

Programação III

Polimorfismo
Jocélio Passos
joceliopassos@bol.com.br

Polimorfismo - Conceitos

- Deitel Cap 9, páginas 405... [454...]
- Sebesta Cap 11 pág 421, 422
- Visto como o conceito mais simples da OOP
- Do grego (polys e morfê) : muitas formas
- Diferentes classes podem definir atributos e métodos como o mesmo nome
- Quando uma operação assume **muitas formas** de implementação, dependendo do tipo do objeto
- Mensagem do mesmo nome em classes diferentes
- Permite escrever rotinas e métodos que operam com qualquer classe

2/2/2007

2

Polimorfismo - Exemplo

- Pessoa.nome;
- Felino.nome;
- Pessoa.correr ();
- Felino.correr ();
- cubo.volume ();
- caixaDagua.volume ();

2/2/2007

3

Exemplo sem Polimorfismo

- class Casa
- {
- void abrirTudo ()
- {Para cada componente (objeto) da casa
- { switch (objeto)
- { case janela : objeto.janelaAbrir ();
- case porta : objeto.portaAbrir ();
- case basculante : objeto.basculanteAbrir ();
- }
- }
- }...

2/2/2007

4

Usando Polimorfismo e Vinculação Dinâmica

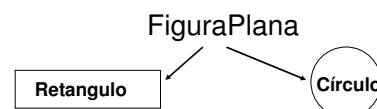
- class Casa
- {void abrirTudo ()
- {Para cada componente (objeto) da casa
- { objeto.abrir ();}
- } ...
- O método "abrir ()" é chamado para cada componente por meio da variável polimórfica "objeto".
- Este processo realizado em tempo de execução é chamado **Vinculação Dinâmica**.
- Deitel 405,415 [454,465] e Sebesta 421, 453 e 454

2/2/2007

5

Polimorfismo - Exemplo com Código

- A partir da superclasse FiguraPlana criar as subclasses Retangulo e Circulo. Criar um programa LerFiguraPlana que instancie as duas subclasses, leia seus atributos e mostre seus dados. Use o polimorfismo.



2/2/2007

6

Polimorfismo - Classe FiguraPlana

```
class FiguraPlana {
    double area ()
    {return (0);}
    double perimetro ()
    {return (0);}
    void mostrarDados () {
        System.out.println ("\n Nome do Objeto : " +
            this.toString ());
        System.out.println ("\nArea = " + this.area ());
        System.out.println ("\nPerimetro = " +
            this.perimetro());
    }
}
```

2/2/2007

7

Polimorfismo - Classe Retângulo

```
class Retangulo extends FiguraPlana {
    private float ladoa;
    private float ladob;

    public Retangulo (float ladoa, float ladob)
    {this.ladoa = ladoa;
    this.ladob = ladob;}

    public float getLadoa ()
    {return (this.ladoa);}
}
```

2/2/2007

8

Polimorfismo - Classe Retângulo

```
public void setLadoa (float ladoa)
{this.ladoa = ladoa;}

public float getLadob ()
{return (this.ladob);}

public void setLadob (float ladob)
{this.ladob = ladob;}

double area () // override
{return this.ladoa * this.ladob;}
```

2/2/2007

9

Polimorfismo - Classe Retângulo

```
double perimetro () // override
{return 2 * (this.ladoa + this.ladob);}

void mostrarDados () // override
{System.out.println ("Retangulo com lados " +
    this.ladoa + " e " + this.ladob);
    super.mostrarDados ();
}
```

2/2/2007

10

Polimorfismo - Classe Circulo

```
class Circulo extends FiguraPlana {
    private float raio;

    public Circulo (float raio)
    {this.raio = raio;}

    public void setRaio (float raio)
    {this.raio = raio;}

    public float getRaio ()
    {return (this.raio);}
}
```

2/2/2007

11

Polimorfismo - Classe Circulo

```
float diametro () // Método exclusivo
{return raio * 2f;}

double area () // override
{return Math.PI * Math.pow (this.raio,2);}

double perimetro () // override
{return (this.comprimento ());}

double comprimento () // Método exclusivo
{return 2 * Math.PI * this.raio;}
```

2/2/2007

12

Polimorfismo - Classe Circulo

```

■ double area (float comprimento) // em PI
■ {float raio; // Método exclusivo
■   raio = comprimento / 2 ;
■   return (Math.pow (raio,2)); }

■ void mostrarDados () // override
■ {System.out.println ("Informacoes do "+
■   "circulo de raio " + this.raio);
■   super.mostrarDados ();
■   System.out.println ("nDiametro = " +
■     this.diametro ());
■ }

```

2/2/2007

13

Polimorfismo - Classe LerFiguraPlana

```

■ class LerFiguraPlana
■ {
■   public static char lerTipo () throws Exception
■   {char tipo;
■   do
■   {System.out.print ("Informe 'R' para Retangulo,
■     'C' para Circulo ou 'F' para Fim-> ");
■   tipo = JUtil.readChar ();
■   } while ((tipo != 'R') && (tipo != 'C') && (tipo != 'F'));
■   return (tipo);}

```

2/2/2007

14

Polimorfismo - Classe LerFiguraPlana

```

■ public static void main(String[] args) throws
■   Exception
■ { FiguraPlana fp = new FiguraPlana () ;
■   char tipo; // Retangulo ou Circulo
■   System.out.println ("Dados da Figura Plana");
■   tipo = lerTipo ();
■   while (tipo != 'F')
■   {switch ( tipo)
■   {

```

2/2/2007

15

Polimorfismo - Classe LerFiguraPlana

```

■ case 'R':
■ {fp = new Retangulo(0,0);
■   System.out.print ("Informe a base -> ");
■   ((Retangulo) fp).setLadoa (JUtil.readFloat
■     ());
■   System.out.print ("Informe a largura -> ");
■   ((Retangulo) fp).setLadob (JUtil.readFloat
■     ());
■   break;
■ }

```

2/2/2007

16

Polimorfismo - Classe LerFiguraPlana

```

■ case 'C':
■ {fp = new Circulo(0);
■   System.out.print ("Informe o raio -> ");
■   ((Circulo) fp).setRaio (JUtil.readFloat ());
■   break;
■ }
■ } // switch

```

2/2/2007

17

Polimorfismo - Classe LerFiguraPlana

```

■ fp.mostrarDados (); // polimorfismo aqui

■ if ( fp instanceof Circulo )
■ { System.out.println ("n Area em PI " +
■   "circulo com comprimento 20 PI " + ((Circulo)
■     fp).area (20));}
■ tipo = lerTipo ();
■ } // while
■ } // main
■ } // class

```

2/2/2007

18

Polimorfismo - Observações

- **((classe) objeto).mensagem** - Faz com que um objeto genérico (declarado com sendo de uma superclasse) se comporte como uma instância de uma classe específica (subclasse)
- Exemplo : **((Circulo) fp).diametro ()**
- objeto **instanceof Classe** - operador que checka qual é a classe do referido objeto (objeto **é uma instância de Classe** ?)
- Exemplo : if (fp **instanceof** Circulo) ...
- // A pergunta acima é : fp **é um círculo** ?

2/2/2007

19

Exercícios Propostos

- 1) "O polimorfismo é proporcionado pela vinculação dinâmica de mensagens a definições de métodos" Verdadeiro ou Falso ?
- 2) Como o polimorfismo e a vinculação dinâmica facilitam a extensão dos sistemas ? Deitel 405 [454] e Sebesta 421
- 3) O que é um método final ? Deitel 406 [455]
- 4) Quais as vantagens do polimorfismo para o programador ? Deitel 407 [456]

2/2/2007

20

Exercícios Propostos

- 5) "Quando uma solicitação de uma operação vai para uma subclasse, a lista de operações permissíveis dessa classe é verificada. Se a operação for encontrada na lista, ela é invocada; se não, as superclasses são examinadas para a localização da operação". Verdadeiro ou Falso ?
- 6) "O polimorfismo é importante por tornar as classes mais independentes entre si e por permitir que novas classes sejam incluídas num sistema com mínimo impacto sobre as classes já existentes". Verdadeiro ou Falso ?

2/2/2007

21

Exercícios Propostos

- 7) "O polimorfismo conduz a um sistema mais simples e mais preparado para evoluir com o tempo, de modo a atender as necessidades de mudança". Verdadeiro ou Falso ?
- 8) O que é lógica switch e quais os seus inconvenientes ? Deitel 405 [454], Sebesta 421

2/2/2007

22

Exercícios Propostos

- 9) Implemente em Java o exemplo de polimorfismo (classe Quadrilatero) da página 406 [455] do Deitel.
- 10) Implemente em Java o exemplo de polimorfismo (folha de pagamento) da página 408 [454] do Deitel. Faça também as alterações recomendadas na página 441 [505] (9.25).
- 11) Acrescente um Triângulo no exercício resolvido.

2/2/2007

23