Programação Orientada a Objetos

Aula 5 – Professor Diogo Orlando Nunes de Almeida

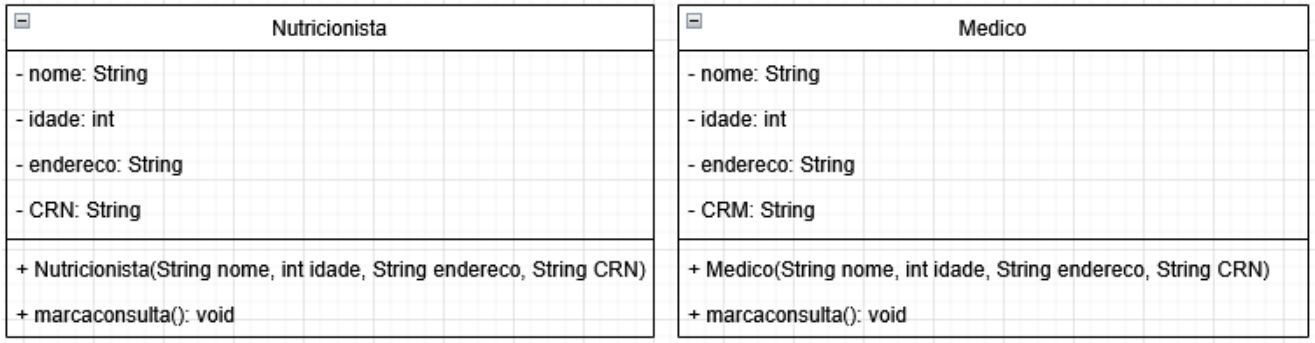
Conteúdo

- Conceitos do paradigma da Programação orientada a objetos;
- Diagrama de classes (UML);
- Introdução a classes e objetos;
- Atributos, métodos e interação entre objetos;
- Construtores e destrutores;
- Agregação e composição de objetos;
- Encapsulamento;
- Herança e polimorfismo;
- Tratamento de Exceções.

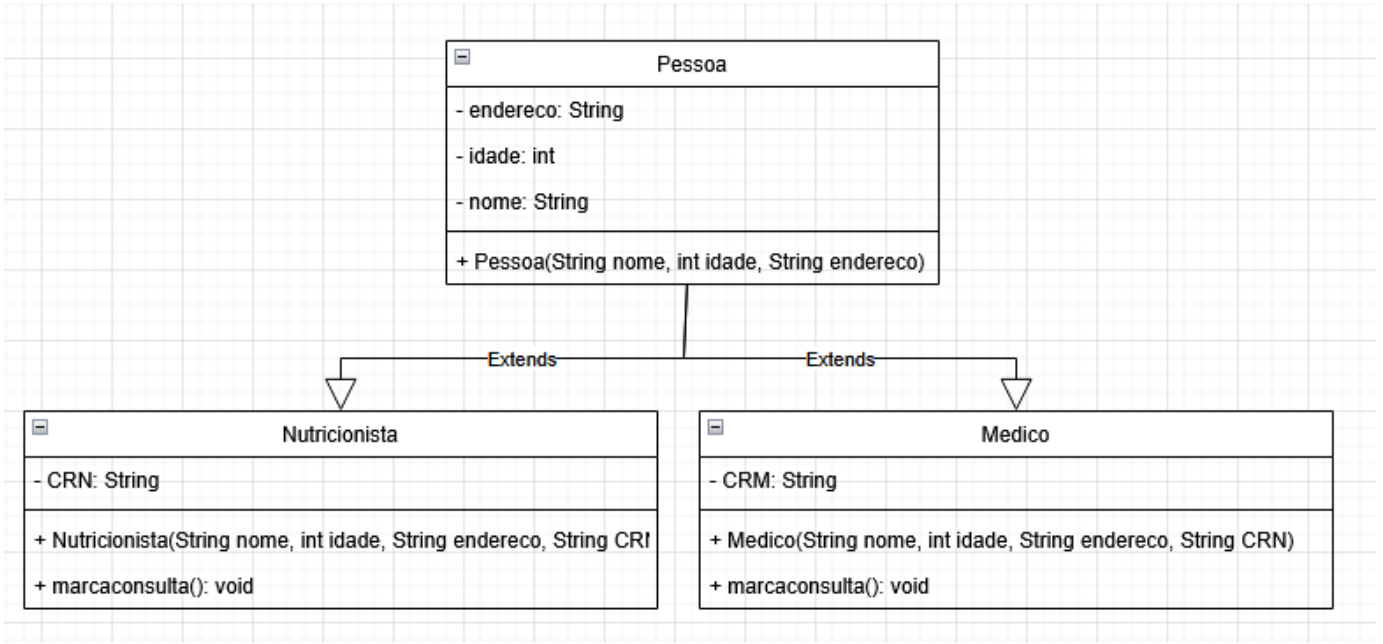
Herança

- Herança possibilita que os objetos de uma classe especifica podem utilizar os atributos e operações que iriam, de outra forma, somente estar disponíveis para os objetos de uma outra classe. (Adpt. Page-Jones, 2001, p. 33)
- De uma a infinitas classes podem herdar os métodos e atributos de outra classe.

Exemplo de herança



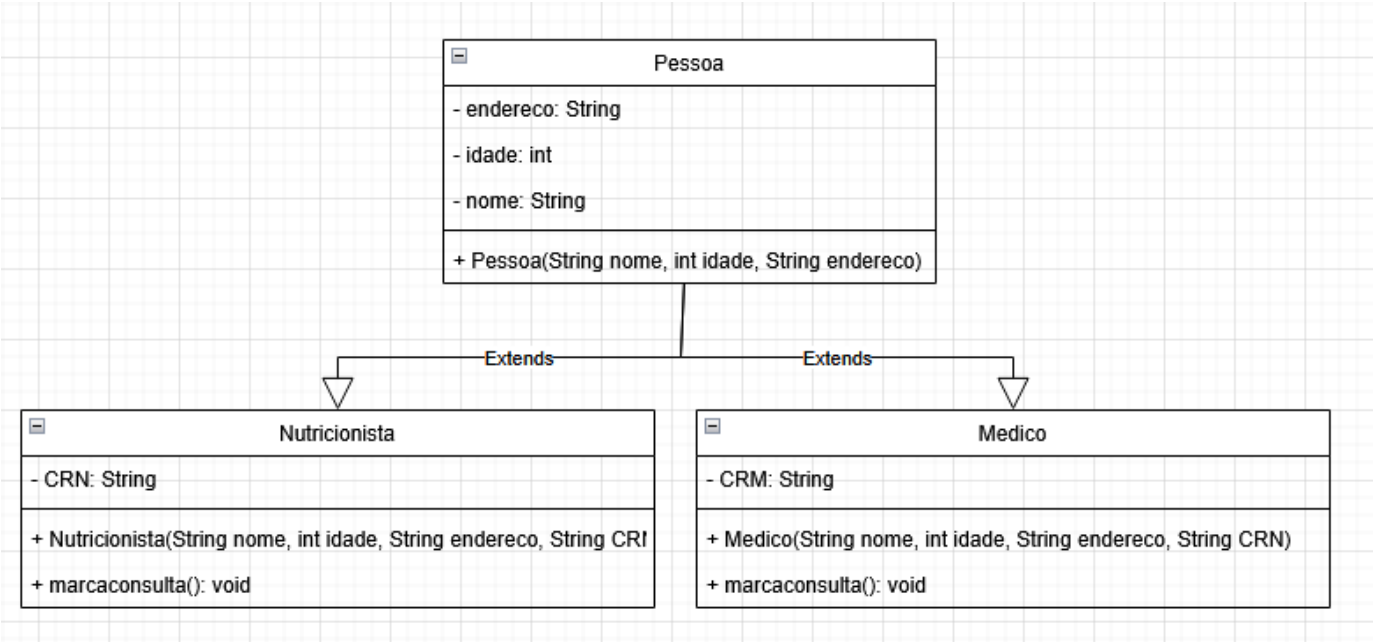
Sem herança



Com herança

Generalização

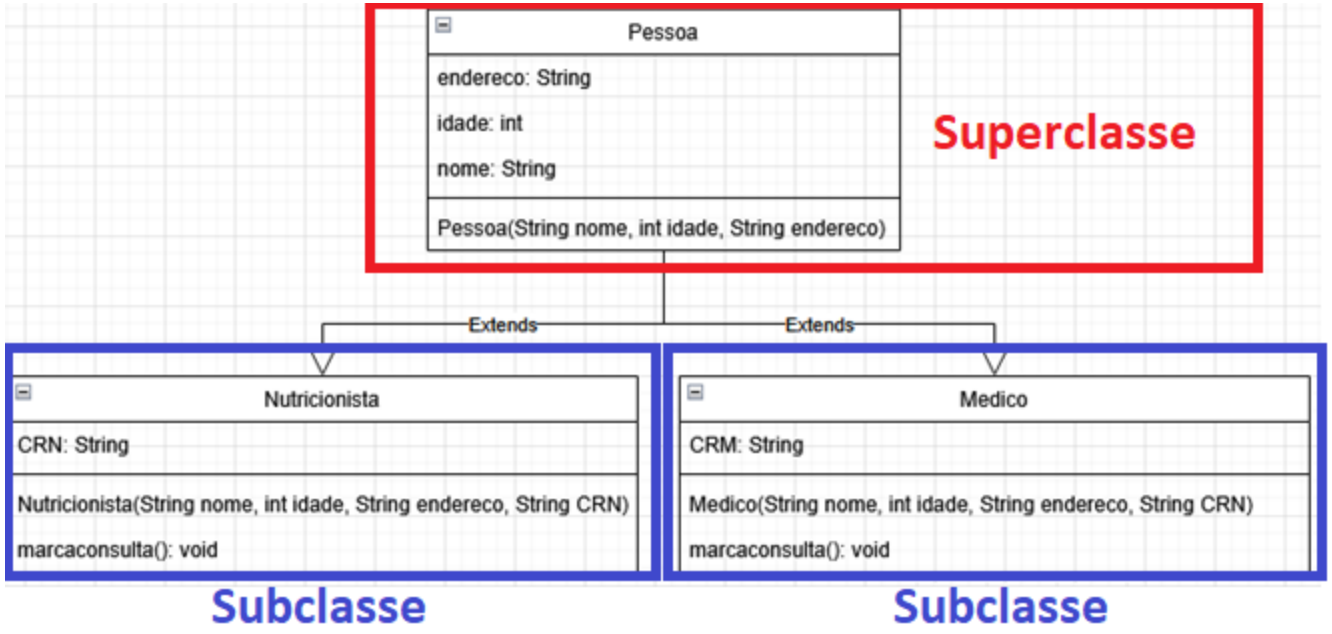
- A generalização permite que uma classe parametrizada tome uma classe como argumento sempre que um objeto for gerado. (Page-Jones, 2001, p. 33)



A Classe Pessoa é uma generalização das classes nutricionista e médico, pois seus atributos foram abstraídos por ela.

Superclasse e subclasse

- Superclasse é a classe pai em uma herança.
- Subclasse é uma classe filha que herda os atributos do pai.



Herança

- “extends”: representa a herança, nesse caso Pneu herda a classe carro.
- “super”: Inicializa o construtor da classe pai Carro.

```java
package Projeto;
public class Carro {
    private double velocidade;
    private double distancia;

    public Carro(double velocidade, double distancia){
        this.velocidade = velocidade;
        this.distancia = distancia;
        System.out.println("Objeto criado");
    }

    public double velocidade_media(){
        double media = this.velocidade/this.distancia;
        return media;
    }
}
```

Carro.java

```java
package Projeto;

public class Pneu extends Carro{
    private String marca;
    private int aro;

    public Pneu(double velocidade, double distancia, String marca, int aro) {
        super(velocidade, distancia);
        this.marca = marca;
        this.aro = aro;
    }
}
```

Pneu.java

Encapsulamento com herança

```java
package Projeto;

public class Pneu extends Carro{
    private String marca;
    private int aro;
    private int enchimento;

    public Pneu(double velocidade, double distancia, String marca, int aro) {
        super(velocidade, distancia);
        this.marca = marca;
        this.aro = aro;
    }

    public double velocidade_pneu() {
        double nova_velocidade;
        nova_velocidade = this.velocidade * this.enchimento;
        return nova_velocidade;
    }
}
```

Problems | Javadoc | Declaration | Console

2 errors, 7 warnings, 0 others

Description

- Errors (2 items)
  - The field Carro.velocidade is not visible
  - Type mismatch: cannot convert from double to int
- Warnings (7 items)

Pneu.java

Nome	Escopo
Public (+)	Em qualquer local.
Private (-)	Apenas na classe.
Protected (#)	Usado em herança, superclasse para as subclasses.

A classe Pneu não consegue acessar o atributo velocidade que herdou da classe Carro.java, pois está privado na classe Carro.java.

Encapsulamento com herança

```java
package Projeto;
public class Carro {
    protected double velocidade;
    private double distancia;

    public Carro(double velocidade, double distancia){
        this.velocidade = velocidade;
        this.distancia = distancia;
        System.out.println("Objeto criado");
    }

    public double velocidade_media(){
        double media = this.velocidade/this.distancia;
        return media;
    }
}
```

Carro.java

```java
package Projeto;

public class Pneu extends Carro{
    private String marca;
    private int aro;
    private int enchimento;

    public Pneu(double velocidade, double distancia, String marca, int aro) {
        super(velocidade, distancia);
        this.marca = marca;
        this.aro = aro;
    }

    public double velocidade_pneu() {
        double nova_velocidade;
        nova_velocidade = this.velocidade * this.enchimento;
        return nova_velocidade;
    }
}
```

Pneu.java

Após a troca de private para protected a falha desaparece.

Importante!

- Em C++ e Python uma classe pode herdar atributos e métodos de mais de duas classes simultâneas, embora isso não seja o mais correto.
- Em Java não se pode herdar atributos e métodos de mais de duas classes simultâneas, caso isso seja necessário, futuramente será estudado interfaces.

Atividades com o professor

- Aplicando um caso de herança em um supermercado, em cima de seus produtos como carne (vendido a quilo), limpeza (vendido a unidade), mercearia (vendido a unidade) e padaria (vendido a quilo).
- Cada produto vai ter uma taxa diferente de lucro.
- Produtos de mercearia e limpeza tem limitação de compra.

Exercícios

- Crie um diagrama UML implementando os conceitos de herança para uma calculadora com dois operandos.
- Crie um diagrama UML implementando os conceitos de herança para um sistema de hospital.
- Crie um diagrama UML implementando os conceitos de herança para um sistema industrial.
- Escolha um dos 4 diagramas anteriores e implemente em Java.

Referências

- PAGE-JONES, Meilir. **Fundamentos do desenho orientado a objeto com UML**. São Paulo: Pearson, 2001. E-book. Disponível em: https://plataforma.bvirtual.com.br. Acesso em: 25 mar. 2025.