Linguagem de Programação Orientada a Objeto

LPOO 06- Polimorfismo Sobrescrita e Sobrecarga

Prof. Msc Wanderley Gonçalves Freitas

Wanderley.unip@gmail.com

www.professor.wanderley.nom.br

Agenda

- Conceitos básicos de polimorfismo
- Os Tipos de Polimorfismo :
  - *Polimorfismo Dinâmico - Sobrescrita ou redefine o método (Override);*
  - Polimorfismo Estático : Sobrecarga de métodos (Overload).
- Conceito de assinatura de Método
- Exercícios de fixação
  - Sobrescrita

Objetivo

- Capacitar o aluno a compreender os conceitos de orientação a objetos e sua utilização no desenvolvimento de software de qualidade numa linguagem de programação moderna (JAVA)
- Compreender o conceito de polimorfismo.
- Utilizar métodos sobrescritos para executar o polimorfismo.

Polimorfismo

- O termo Polimorfismo origina-se do grego e quer dizer **"o que possui várias formas".**

- *Polimorfismo* significa que a mesma operação (método) pode se comportar de forma diferente em classes diferentes.

- **Polimorfismo** - princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura) mas comportamentos diferentes para cada umas das classes derivadas(subclasse).

-

Tipos de Polimorfismo

- São dois os tipos de Polimorfismo :
  - **Polimorfismo Dinâmico** - Sobrescrita ou redefinição o método (**Override**);
  - **Polimorfismo Estático** : Sobrecarga de métodos (**Overload**).
- **O Polimorfismo Dinâmico** acontece na **herança**, quando a subclasse sobrepõe o método original. Agora o método escolhido se dá em **tempo de execução.** A escolha de qual método será chamado depende do tipo do objeto que recebe a mensagem (invocação do método).
- **O Polimorfismo Estático** se dá quando temos a mesmo método implementada várias vezes na mesma classe. A escolha de qual método será chamada depende da assinatura dos métodos sobrecarregados, ou seja, em **tempo de compilação**.

Tipos de Polimorfismo

- Qual a diferença entre sobrescrita(override) e sobrecarga(overload) de metodos?

  - **Sobrescrita** são métodos de mesma assinatura existindo em **classes relacionadas por herança**, direta ou indiretamente.

  - **Sobrecarga** são métodos na **mesma classe**, com o mesmo nome mas com assinaturas diferentes (com argumento diferente)

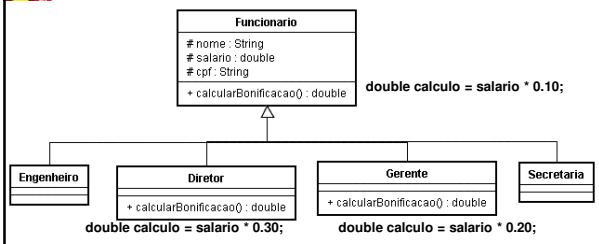
Assinatura de Método

- Assinatura de um método é uma combinação do nome do método, tipo; ordem e número de seus parâmetros.

Métodos	Assintura do método
public double calcularPagamento(double valor)	calcularPagamento(double)
public void calcularImposto()	calcularImposto()
public double processarMedia(double nota1, double nota2)	processarMedia(double, double)
public double processarMedia(double nota1, double nota2, int peso1, int peso2)	processarMedia(double, double , int , int)
public void exibeDados(String pnome, String endereço)	exibeDados(String, String)

Tipos de Polimorfismo

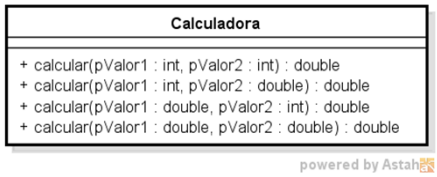
Exemplo (1) Sobrescrita (**Override**);



•**Sobrescrita** são métodos de mesma assinatura existindo em **classes relacionadas por herança**, direta ou indiretamente.

Tipos de Polimorfismo

Exemplo (2) Sobrecarga de métodos (**Overload**).



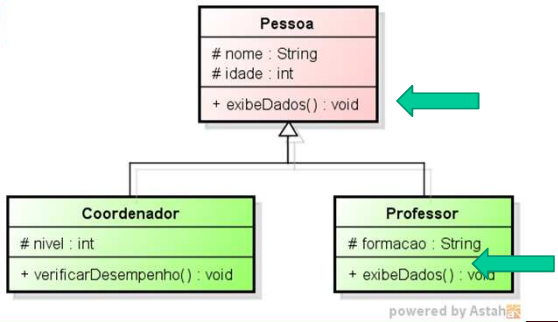
•**Sobrecarga** são métodos na mesma classe, com o mesmo nome mas com assinaturas diferentes.

Sobrescrito (Override)

- Permite a existência de vários métodos com **assinaturas idênticas**, porém com **implementações diferentes**.
- *Quando um método tem uma **implementação na classe pai** e outra **implementação na classe filha***
- Quando uma subclasse redefine um método de sua superclasse a fim de prover ao método um comportamento mais adequado às suas características
- *Para sobrescrever um método na subclasse é necessário que o tipo de retorno, o nome e os parâmetros e seus tipos sejam exatamente iguais.*

Sobrescrita (Override) em Java

Exemplo (3)



de) em Java

```
3  public class Pessoa {
4      protected String nome;
5      protected int idade;
6
7      public void exibeDados(){
8          System.out.println("metodos - pessoa");
9          System.out.println("nome : " + nome);
0          System.out.println("idade  " + idade);
```

```
3  public class Professor extends Pessoa{
4      protected String formacao;
5
6      public void exibeDados(){
7          System.out.println("metodos - sobrecritos");
8          System.out.println("nome : " + nome);
9          System.out.println("formacao: " + formacao);
```

```
3  public class Coordenador extends Pessoa{
4      protected int nivel;
5      public void verificarDesempenho(){
6          double resultado = nivel + 2;
7          if (resultado > 10 )
8              resultado =10;
9          System.out.println("nome : " + nome);
0          System.out.println("idade :  " + idade);
1          System.out.println("nivel:  " + resultado);
```

ide) em Java

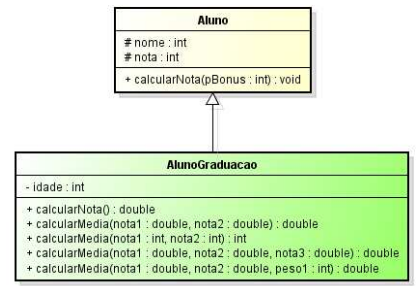
```
blic class TesteSobrecrito {

  public static void main(String[] args) {
      Pessoa luis = new Pessoa();
      luis.setNome("luis");
      luis.setIdade(34);
      luis.exibeDados();

      System.out.println("");
      Professor marcia = new Professor();
      marcia.setNome("paulo");
      marcia.setIdade(67);
      marcia.setFormacao("matematica");
      marcia.exibeDados();

      System.out.println("");
      Coordenador paulo = new Coordenador
      paulo.setNome("paulo");
      paulo.setIdade(67);
      paulo.exibeDados();
```

```
<terminated> TesteSobrecrito [Java Applicat
metodos - pessoa
nome : luis
idade  34

metodos - sobrecritos
nome : paulo
formacao: matematica

metodos - pessoa
nome : paulo
idade  67
```


Sobrecarga(overload) em Java

Exemplo (4)

- Permite a existência de vários métodos de mesmo **nome**, porém com **assinaturas** diferentes (tipo e qtd de p



ga(overload)

```
public class Aluno {
    protected String nome;
    protected int nota;

    public void calcularNota(int pBonus){
        double resultado = nota + pBonus;
        if (resultado > 10 )
            resultado =10;
        System.out.println("nome : " + nome);
        System.out.println("nota:  " + resultado);
    }
```

```
public class AlunoGraduacao extends Aluno{
    private int idade;
    public void calcularNota() {
        double resultado = nota + 3;
        System.out.println("nome : " + nome);
        System.out.println("nota:  " + resultado);
    }
    // metodo 1
    public double calcularMedia(double nota1, double nota2) {
        double calculo = (nota1 + nota2)/2;
        return calculo;
    }
    // metodo 2
    public int calcularMedia(int nota1, int nota2) {
        int calculo = (nota1 + nota2)/2;
        return calculo;
    }
    // metodo 3
    public double calcularMedia(double nota1, double nota2, double nota3) {
        double calculo = (nota1 + nota2 + nota3)/3;
        return calculo;
    }
    // metodo 4
    public double calcularMedia(double nota1, double nota2, int peso1) {
        double calculo = ((nota1 * peso1) + (nota2 * peso1)) / (peso1);
        return calculo;
```


oad)

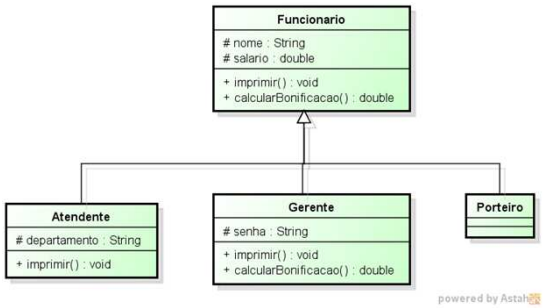
```
public class TesteSobrecarga {

    public static void main(String[] args) {
        Aluno marcos = new Aluno();
        marcos.setNome("Marcos");
        marcos.setNota(6);
        marcos.calcularNota(8);

        System.out.println("");

        AlunoGraduacao alunoGraduacao = new AlunoGraduacao();
        alunoGraduacao.setNome("katia");
        alunoGraduacao.setNota(6);
        alunoGraduacao.calcularNota();
        System.out.println("calcularMedia(double, double): " +
                           alunoGraduacao.calcularMedia(7.0, 6.0));

        System.out.println("calcularMedia(int, int) : " +
                alunoGraduacao.calcularMedia(5, 8));

        System.out.println("calcularMedia(double, double, int) : " +
                alunoGraduacao.calcularMedia(5,4,3));

        System.out.println("calcularMedia(double, double, double): " +
                           alunoGraduacao.calcularMedia(6, 8.5,9.5));
```

```
nome : Marcos
nota:  10.0

nome : katia
nota:  9.0
calcularMedia(double, double): 6.5
calcularMedia(int, int) : 6
calcularMedia(double, double, int) : 9.0
calcularMedia(double, double, double): 8.0
```


Sobrescrita (Override) - Exercício

Sobrescrita (Override) – Exercício(1)

- **Exercício de Fixação::** Observe o diagrama de classes de gere todas as classes na ferramenta eclipse



Exercícios(1)

- 1 – Crie a classe Funcionario.
  - Pacote : br.unip.lpoo.np2.si2p30.
    - Atributos (protected) : nome(String), salario(double).
    - Cria os seguintes método indicados na tabela?

Tipo Retorno	Nome método	Parâmetro	Objetivo do método - corpo
void	imprimir	ausente	❖Regra de Negócio – exibe no console todos os atributos da classe
double.	calcularBonificacao	ausente	Dever retorna o resultado da formula: ❖ Regra de Negócio – Para qualquer Funcionário, a bonificação é 10% sobre o valor do salario **Formula** : vCalculo=(salario * 0.10) + salario

Exercícios(1)

- 2 - Crie classe Atendente subclasse da classe Funcionario:
  - Atributo(protected) : departamento(String).
  -
- O método imprimir será sobrescrever (sobrescritos, redefine, override), conforme tabela

Nome método	Objetivo do método - corpo
imprimir	❖Regra de Negócio – Exibe a uma mensagem "Categoria Especial" e todos os atributos da classe no console

Exercícios(1)

- 3 - Crie classe Gerente subclasse da classe Funcionario:
  - Atributo(protected) : senha(String)
- Os métodos calcularBonificacao e imprimir serão sobrescrever (sobrescritos, redefine, override), conforme tabela

Nome método	Objetivo do método - corpo
imprimir	❖Regra de Negócio – Exibe a uma mensagem "Categoria Executivo" e todos os atributos da classe no console
calcularBonificacao	Dever retorna o resultado da formula: ❖ Regra de Negócio – Para qualquer Gerente, a bonificação é 30% sobre o valor do salario **Formula** : vCalculo=(salario * 0.30) + salario

Exercícios(1)

- 4 - Crie classe Porteiro subclasse da classe Funcionario:

Resposta(1)

```
public class Funcionario {

    protected  String nome;
    protected double salario;


    public void imprimir() {
        System.out.println("nome:  " + nome );
        System.out.println("salario: " + salario);
    }

    public double calcularBonificacao(){
        double calculo = salario * 0.10;
        return calculo;
    }
```

```
public class Atendente extends Funcionario {

    protected String departamento;

    public void imprimir() {
        System.out.println("Categoria Especial");
        System.out.println("nome:  " + nome );
        System.out.println("salario: " + salario);
        System.out.println("departamento: " + departamento);
    }
```


Resposta(1)

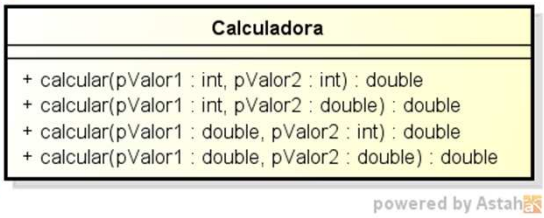
```
public class Gerente extends Funcionario {

    protected String senha;

    public void imprimir() {
        System.out.println("Categoria Especial");
        System.out.println("nome:  " + nome );
        System.out.println("salario: " + salario);
        System.out.println("senha: " + senha);
    }

    public double calcularBonificacao(){
        double calculo = salario * 0.30;
        return calculo;
    }
```

```
public class Porteiro extends Funcionario{

}
```

SobreCarga(overload) - Exercício

Sobrecarga(overload) – Exercício(2)

- **Exercício de Fixação::** Observe o diagrama de classe de gere a classe na ferramenta eclipse



Sobrecarga(overload) – Exercício(2)

1 – Crie a classe Calculadora.

- Pacote : br.unip.lpoo.np2.si2p30.
- Cria os seguintes métodos indicados na tabela?

Tipo Retorno	Nome método	Parâmetro	Objetivo do método - corpo
double	calcular	pValor1 (int), pValor2(int)	**Formula** : vResultado = (pValor1 + pValor2
double	calcular	pValor1 (double), pValor2(int)	**Formula** : vResultado = (pValor1 * pValor2
double	calcular	pValor1 (int), pValor2(double)	**Formula** : vResultado = (pValor1 / pValor2
double	calcular	pValor1(double), pValor2(double)	**Formula** : vResultado = (pValor1 % pValor2

Resposta(2)

```
public class Calculadora {

    public double calcular(int valor1, int valor2) {
        double calculo = valor1 + valor2;
        return calculo;
    }

    public double calcular(double valor1, int valor2) {
        double calculo = valor1 * valor2;
        return calculo;
    }

    public double calcular(int valor1, double valor2) {
        double calculo = valor1 / valor2;
        return calculo;
    }

    public double calcular(double valor1, double valor2) {
        double calculo = valor1 % valor2;
        return calculo;
    }
```


Obrigado

Wanderley

Wanderley.unip@gmail.com

www.professor.wanderley.nom.br

Referências Bibliográficas

BRUCE, Eckel. **Thinking in Java**. 4 ed. Massachusetts: Editora Prentice hall: 2006.

FOWLER, Martin. **UMl essencial : um breve guia para a linguagem** – padrão de modelagem de objetos. Tradução Vera Pezerico e Christian Thomas Prices. 3. ed. Porto Alegre : Bookman, 2005.

FURLAN, Jose Davi **Modelagem de Objetos Através da UML** São Paulo: **Editora Makron Books, 2000, ISBN 8534609241,**

GRADY, Booch; RUMBAUGH, James; JACOBSON, Ivar. **UML, guia do usuário** . tradução de Fábio Freitas da Silva. Rio de Janeiro: Editora Campus. 2000.

SIERRA, kathy: BATES, Bert. **Certificação Sun para programadores e desenvolvedor Java 2**. Rio de Janeiro: Editora Altas Books. 2003.

SIERRA, kathy: BATES, Bert. **Head first java 5**. 2 ed. Sebastopolo: Editora O' Reilly Media. Books. 2005.

TOM, Pender. **UML, a bíblia.** Rio de Janeiro: Elsevier, 2004.