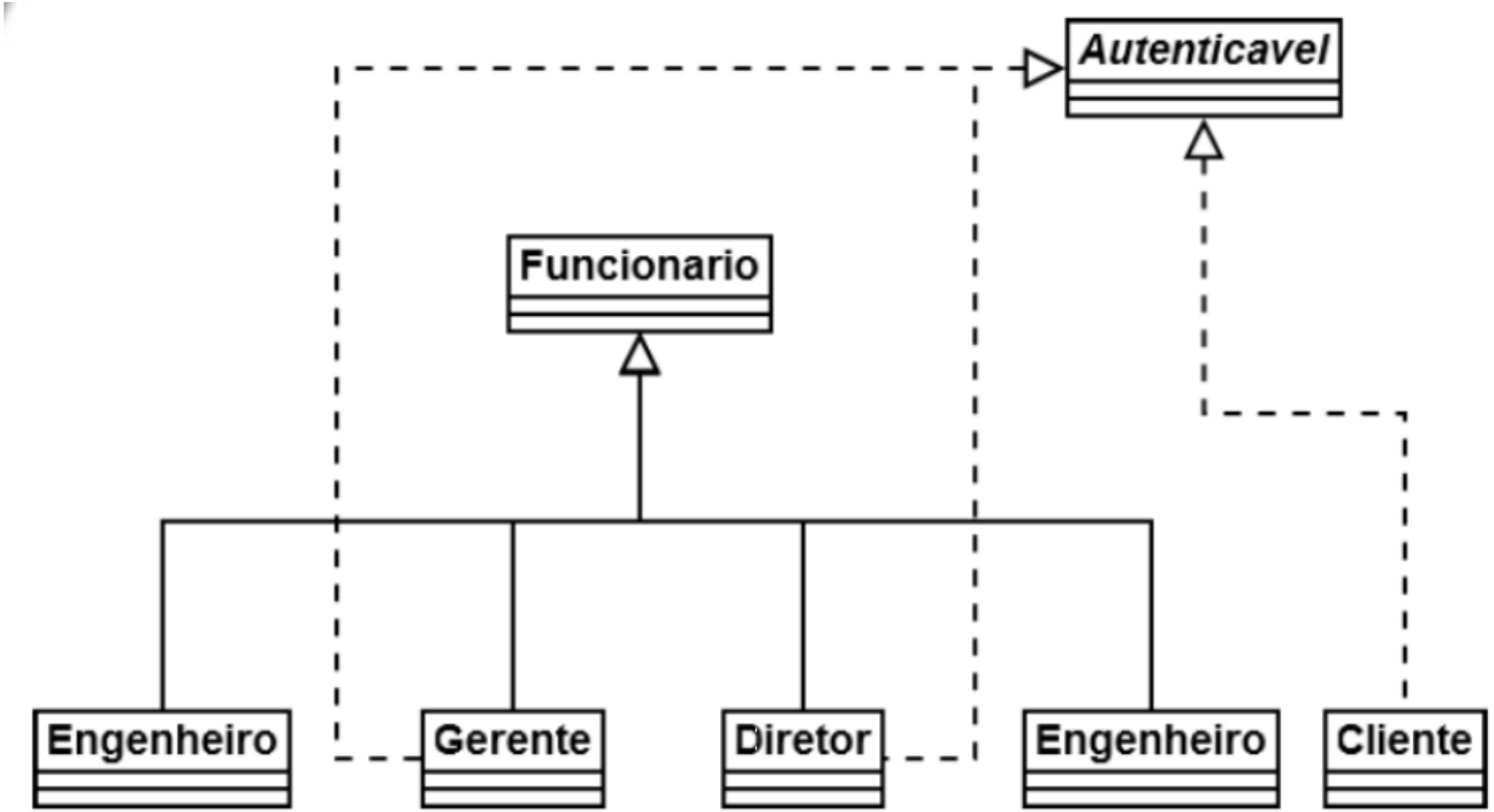
Interfaces

- **Solução:**
  - Para resolver esse tipo de problema, o JAVA define o conceito de ***Interfaces.***
    - Uma interface é criada de forma semelhante a uma classe abstrata, exceto pelo fato de utilizar a palavra chave **interface** no lugar de **abstract class** na definição da classe.
  - **Interface é um contrato onde quem assina se responsabiliza por implementar os métodos definidos na interface (cumprir o contrato).**
  - Ela só **expõe o que o objeto deve fazer, e não como ele faz, nem o que ele tem**. **Como** ele **faz** vai ser definido em uma **implementação** dessa interface.

Interfaces

- **Quais as diferenças entre interface e classe abstrata?**
  - Declaração de métodos
    - Em uma classe abstrata podem ser definidos métodos abstratos e não-abstratos
    - Todos os métodos de uma interface são implicitamente abstratos
  - Declaração de variáveis
    - Em uma classe abstrata podem ser definidas variáveis de instância, de classe e constantes.
    - **Todos os atributos de uma interface são sempre public static final** (constantes), mesmo que essas palavras reservadas não sejam usadas.

Interfaces - UML



Interfaces

```
public interface Autenticavel {
   public boolean autentica(int senha);
}
```

```
class Gerente extends Funcionario
                        implements Autenticavel {
   private int senha;

   // outros atributos e métodos
   public boolean autentica(int senha) {
      if (this.senha != senha) {
         return false;
      }
      // pode fazer outras possiveis verificacoes,
      // como saber se esse departamento do
      // gerente tem acesso ao Sistema
      return true;
   }
}
```

Herança entre Interfaces

- Diferentemente das classes, uma interface pode herdar de mais de uma interface.
- É como um contrato que depende de que outros contratos sejam fechados antes deste valer.
- Você não herda métodos e atributos, e sim responsabilidades.

Por que usar Interfaces?

- Para deixar o código mais flexível, e possibilitar a mudança de implementação sem maiores traumas.
- Elas também trazem vantagens em não acoplar as classes.
  - Uma vez que herança através de classes traz muito acoplamento, muitos autores clássicos dizem que em muitos casos herança quebra o encapsulamento.

Exercício para praticar!

- Crie a seguinte hierarquia de classes:
  - Uma interface para representar qualquer forma geométrica, definindo métodos para cálculo do perímetro e cálculo da área da forma;
  - Uma classe abstrata para representar quadriláteros. Seu construtor deve receber os tamanhos dos 4 lados e o método de cálculo do perímetro já pode ser implementado;
  - Classes para representar retângulos e quadrados. A primeira deve receber o tamanho da base e da altura no construtor, enquanto a segunda deve receber apenas o tamanho do lado;
  - Uma classe para representar um círculo. Seu construtor deve receber o tamanho do raio.

No programa principal, pergunte ao usuário quantas formas ele deseja criar. Em seguida, para cada forma, pergunte se deseja criar um quadrado, um retângulos ou um círculo, solicitando os dados necessários para criar a forma. Todas as formas criadas devem ser armazenadas em um vetor. Finalmente, imprima: (a) os dados (lados ou raio); (b) os perímetros; e (c) as áreas de todas as formas. Para (b) e (c), tire vantagem do polimorfismo.