UNIR | UNIVERSIDADE FEDERAL DE RONDÔNIA

Programação Orientada a Objetos

Aula 08 – Classes Abstratas

INF31098/INF31030

Prof. Dr. Jonathan Ramos

jonathan@unir.br

Departamento Acadêmico de Ciências de Computação – DACC

Núcleo de Tecnologia – NT

31/10/2022

Sumário

Revisão de Herança: SuperClasse **Pessoa** e **Aluno**

```java
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class Pessoa {
  private String nome, CPF;
  private LocalDate dataNasc;

  public Pessoa() {
  }

  public Pessoa(String n, String c, LocalDate d) {
    this.nome = n;
    this.CPF = c;
    this.dataNasc = d;
  }

  public String toString () {
    return "Nome: " + this.nome
        + "\nCPF: " + this.CPF
        + "\nNascimento: " + this.dataNasc.format(DateTimeFormatter.
     ofPattern("dd/MM/yyyy"));
  }
  // Getters and Setters
}
```

Revisão de Herança: SuperClasse **Pessoa** e **Aluno**

```java
import java.time.LocalDate;

public class Aluno extends Pessoa {
  private String matricula;

  public Aluno() {
    super();
  }

  public Aluno(String matricula, String nome, String CPF, LocalDate
     data) {
    super(nome, CPF, data);
    this.matricula = matricula;
  }

  public String toString() {
    return super.toString() + "\nMatricula: " + this.matricula;
  }
  // Getters and Setters
}
```

Herança

Note que agora temos:

- **extends**: invoca a classe pessoa, e adiciona ela como se fosse parte da classe Aluno (estende-a, por isso **extends**);
- **super()**: seta os valores da superclasse Pessoa, por isso **super**.

Vendo o resultado no main:

```
1  import java.time.LocalDate;
2  public class Main {
3    public static void main(String []args) {
4        Definindo Valores
5      String nome = "Jonathan Ramos";
6      String cpf = "123.456.789-00";
7      LocalDate d = LocalDate.of(1991, 9, 29);
8      String matricula = "123456";
9      // Instanciando a classe Aluno
10     Aluno aluno = new Aluno(matricula, nome, cpf, d);
11     //Outra Forma de instanciar
12     // Aluno aluno = new Aluno("123456", "Jonathan Ramos",
       "123.456.789-00", LocalDate.of(1991, 9, 29));
13     // Imprimindo valores com toString;
14     String minhaString = aluno.toString();
15     System.out.println(minhaString);
16   }
17 }
```

Sumário

Classes abstratas

Classe abstratas

- Classe onde **terá métodos abstratos**, ou seja, métodos **não implementados** que serão **obrigatoriamente** implementados **quando forem herdados por outra classe**.
- Contudo, a classe abstrata ainda assim poderá conter variáveis e métodos próprios, que podem ser herdados pela subclasse;
- **Não poderá ser instanciada!**

Classes abstratas

Classe abstratas

- Classe onde **terá métodos abstratos**, ou seja, métodos **não implementados** que serão **obrigatoriamente** implementados **quando forem herdados por outra classe**.
- Contudo, a classe abstrata ainda assim poderá conter variáveis e métodos próprios, que podem ser herdados pela subclasse;
- **Não poderá ser instanciada!**

Polimorfismo

O polimorfismo permite que **classes abstratas consigam receber comportamentos através de classes concretas**. Por exemplo:

- USB: seria uma **classe abstrata** enquanto os **dispositivos** (pendrive, mouse, teclado, etc) seriam as **classes concretas**.
- USB é uma especificação que **pode ter várias implementações** com características diferentes.

Exemplo de classe abstrata

Exemplo para implementar no eclipse:

- Classe Abstrata: **Pagamento**;
- Classes concretas:
  - **Dinheiro**: 10% de desconto;
  - **CartaoCredito**: em 1x sem juros, mais que 1x, 1.5% de juros a.m.;
  - **Cheque**[a]: 3% de juros para 30 dias, 6% para 60 dias, 9% para 90 dias;

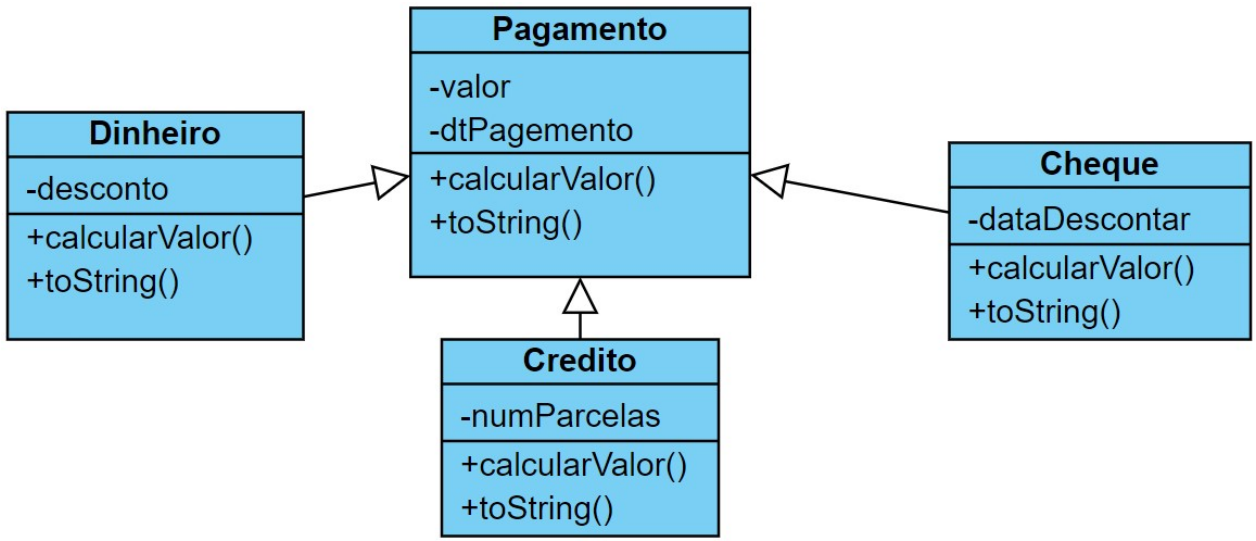

Figura: Representação Gráfica do Modelo de tipos de pagamento.

[a] Confiamos no cliente e o cheque n vai voltar!

Classe Abstrata: **Pagamento**

Note que a classe/método abstrato não possui implementação:

```java
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
// SuperClasse Abstrata
abstract class Pagamento {
  private double valor;
  private LocalDate data;
  // Construtor sem Parâmetros
  public Pagamento() {
    this.setValor(0);
    this.setData(LocalDate.now());
  }
  // Construtor com Parâmetros
  public Pagamento(double v, LocalDate d) {
    this.setValor(v);
    this.setData(d);
  }
  // Método abstrato não implementado aqui, porém obrigatório nas
     subclasses
  public abstract double calcularValor();
  // Imprimindo os valores da classe
  public String toString() {
    return String.format("Valor: R$ %,.2f", this.valor) + "\nData: "
    + this.data.format(DateTimeFormatter.ofPattern("dd/MM/yyyy"));
  }
  // Getters and setters
}
```

Classe: **Cheque**

```java
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;

public class Cheque extends Pagamento {
  private LocalDate dataDescontar;
  public Cheque() { // Dia atual mais trinta dias
    super(); this.dataDescontar = LocalDate.now().plusDays(30);
  }
  public Cheque(double v, LocalDate dDesc, LocalDate d) {
    super(v, dDesc); this.dataDescontar = d;
  }
    // Obrigatoriamente deve ser implementado
  public double calcularValor() {
    double valor = 0;
    int diffDias = this.dataDescontar.compareTo(this.getData());
    if (diffDias < 30) { // 3%
      valor = this.getValor() + this.getValor() * 0.03;
    } // implementar os demais juros
    return valor;
  }
  public String toString() {
    return super.toString() + "\nData Descontar: " + this.
     dataDescontar.format(DateTimeFormatter.ofPattern("dd/MM/yyyy"))
     +String.format("\nValor Pagar: R$ %,.2f",this.calcularValor());
  }
  // Getters and setters
}
```

Sumário

Exercício Prático

Criar projeto para o seguinte mapeamento:

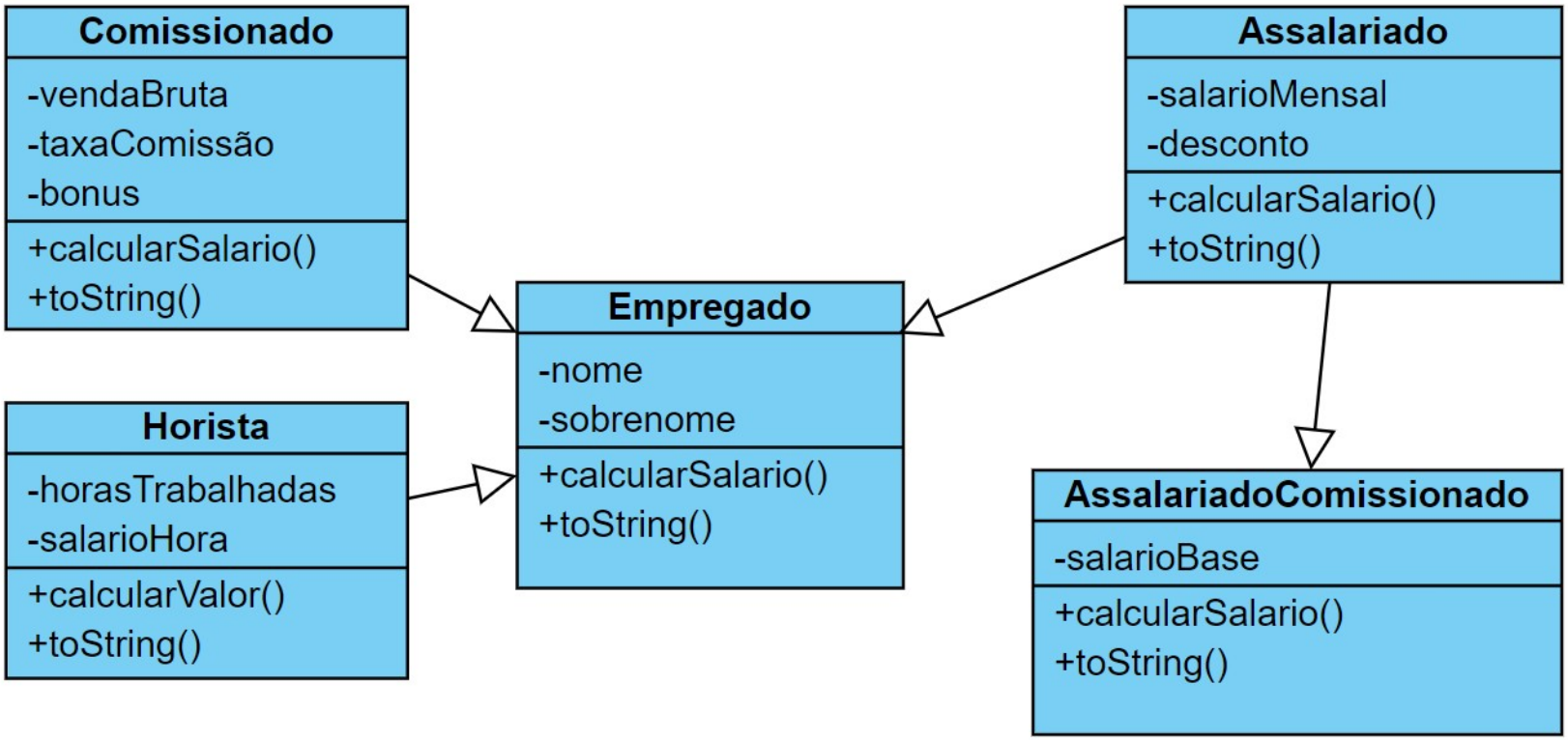

Figura: Exercício prático no eclipse.

Regras do mapeamento I

Empregado

Classe abstrata, precisa ter dois construtores, um vazio e um com os parâmetros. Precisa ter uma função abstrata **+calcularSalario()**

Assalariado

Para calcular o salário, será apenas subtraído o valor do **-desconto** de **-salarioMensal**:

$$valorSalario = salarioMensal - desconto \quad (1)$$

AssalariadoComissionado

Para calcular o salário, o **-salárioBase** será somado com o resultado da Equação 1.

Horista

Para calcular o salário:

$$\textbf{sal} = \begin{cases} \textbf{horasTrab} \times \textbf{salarioHora}, & \text{se } \textbf{horasTrab} \leq 40 \\ (\textbf{salarioHrs} * 40) + (\textbf{hrsExtra} * (\textbf{salarioHrs} * 1.5)), & \text{se } \textbf{hrsTrab} > 40 \end{cases} \quad (2)$$

Regras do mapeamento II

Comissionado

Para calcular o salário:

$$\textbf{sal} = (\textbf{brutoVendas} * (\frac{\textbf{taxaComissao}}{100})) + \textbf{bonus} \quad (3)$$

FIM!

jonathan@unir.br