FATEC – Votorantim
Profº Rodrigo Diver

Aula 6 – Classes Abstratas e Interfaces

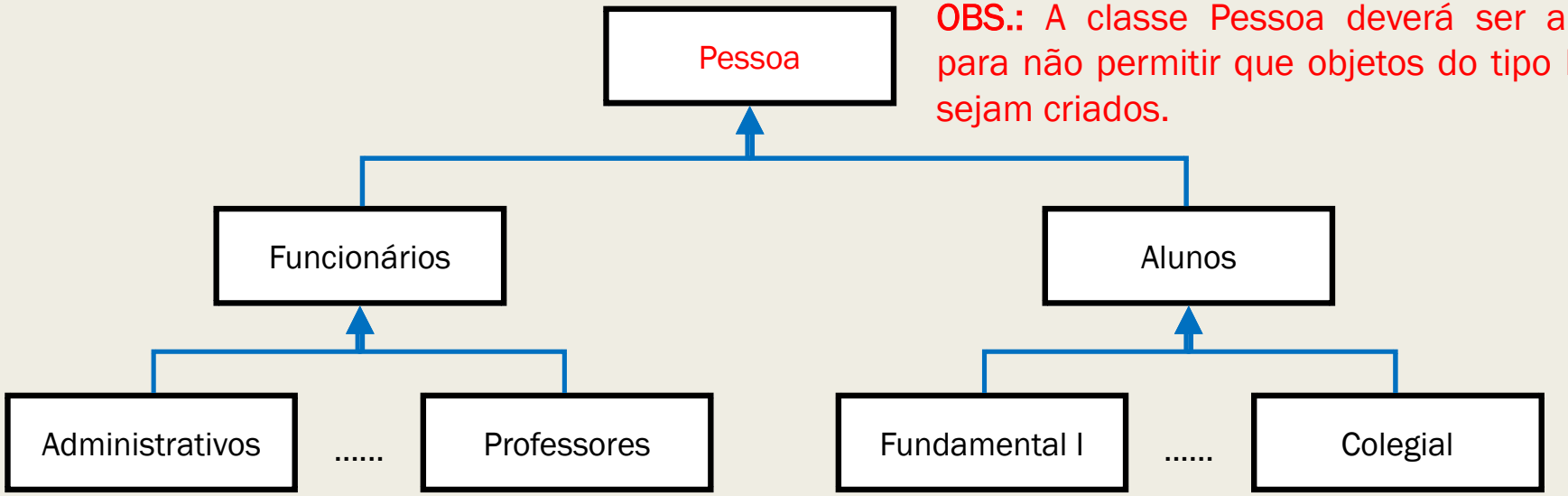
- **Classes Abstratas.**
  - *O que é uma classe Abstrata?*
  - *Utilização de Herança com classes Abstratas.*

- **Interfaces.**
  - *O que é uma Interface?*
  - *Utilização de Interfaces.*

Herança entre Classes

- Devemos construir um sistema de gestão escola, que implemente diversos tipos de objetos Funcionários e diversos tipos de objetos Alunos. Com todos os objetos sendo herdados de uma classe mais geral Pessoas. Porém, foi definido que a classe Pessoas não pode ser instanciada.

- Como podemos utilizar os conceitos de Herança para resolver este problema?



OBS.: A classe Pessoa deverá ser abstrata para não permitir que objetos do tipo Pessoa sejam criados.

Métodos e Classes Abstratas

- Como vimos, subclasses podem redefinir (@Override) um método definido em sua superclasse. Assim como podem usar o método da forma como foi herdado da superclasse.

- Para indicar que um método de uma classe deve ser necessariamente redefinido em cada uma de suas subclasses, este método deve ser declarado como abstract.

- O que é um método abstrato?
  - É um método que não foi implementado.

- Uma classe que contém um ou mais métodos abstratos deve ser declarada explicitamente como abstrata. Essa classe, no entanto, pode ter métodos concretos (não-abstratos).
  - Porém não é permitido instanciar ou seja criar objetos de uma classe abstrata.

Métodos e Classes Abstratas

Classe Abstrata

```
public abstract class Pessoa{
    protected String nome,
    private String genero;
    private int idade;

    public abstract String getNome();

    public String getGenero(){
        return genero;
    }
    public int getIdade(){
        return idade;
    }
}
```

Método Abstrato

Método Concreto

Método Concreto

Por exemplo:

As classes do tipo **Funcionários** quando herdarem da **classe Pessoa**, deverão implementar o **método getNome** adicionando os prefixos **"Sr.", Sra.", "Profº.", "Profª.",** dependendo do gênero da pessoa e do tipo do funcionário.

As classes do tipo **Aluno** não deverão utilizar prefixos.

Atributos que precisam ser acessados pelas subclasses, devem ser declarados como ***protected***.

Métodos e Classes Abstratas

Classe Abstrata

```
public abstract class Funcionario extends Pessoa{
    protected float salarioBase;

    public abstract double calculaSalario();
    @Override
    public String getNome(){
        if (super.genero.equals("masculino")) return "Sr. " +
super.nome ;
        else      "Sra. " + super.nome ;
    }
    public void registrarPonto(){
        /* === Acessar o BD e registrar o horário == */
    }
}
```

Funcionario é uma subclasse de Pessoa

Método Abstrato

Método Concreto

O método abstrato getNome é implementado na subclasse Funcionario

Método Concreto

Métodos e Classes Abstratas

Uma classe final não poderá ser herdada.

Classe Final

```
public final class Administrativo extends Funcionario{

    @Override
    public double calculaSalario(){
        return super.salarioBase;
    }

}
```

Administrativo é uma subclasse de Funcionario, que é um subclasse de Pessoa

Método Concreto

O método abstrato calculaSalario é implementado na subclasse Administrativo

Métodos e Classes Abstratas

Classe Final

```
public final class Professor extends Funcionario{
    private float salarioHora, qtdHoras

    @Override
    public double calculaSalario(){
        return super.salarioBase + (salarioHora * qtdHoras);
    }
    @Override
    public String getNome(){
        if (super.gênero.equals("masculino"))
            return "Professor " + super.nome ;
        else
            return "Professora " + super.nome ;
    }}
```

Administrativo é uma subclasse de Funcionario, que é um subclasse de Pessoa

Método Concreto Implementado

O método abstrato calculaSalario é implementado na subclasse Administrativo

Método Concreto Redefinido

O método concreto getNome é redefinido (@Override) na subclasse Professor

Métodos e Classes Abstratas

Exercício

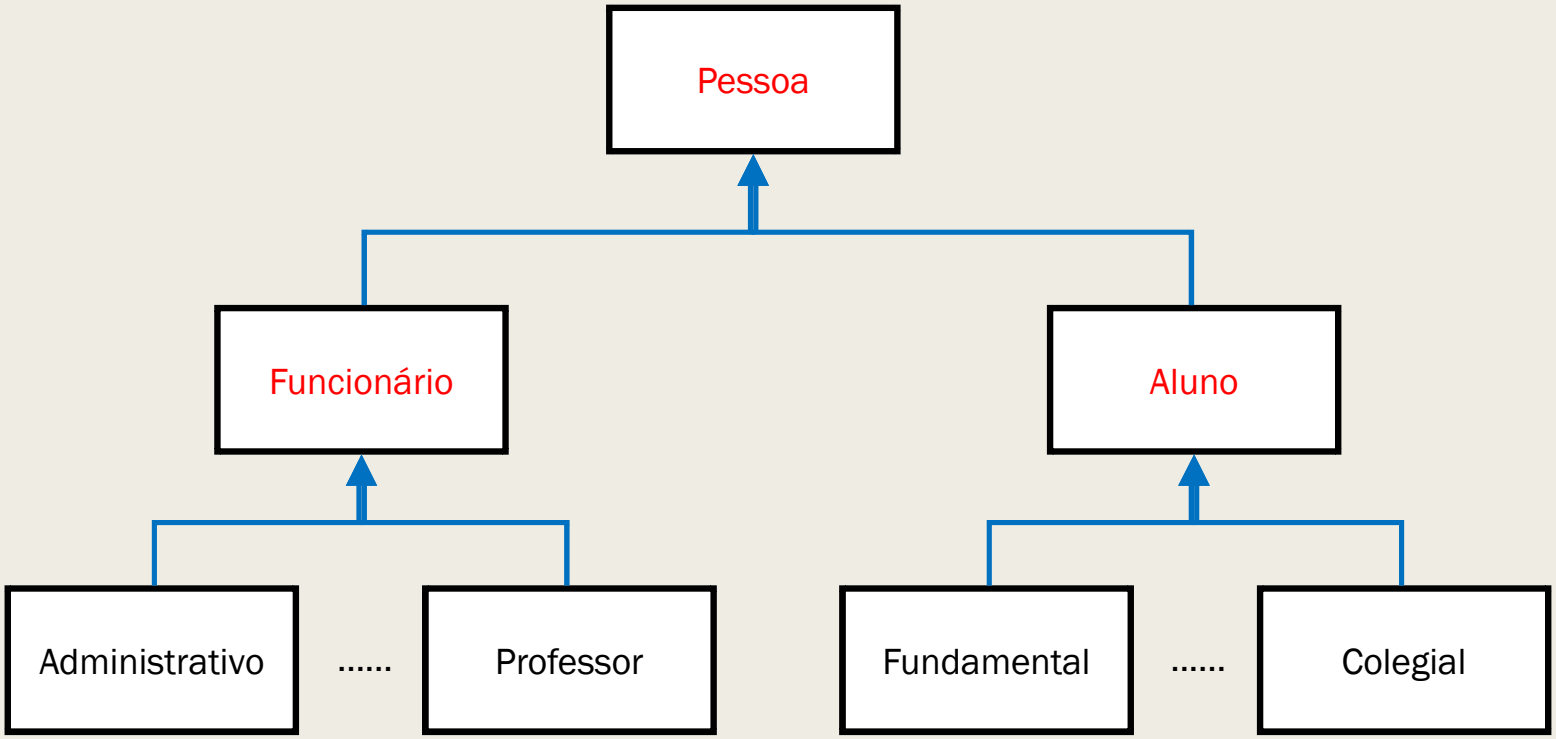
I. Implemente as classes abstratas **Pessoa, Funcionario e Aluno** e as respectivas classes do tipo final **Administrativo, Professor, Fundamental, Colegial.**

II. A Classe abstrata **Aluno** deverá possuir um atributo chamado **nota** do **tipo inteiro** e um método abstrato **String exiberNotas**, este método deverá ser implementado nas subclasses **Fundamental** e **Colegial**.

III. Na classe **Fundamental** o método **exibirNotas** deverá retornar valores de **A à D**, já a classe **Colegial** deverá retornar valores de **0 à 10** . Em ambas as classes o valor da nota será armazenada no **atributo nota** da superclasse **Aluno**.

IV. Implemente os códigos de exemplo desta aula e os conceitos da aula 5 para construir as classes, implementando os respectivos construtores e métodos de acesso **getters** e **setters**.

Métodos e Classes Abstratas



Interfaces

- Interfaces são classes abstratas que contêm apenas métodos abstratos. Portanto, em uma interface não existe implementação de nenhum método. A definição de uma interface é feita pela palavra-chave interface (em vez de class).

- Para utilizar uma interface, uma classe deve especificar que implementa (implements) a interface e a classe deve definir cada método da interface respeitando o número de argumentos e o tipo de retorno especificado na interface. Se a classe deixar de implementar algum método da interface, a classe se torna abstrata.

- Uma classe pode implementar mais de uma interface.

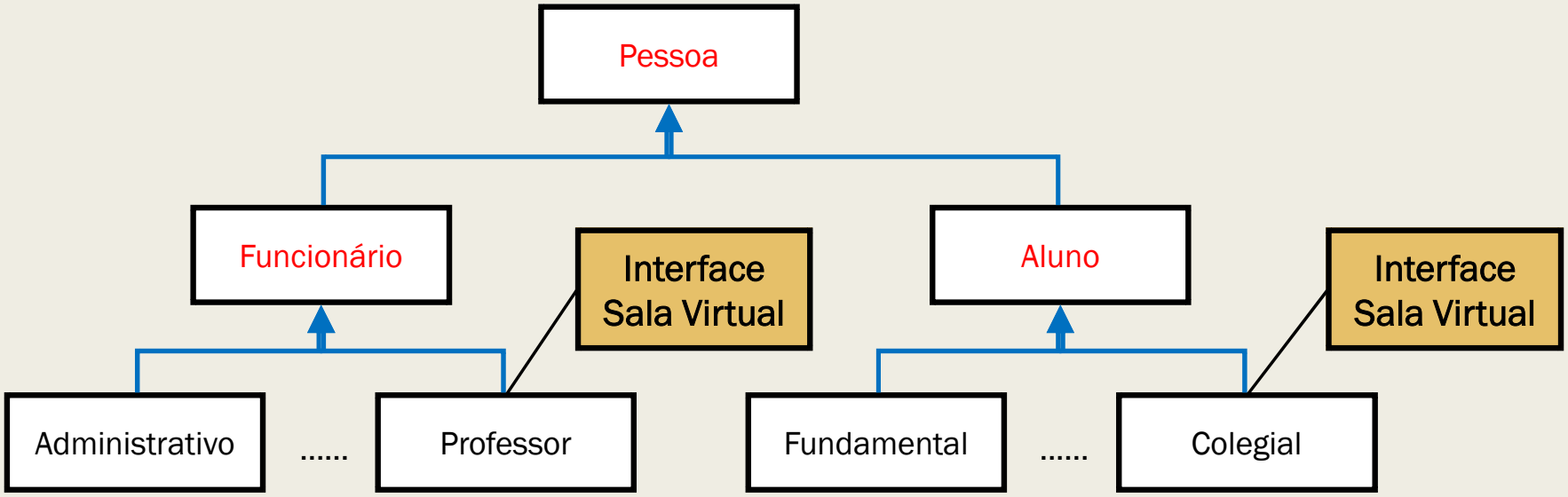
- Uma mesma interface pode ser implementada por várias classes.

- Uma interface pode ser usada como uma forma de apresentar aos usuários os métodos disponíveis, sem revelar os detalhes de implementação desses métodos.

Interfaces

Por exemplo: se precisarmos implementar de forma obrigatória, somente nas classes **Professor** e **Colégio** os métodos para permitir o acesso a sala de aula virtual.

Não poderíamos definir esses métodos nas classes abstratas **Funcionário** e **Aluno**, pois as demais subclasses **Administrativo** e **Fundamental** também teriam que implementar os métodos. A utilização de uma interface permite resolver esse problema.



Interfaces

Interface

```
public interface SalaVirtual{

    public String login();
    public String logout ();
}

public final class Professor extends Funcionario implements SalaVirtual{
    @Override
    public String login(){
        ..................  }
    @Override
    public String logout{
        ..................  }
}
```

Métodos Concretos Implementados

Professor é uma subclasse de Funcionario, e implementa a interface SalaVirtual

Métodos e Classes Abstratas

Exercício

I. Crie uma interface **SalaVirtual** com os métodos abstratos **login**() e **logout**().

II. Implemente a interface **SalaVirtual** nas classes **Professor** e **Colegial** do exercício anterior.

III. Implemente os métodos abstratos da interface **SalaVirtual** nas classes, retornando a mensagem "*Conectado na sala virtual !*" para o método **login()** e "*Desconectado da Sala Virtual !*" para o método **logout().**

IV. Crie uma classes do tipo enum com o nome de Genero e as seguintes constantes: "MASCULINO", "FEMININO". Altere o atributo gênero da classe Pessoa do tipo String para o tipo enum Genero.

V. **Responda**: *Porque é mais adequado utilizar um tipo enum no atributo gênero?*

VI. Instancie os objetos das classes criadas e faça o teste dos métodos das respectivas classes.

VII. **Responda**: *É possível criar objetos do tipo **Pessoa**, **Funcionario** e **Aluno** ?? Justifique a resposta !*

Bibliografia

- **Java Básico e Orientação a Objeto (livro);**
  - Clayton Escouper das Chagas; Cássia Blondet Baruque; Lúcia Blondet Baruque.
  - Fundação CECIERJ, 2010: https://canal.cecierj.edu.br/012016/d7d8367338445d5a49b4d5a49f6ad2b9.pdf
- **Java e Orientação a Objetos (apostila);**
  - Caelum, 2023: https://www.caelum.com.br/apostila/apostila-java-orientacao-objetos.pdf
- **Modificadores de acesso em Java(artigo);**
  - DevMedia, 2023: https://www.devmedia.com.br/modificadores-de-acesso-em-java/18822
- **Get e Set - Métodos Acessores em Java(artigo);**
  - DevMedia, 2023: https://www.devmedia.com.br/get-e-set-metodos-acessores-em-java/29241
- **Sobrecarga e sobreposição de métodos em orientação a objetos(artigo);**
  - DevMedia, 2023: https://www.devmedia.com.br/sobrecarga-e-sobreposicao-de-metodos-em-orientacao-a-objetos/33066
- **Análise e modelagem de sistemas com a UML: com dicas e exercícios resolvidos (livro);**
  - Luiz Antônio Pereira, 2011: https://luizantoniopereira.com.br/downloads/publicacoes/AnaliseEModelagemComUML.pdf

Bibliografia

- **Herança em Orientação a Objeto(artigo);**
  - DevMedia, 2023: https://www.devmedia.com.br/conceitos-e-exemplos-heranca-programacao-orientada-a-objetos-parte-1/18579