Programação Orientada a Objetos

Professor: Anderson Elias

Estácio

1

Polimorfismo

Contato:
anderson.elias@estacio.br

Estácio

2

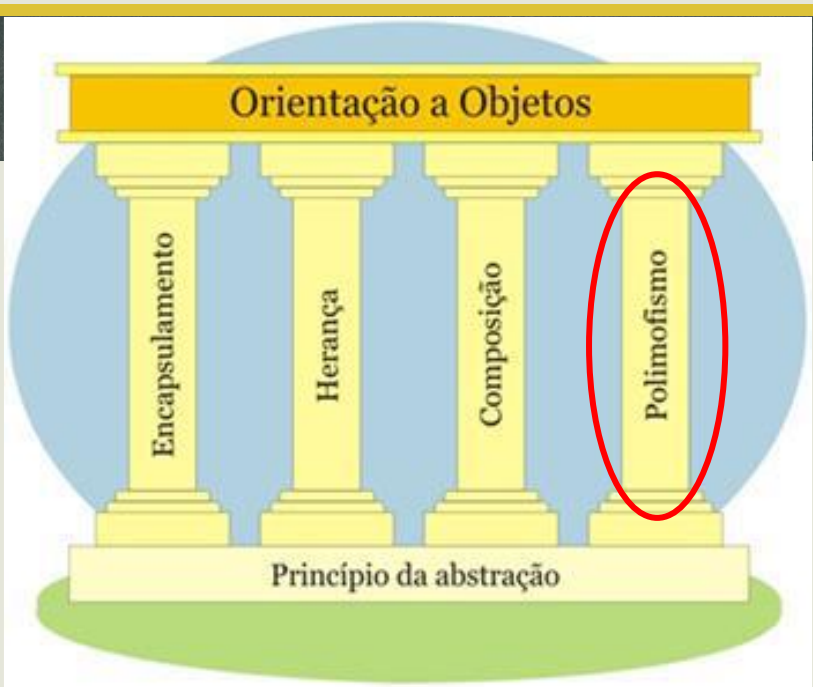
"Quando penso que cheguei ao meu limite, descubro que tenho forças para ir além."

Ayrton Senna

Contato:
anderson.elias@estacio.br

Estácio

3



4

Polimorfismo

- A herança é um poderoso mecanismo de **especialização**;
- Já o polimorfismo traz um importante mecanismo para a **generalização**;
- No polimorfismo podemos criar objetos a partir de uma superclasse onde este pode assumir diferentes comportamentos, que depende de certas condições;

5

Polimorfismo

Polimorfismo significa "*muitas formas*".

O polimorfismo indica o princípio de que o comportamento pode variar, dependendo do tipo real de um objeto.

A mesma computação funciona para objetos de muitas formas e adapta-se à natureza dos objetos.

6

Polimorfismo

Em Java, todos os métodos de instância são polimórficos.

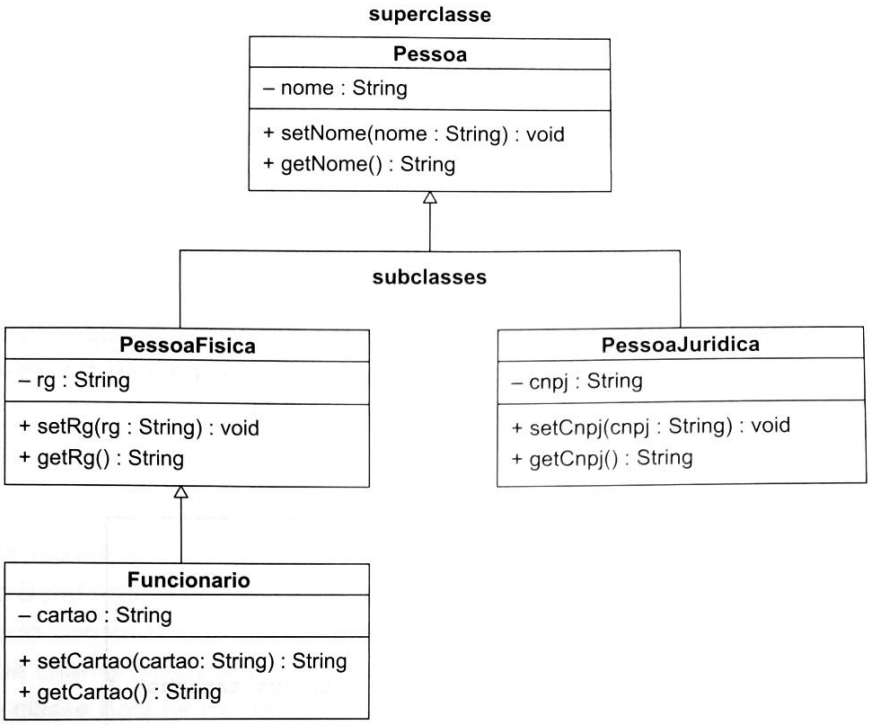
Métodos de Instância:
São métodos não estáticos e que para serem invocados é preciso você instanciar um objeto e invocar o método a partir desse objeto.

7

Polimorfismo

Codifique em java

Este diagrama de classe.



8

Polimorfismo

Ao final de cada uma das quatro classes, adicione um método chamado **mostrarClasse** como segue:

```
public void mostrarClasse() {
    System.out.println("Classe Pessoa.");
}
```

Para cada uma imprima o nome da classe respectivamente:

"Classe Pessoa." , *"Classe PessoaFísica."* ,

"Classe PessoaJurídica." e *"Classe Funcionário."*

9

Polimorfismo

```
public class Pessoa {
  private String nome;
  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public void mostrarClasse() {
    System.out.println("Classe Pessoa.");
  }
}
```

10

Polimorfismo

```
public class PessoaFisica extends Pessoa{
  private String rg;
  public String getRg() {
    return rg;
  }
  public void setRg(String rg) {
    this.rg = rg;
  }
  public void mostrarClasse() {
    System.out.println("Classe PessoaFísica.");
  }
}
```

11

Polimorfismo

```
public class PessoaJuridica extends Pessoa{
  private String cnpj;

  public String getCnpj() {
    return cnpj;
  }

  public void setCnpj(String cnpj) {
    this.cnpj = cnpj;
  }
  public void mostrarClasse() {
    System.out.println("Classe PessoaJurídica.");
  }
}
```

12

Polimorfismo

```
public class Funcionario extends PessoaFisica{
  private String cartao;

  public String getCartao() {
    return cartao;
  }

  public void setCartao(String cartao) {
    this.cartao = cartao;
  }
  public void mostrarClasse() {
    System.out.println("Classe Funcionário.");
  }
}
```

13

Polimorfismo

```
public class PessoaPolimorfica {
public static void main(String[] args) {
  Pessoa pessoa = null;
  int tipo = Integer.parseInt(
      JOptionPane.showInputDialog(
      "Digite um número de 1 a 4."));
  switch (tipo) {
    case 1:
      pessoa = new Pessoa();
      break;
    case 2:
      pessoa = new PessoaFisica();
      break;
    case 3:
      pessoa = new PessoaJuridica();
      break;
    case 4:
      pessoa = new Funcionario();
      break;
    default: {
      System.out.println("Tipo Inválido.");
      System.exit(0);
      break;
      }
    }
  pessoa.mostrarClasse();
  }
}
```

14

Polimorfismo

Sobre os métodos polimórficos, é quando duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (*assinatura*);

Estes métodos podem ter a mesma identificação, mas seus comportamentos são distintos, especializados para cada classe derivada.

15

Polimorfismo

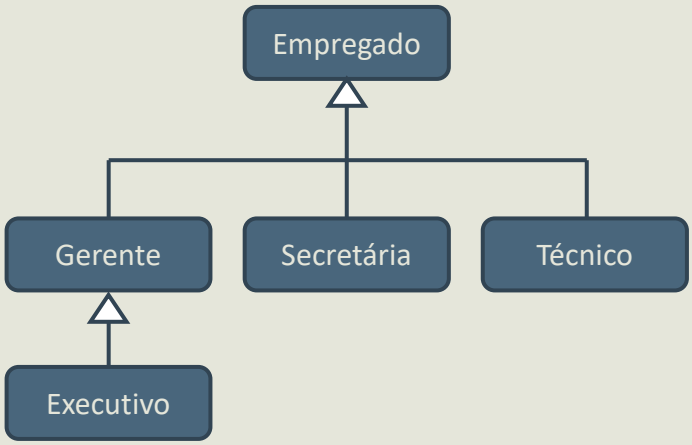
Para acontecer o polimorfismo, é necessário que os métodos tenham exatamente a mesma identificação.

Quando o polimorfismo está sendo utilizado, o comportamento que será adotado por um método só será definido durante a execução.

16

Polimorfismo

Exemplo:
Precisamos listar o nome e o salário de vários empregados, porém para o empregado Gerente, terá um acréscimo a título de *Bonus* em seu salário por ser Executivo.



17

Polimorfismo

Exemplo:
Precisamos listar o nome e o salário de vários empregados, porém para o empregado Gerente, terá um acréscimo a título de *Bonus* em seu salário.

```
public class Empregado {
  private String nome;
  private double salario;
  public Empregado(String n, double s){
    nome = n;
    salario = s;
  }
  public String getNome(){
    return nome;
  }
  public double getSalario(){
    return salario;
  }
  public void atualizaSalario(double percentual){
    double atualizacao = (salario * percentual) /100;
    salario +=atualizacao;
  }
}
```

18

Polimorfismo

Exemplo:
Precisamos listar o nome e o salário de vários empregados, porém para o empregado Gerente, terá um acréscimo a título de *Bonus* em seu salário.

```
public class Gerente extends Empregado{
  private double bonus;
  public Gerente(String n, double s){
    super(n, s);
    bonus = 0;
  }
  public double getSalario(){
    double salarioBase =  super.getSalario();
    return salarioBase + bonus;
  }
  public void setBonus(double b){
    bonus = b;
  }
}
```

19

Polimorfismo

Exemplo:
Precisamos listar o nome e o salário de vários empregados, porém para o empregado Gerente, terá um acréscimo a título de *Bonus* em seu salário.

```
public class GerenteTeste {
  public static void main(String[]args){
    Gerente chefe = new Gerente("José", 80000);
    chefe.setBonus(5000);
    Empregado [] pessoal = new Empregado[3];
    pessoal[0] = chefe;
    pessoal[1] = new Empregado("Maria", 5000);
    pessoal[2] = new Empregado("João", 4500);
    for(Empregado e : pessoal){
      System.out.println("Nome= "+e.getNome()
          +" , salário="+e.getSalario());
    }
  }
}
```

20

Polimorfismo

Observamos que cada gerente *é um* empregado.

```
Empregado e;
e = new Empregado("Pedro", 3000);
e = new Gerente("Carla", 5500);
```

Em Java as variáveis de objetos são polimorfas.

Portanto, uma variável do tipo Empregado pode se referir a uma classe Empregado como: *Gerente*, e outras subclasses derivadas de Empregado.

21

Polimorfismo

Observamos que neste ponto do código, foi tirada vantagem do polimorfismo:

```
Gerente chefe = new Gerente("José", 80000);
Empregado [] pessoal = new Empregado[3];
pessoal[0] = chefe;
```

Aqui **pessoal[0]** e chefe se referem ao mesmo objeto, mas a variável **pessoal[0]** é um objeto Empregado.

22

Polimorfismo

Por isto, podemos usar:

```
chefe.setBonus(5000); //OK
```

E não podemos usar:

```
pessoal[0].setBonus(5000); //ERRO
```

O tipo declarado **`pessoal[0]`** é Empregado, e o método **`setBonus`** não é um método da classe Empregado.

23

Polimorfismo

Neste sentido, não se pode atribuir uma referência da superclasse em uma variável da subclasse.

Exemplo:

```
Gerente g = pessoal[0];
```

Observe que para isto é muito simples, pois nem todos os funcionários são gerentes.

24

Polimorfismo

LISTA 8

25

Dúvidas?

26