Programação II

Prof. Alcemir Rodrigues Santos

Herança

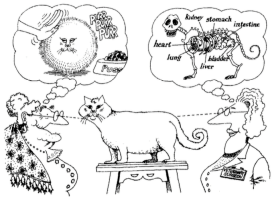
Agenda

- Princípios do paradigma OO
- Polimorfismo
  - Herança
  - Classes Abstratas
- Exercícios

Princípios do Paradigma OO

Abstração

- A representação computacional do objeto real deve se concentrar nas características que são relevantes para o problema



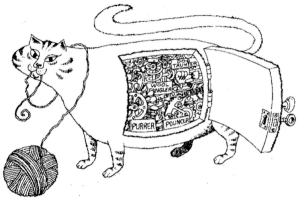
Fonte: livro "Object-Oriented Analysis and Design with Applications"

Abstração

- São criados somente os atributos e métodos necessários para o problema em mãos
- Quais seriam os atributos e métodos para o objeto Carro em cada uma das situações seguintes?
  - Sistema de uma locadora de carros
  - Sistema de uma revendedora de carros
  - Sistema de uma oficina mecânica
  - Sistema do DETRAN

Encapsulamento

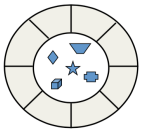
- O objeto deve esconder seus dados e os detalhes de sua implementação



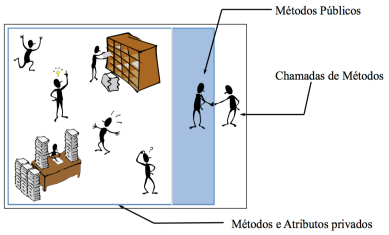
Fonte: livro "Object-Oriented Analysis and Design with Applications"

Encapsulamento

- Atributos e Métodos
  - Os métodos formam uma "cerca" em torno dos atributos
  - Os atributos não devem ser manipulados diretamente
  - Os atributos somente devem ser alterados ou consultados através dos métodos do objeto

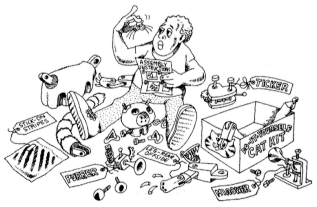


Encapsulamento



Modularidade

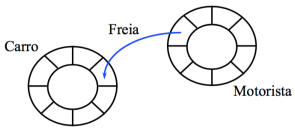
- Um sistema deve ser decomposto em um conjunto **altamente coeso** e **fracamente acoplado** de objetos



Fonte: livro "Object-Oriented Analysis and Design with Applications"

Modularidade

- Um programa OO é um conjunto de objetos que colaboram entre si para a solução de um problema
- Objetos colaboram através de chamadas de métodos uns dos outros



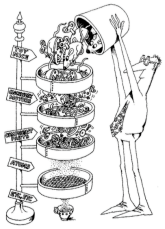
Modularidade

- Em um sistema acadêmico, há conceitos aluno, professor, disciplina, turma e inscrição. Onde colocar cada um dos métodos a seguir:
  - Exibição do histórico do aluno
  - Cálculo da média do aluno em uma turma
  - Obtenção do horário de uma aula
  - Descrição da ementa de uma disiplina
  - Cálculo do CR de um aluno

Faça Você Mesmo!

Hierarquia

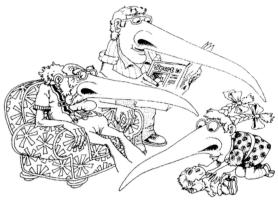
- Os objetos devem ser organizados no sistema de forma hierárquica



Fonte: livro "Object-Oriented Analysis and Design with Applications"

Hierarquia

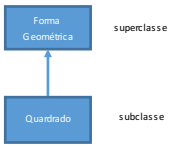
- Objetos herdam atributos e métodos dos seus ancestrais na hierarquia



Fonte: livro "Object-Oriented Analysis and Design with Applications"

Herança

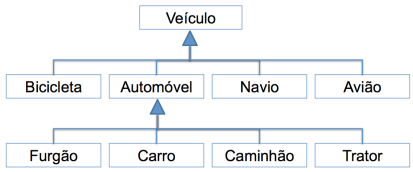
- Para viabilizar a hierarquia entre objetos, as classes são organizadas em estruturas hierárquicas
  - A classe que forneceu os elementos herdados é chamada de superclasse
  - A classe herdeira é chamada de subclasse
  - A subclasse pode herdar os métodos e atributos de suas superclasses
  - A subclasse pode definir novos atributos e métodos específicos



Polimorfismo

- Uma subclasse pode redefinir (sobrescrever) um método herdado
  - Este mecanismo é chamado de polimorfismo
  - O polimorfismo se realiza através da recodificação de um ou mais métodos herdados por uma subclasse
  - Em tempo de execução, o Java saberá qual implementação deve ser usada

Exemplo de Herança



Teste da Leitura: "*subclasse* é um *superclasse*"
Ex.: Carro é um Automóvel; Trator é um Veículo; ...

Exemplo de Herança

```
public class Carro {
  private int velocidade;

  public Carro(int velocidadeInicial)
    velocidade = velocidadeInicial;
  }

  public void acelera() {
    velocidade++;
  }

  public void freia() {
    velocidade--;
  }
}
```


Exemplo de Herança

- Declaração:

```
public class CarroInteligente extends Carro {
  public CarroInteligente(int velocidadeInicial) {
    super(velocidadeInicial);
  }

  public void estaciona() {
    // código mágico para estacionar sozinho
  }
}
```

- Uso:

```
CarroInteligente tiguan = new CarroInteligente(10);
for (int i = 10; i > 0; i--) {
  tiguan.freia();
}
tiguan.estaciona();
```

Criando um Carro Inteligente

De onde veio isso?

Exemplo de Herança

- Declaração:

```
public class CarroCorrida extends Carro {
  public CarroCorrida(int velocidadeInicial) {
    super(velocidadeInicial);
  }

  public void acelera() {
    velocidade+=5;
  }
}
```

Criando um Carro de Corrida

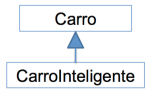
- Uso:

```
CarroCorrida f1 = new CarroCorrida(10);
f1.acelera();
```

Qual a velocidade agora?

Compatibilidade de Tipos

- Qualquer **subclasse** é compatível com a sua **superclasse**
  - Contudo, a reciproca não é verdadeira



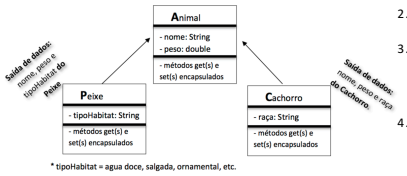
✓
```
Carro c = new CarroInteligente(20);
c.acelera();
c.freia();
```

✗
```
CarroInteligente c = new Carro(20);
c.acelera();
c.freia();
c.estaciona();
```


Herança em Java

- Uma classe só pode herdar de uma outra classe (herança simples)
- Caso não seja declarada herança, a classe herda da classe Object
  - – Ela define o método toString(), que retorna a representação em String do objeto
  - – Qualquer subclasse pode sobrescrever o método toString() para retornar o que ela deseja.
- Veja os demais métodos da classe Object em no JavaDoc

Questão 1

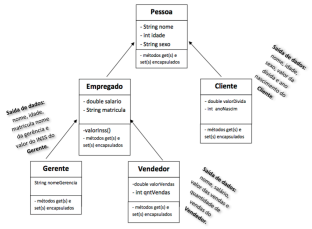


1. Crie as classes solicitadas.
2. Faça o relacionamento (herança) entre as classes.
3. Defina a saída dos dados (toString) nas classes indicadas. A classe onde tem a indicação é onde estará a saída – toString().
4. Faça a classe de teste e execute.

Mais sobre Encapsulmento

- O controle de visibilidade **protected**

Questão 2



- Crie as classes solicitadas.
- Faça o relacionamento (herança) entre as classes.
- Defina a saída dos dados (toString()) nas classes indicadas. A classe onde tem a indicação é onde estará a saída – toString().
- Faça a classe de teste e execute.
- O método valorInss() tem a fórmula (salário * 11%).

Questão 3

DICAS: Leia o texto atentamente, a medida da leitura no passo a passo. Desenhe o diagrama de classes para seu entendimento e após isso, implemente.

- Crie uma Classe Pessoa, contendo os atributos encapsulados, com seus respectivos seletores (getters) e modificadores (setters). Atributos: String nome; String endereço; String telefone;
- Considere, como subclasse da classe Pessoa (desenvolvida no item anterior) a classe Fornecedor. Considere que cada instância da classe Fornecedor tem, para além dos atributos que caracterizam a classe Pessoa, os atributos valorCredito (correspondente ao crédito máximo atribuído ao fornecedor) e valorDivida (montante da dívida para com o fornecedor).
- Implemente na classe Fornecedor, para além dos usuais métodos seletores e modificadores, um método obterSaldo() que devolve a diferença entre os valores dos atributos valorCredito e valorDivida.
- Depois de implementada a classe Fornecedor, crie um programa de teste adequado que lhe permita verificar o funcionamento dos métodos implementados na classe Fornecedor e os herdados da classe Pessoa.

- Considere, como outra subclasse da classe Pessoa, a classe Empregado. Considere que cada instância da classe Empregado tem, para além dos atributos que caracterizam a classe Pessoa, os atributos codigoSetor (inteiro), salarioBase (vencimento base) e imposto (porcentagem retida dos impostos).
- Implemente a classe Empregado com métodos seletores e modificadores e um método calcularSalario.
  - * Escreva um programa de teste adequado para a classe Empregado.
- Implemente a classe Administrador como subclasse da classe Empregado. Um determinado administrador tem como atributos, para além dos atributos da classe Pessoa e da classe Empregado, o atributo ajudaDeCusto (ajudas referentes a viagens, estadias, …). Note que deverá redefinir na classe Administrador o método herdado calcularSalario (o salário de um administrador é equivalente ao salário de um empregado usual acrescido das ajuda de custo).
  - *Escreva um programa de teste adequado para esta classe.

- Implemente a classe Operario como subclasse da classe Empregado. Um determinado operário tem como atributos, para além dos atributos da classe Pessoa e da classe Empregado, o atributo valorProducao (que corresponde ao valor monetário dos artigos efetivamente produzidos pelo operário) e comissao (que corresponde à porcentagem do valorProducao que será adicionado ao vencimento base do operário). Note que deverá redefinir nesta subclasse o método herdado calcularSalario (o salário de um operário é equivalente ao salário de um empregado usual acrescido da referida comissão).
  - *Escreva um programa de teste adequado para esta classe.

- Implemente a classe Vendedor como subclasse da classe Empregado. Um determinado vendedor tem como atributos, para além dos atributos da classe Pessoa e da classe Empregado, o atributo valorVendas (correspondente ao valor monetário dos artigos vendidos) e o atributo comissao (porcentagem do valorVendas que será adicionado ao vencimento base do Vendedor). Note que deverá redefinir nesta subclasse o método herdado calcularSalario (o salário de um vendedor é equivalente ao salário de um empregado usual acrescido da referida comissão).
  - *Escreva um programa de teste adequado para esta classe.

Questão 4

- Crie você mesmo um diagrama de classes com herança que não tenha como base nenhum dos nossos exercícios (da lista ou do caderno), e implemente no NetBeans – o objetivo desse exercício é você mesmo criar outro diagrama e implementar sem copiar dos nossos exemplos e exercícios.