

Programação III

Herança
Jocélio Passos
joceliopassos@bol.com.br

Herança (inheritance) - Conceitos

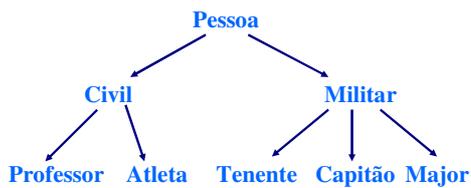
- Deitel Cap 9, páginas 385 e 395 [431 e 436]
- Sebesta Cap 11 pág 418 - 421, 446-454
- Criar novas classes (**extender**) a partir de classes já existentes sem duplicação de código
- Os atributos e comportamentos da super-classe são herdados na sub-classe
- A nova classe pode
 - Acrescentar novas funcionalidades
 - Mudar a funcionalidade redefinindo métodos com o mesmo nome (**override**) Deitel 387 [433]
- Possibilita a reutilização do código



2/2/2007

2

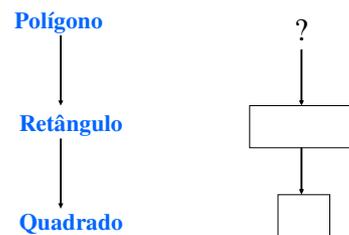
Herança : Acrescendo Funcionalidades



2/2/2007

3

Herança : Redefinindo Métodos

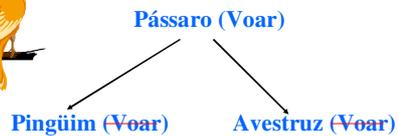


- O método **perímetro** tem implementações diferentes em cada classe

2/2/2007

4

Herança : Cancelando Métodos

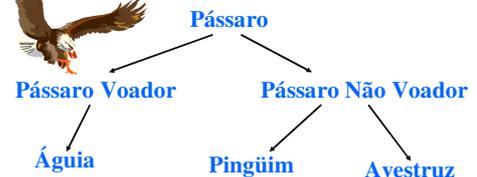


- O método **voar** é inválido na classe pinguim e avestruz
- Cancelamento é uma prática **questionável**
- Por definição todas as características de um tipo aplicam-se a seus subtipos

2/2/2007

5

Herança : Modificando Hierarquia



- Todas as operações relacionadas a voar devem ser desviadas de Pássaro para Pássaro Voador
- Porém esta alteração pode causar **sobrecarga**
- Assim se justifica o desvio do logicamente correto em nome do **desempenho**

2/2/2007

6

Herança - Exemplo com Código

- A partir da Classe Retângulo (figura plana) criar a Classe Paralelepípedo (Figura Espacial).

```

    graph TD
      Retangulo[Retângulo] --> Paralelepipedo[Paralelepípedo]
  
```

2/2/2007 7

Herança - Classe Retângulo

```

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

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

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

2/2/2007 8

Herança - Classe Retângulo

```

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

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

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

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

2/2/2007 9

Herança - Classe Retângulo

```

    void mostrarDados ()
    {System.out.println ("Retangulo com lados " +
    this.ladoa + " e " + this.ladob +
    "\nArea = " + this.area () +
    "\nperimetro = " + this.perimetro ());}
  
```

2/2/2007 10

Herança - Classe Paralelepípedo

```

class Paralelepipedo extends Retangulo {
    private float ladoc;

    public Paralelepipedo () // Sobrecarga de construtor
    {super (0,0);
    this.ladoc =0;}

    public Paralelepipedo (float ladoa, float ladob,
    float ladoc)
    {super (ladoa,ladob);
    this.ladoc = ladoc;}
}
  
```

2/2/2007 11

Herança - Classe Paralelepípedo

```

    public float getLadoc ()
    {return (this.ladoc);}

    public void setLadoc (float ladoc)
    {this.ladoc = ladoc;}

    double area () // override
    {return 2 * (this.ladoa * super.getLadob () +
    this.ladoa * this.ladoc + super.getLadob () *
    this.ladoc) ;}

    double perimetro () // override
    {return (0);}
  
```

2/2/2007 12

Herança - Classe Paralelepípedo

- double **volume ()** // area da base * altura
- {return (**super.area ()** *this.ladoc);}
- void **mostrarDados ()** // overwrite
- {System.out.println ("Paralelepipedo com base " + **super.ladoa** + ", largura " + **super.getLadob ()** + " e altura " + **this.ladoc** + "\nArea = " + **this.area ()** + "\nVolume = " + **this.volume ()**);}
- }

2/2/2007

13

Herança - Palavras chaves

- **Protected** (protegido) - Qualificador que permite que atributos e métodos de uma classe sejam visíveis também em suas classes filhas, mantendo o encapsulamento para as demais
- Deitel 386, 389 [433, 436]
- **Extends** - Define de qual classe se deseja herdar
- Deitel 386 [432]
- Formato Geral
 - Classe subClasse **extends** superClasse
- **super** - Referência a atributos ou métodos da classe pai (**super** classe) Deitel 389 [436]

2/2/2007

14

Herança - Classe LerParalelepipedo

- class LerParalelepipedo
- {public static void main(String[] args) throws Exception
- { Paralelepipedo paralelepipedo = new Paralelepipedo ();
- Paralelepipedo paralelepipedo2 = new Paralelepipedo (5,8,12);
- paralelepipedo2.**mostrarDados ()**;
- System.out.println ("Dados do Paralelepipedo");
- System.out.print ("\nInforme a base -> ");
- paralelepipedo.**setLadoa** (JUtil.readFloat ());
- while (paralelepipedo.**getLadoa** () != 0) {

2/2/2007

15

Herança - Classe LerParalelepipedo

- System.out.print ("\nInforme a largura -> ");
- paralelepipedo.**setLadob** (JUtil.readFloat ());
- System.out.print ("\nInforme a altura -> ");
- paralelepipedo.**setLadoc** (JUtil.readFloat ());
- paralelepipedo.**mostrarDados ()**;
- System.out.print ("\nInforme a base -> ");
- paralelepipedo.**setLadoa** (JUtil.readFloat ());
- }
- }
- }

2/2/2007

16

Exercícios Propostos

- 1) O que é herança simples e herança múltipla ? Quais as suas vantagens e desvantagens ? Dê 2 exemplos. [Sebesta 420 e 425](#)
- 2) O que é herança diamante ? [Sebesta 425](#)
- 3) O que é uma classe derivada ? [Sebesta 422](#)
- 4) O que são mensagens ? [Sebesta 422](#)
- 5) Há alguma desvantagem no uso da herança ? Qual ? [Sebesta 421](#)
- 6) Como funciona os qualificadores : public, private, protected ? [Sebesta 446, Deitel 388-389 \[435-436\]](#)

2/2/2007

17

Exercícios Propostos

- 7) "O Java permite que classes sejam definidas sem ter nenhum pai". Verdadeiro ou Falso ? [Sebesta 452, Deitel 388 \[434\]](#)
- 8) Quais são os três recursos característicos das linguagens OO? [Sebesta Cap 11](#)
- 9) O que é um método de sobreposição ? [Sebesta Cap 11](#)
- 10) Como o "Deitel" define herança ? [Deitel 385 \[431\]](#)
- 11) Como o Java contorna o fato de não suportar herança múltipla ? [Deitel 386 \[432\]](#)

2/2/2007

18

Exercícios Propostos

- 12) "A redefinição de método de superclasse em uma subclasse precisa ter a mesma assinatura que o método de superclasse". Verdadeiro ou Falso ? Deitel 393 [440]
- 13) Qual a diferença entre herança e composição ? Deitel 399 [447]

2/2/2007

19

Exercícios Propostos

- 14) Implemente em Java o problema abaixo :
Uma Pia é uma pia comum, contendo comprimento, largura e altura e cor. PiaLouca é uma pia de louça. PiaInox é uma pia em inox. Há objetos como VasoSanitário, Chuveiro e Banheira. Um Cômodo contém comprimento, largura e altura, cor do piso, parede e teto, e o tipo do teto. Cozinha é um Cômodo com uma pia de inox.

2/2/2007

20

Exercícios Propostos

- Banheiro é um Cômodo com uma pia de louça, vaso sanitário e chuveiro. BanheiroLuxo é um banheiro com banheira. Suite é um Cômodo com banheiro de luxo. Casa contém uma sala (Cômodo), uma Suite, Quarto das crianças, Quarto de visitas, Cozinha e Banheiro social. CasaSimples contém sala, quarto do casal, quarto das crianças, cozinha e banheiro.

2/2/2007

21