Análise e Projeto Orientados a Objetos

Fundamentos e conceitos da orientação a objetos

Diretoria Acadêmica de Gestão e Tecnologia da Informação

Curso Técnico em Informática para a Internet

Introdução

- A Orientação a Objetos é um paradigma de desenvolvimento que procura diminuir a distância entre o problema do mundo real e o modelo abstrato de software construído.
- Métodos orientados a objetos estruturam os sistemas a partir dos objetos que existem no domínio do problema.
- A OO dispõe de uma série de recursos e conceitos que permitem melhor gerenciar a complexidade inerente ao domínio do problema.

Introdução

Conceito do domínio

Veículo

placa
nº do chassi

Modelo conceitual (abstração)

solução computacional e abstração do domínio real

Implementação

```
class Veiculo{
  private String placa;
  private String chassi;
  public void vender(){...}
}
```

modelo::Veículo

- placa : string
- chassi : string

Projeto OO

Conceitos

- Objeto:
  - O mundo real é povoado por objetos que interagem entre si, onde cada um deles desempenha um papel específico. Objetos podem ser coisas concretas ou abstratas, tais como um carro ou uma reserva de passagem aérea.
  - Do ponto de vista da modelagem de sistemas, um objeto é uma entidade que incorpora uma abstração relevante no contexto de uma aplicação. Um objeto possui um estado, exibe um comportamento bem definido e tem identidade única.
  - Do ponto de vista de implementação, um objeto é uma elemento de software que combina estrutura de dados e comportamento funcional.

Conceitos

- Estado:
  - O estado de um objeto compreende o conjunto de suas propriedades associadas a seus valores correntes. Propriedades de objetos são geralmente referenciadas como atributos e, portanto, o estado de um objeto diz respeito aos seus atributos e aos valores a eles associados.

Conceitos

- Comportamento:
  - Um objeto suporta um conjunto de serviços ou operações que outros objetos, ditos clientes, podem requisitar.
  - Operações são usadas para produção de informações ou para alteração e/ou consulta do estado do objeto.
  - A comunicação entre objetos dá-se por meio de troca de mensagens. Uma mensagem consiste do nome de uma operação e os argumentos requeridos (chamada de método).
  - Assim, o comportamento de um objeto representa como este objeto reage às mensagens a ele enviadas. Resumindo, o conjunto de mensagens a que um objeto pode responder representa o seu comportamento.

Conceitos

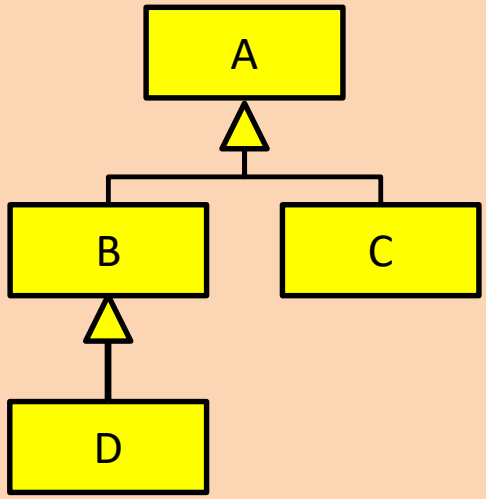
- Classes e instâncias:
  - Uma classe descreve um conjunto de objetos com as mesmas propriedades, o mesmo comportamento, os mesmos relacionamentos e a mesma semântica. Objetos que se comportam da maneira especificada pela classe são ditos instâncias dessa classe.
  - Enquanto um objeto individual é uma entidade real que executa algum papel no sistema como um todo, uma classe captura a estrutura e comportamento comum a todos os objetos que ela descreve.
  - Objetos de mesma classe partilham a mesma estrutura e o mesmo comportamento. Ainda assim, cada instância possui sua individualidade.

Conceitos

- Classes e métodos abstratos:
  - Classes abstratas não possuem instâncias. São basicamente criadas para atuar como superclasses.
  - São utilizadas para descrever um comportamento ou conjunto de características comuns a uma família/hierarquia de classes.
  - É comum que classes abstratas definam métodos abstratos (sem implementação). Estes devem ser implementados pelas suas subclasses concretas. Tais métodos definem uma espécie de contrato entre a classe abstrata e suas subclasses.

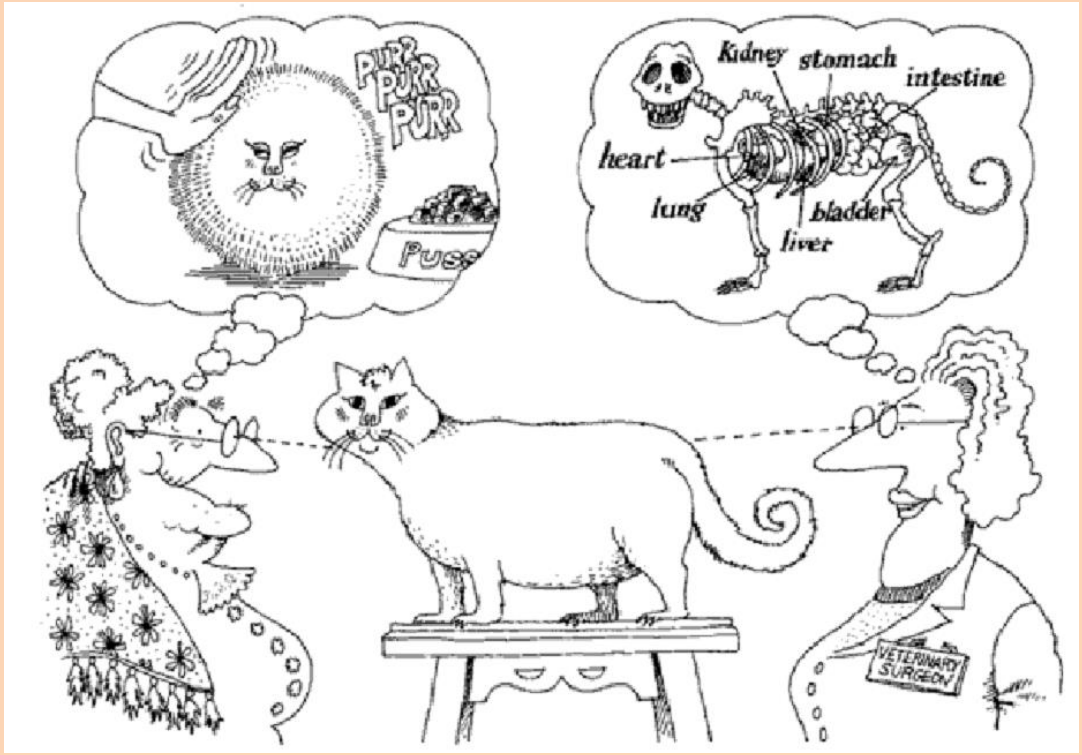
Conceitos

- Polimorfismo
  - Um objeto pode ser “enxergado” de várias formas (tipos).
  - Considere que **A** é super classe de **B** e de **C**, e que **D** é subclasse de **B**.
    - Válido: **A a = new B()**
    - Válido: **A a = new C()**
    - Válido: **A a = new D()**
    - Válido: **B b = new D()**
    - Inválido: **B b = new A()**
    - Inválido: **C c = new D()**
    - Inválido: **B b = new C()**



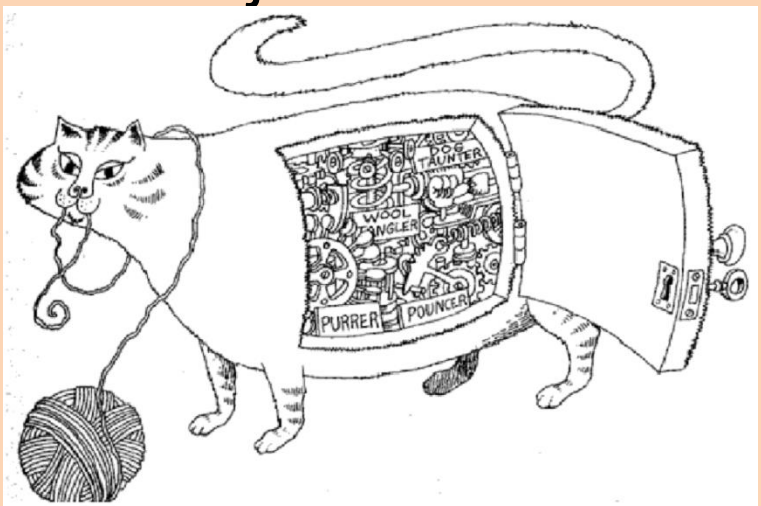
Princípios para administração da complexidade

- **Abstração**: foco nas características relevantes para um determinado observador ou analista.
  - Classes são abstrações do domínio.



Princípios para administração da complexidade

- **Encapsulamento**: ocultação dos detalhes internos de um objeto. A interação é feita com base em uma interface simples.
  - Em código o encapsulamento é tipicamente realizado ao se utilizar visibilidade restritas nos membros de um objeto.



Princípios para administração da complexidade

- **Encapsulamento**
  - Pior: **Controlador** conhece a estrutura interna de **CarrinhoDeCompras**.

```
class Item { ... }
class CarrinhoDeCompras{
 public List<Item> itens = new List<Item>();
}
class Controlador{
 private CarrinhoDeCompras carrinho;
 public void AdicionarItem(Item item){
  carrinho.itens.Add(item);
 }
}
```

Princípios para administração da complexidade

- **Encapsulamento**
  - Melhor:

```
class Item { ... }
class CarrinhoDeCompras {
 private List<Item> itens = new List<Item>();
 public void Adicionar(Item i) {
  itens.Add(i);
 }
}
class Controlador {
 private CarrinhoDeCompras carrinho;
 public void AdicionarItem(Item item) {
  carrinho.Adicionar(item);
 }
}
```

Princípios para administração da complexidade

- **Modularidade**: forma de agrupar elementos relacionados. Possibilita um melhor gerenciamento das unidades de software disponíveis.
  - Em C# o uso de namespaces para agrupar classes relacionadas é uma forma de modularidade.
  - Outra forma de modularidade bastante utilizada é a separação em camadas.

Referências

- BOOCH, Grady. **Object-Oriented Analysis and Design with Applications**, 2ª ed, Benjamin/Cummings Publishing Company, Inc, 1994.
- FALBO, Ricardo de Almeida. **Análise de Sistemas: Notas de Aula**. 2002.

Informações bibliográficas

- Autor: Alexandre Gomes de Lima
- Data de atualização: agosto de 2018
- Licença de uso: Creative Commons BY-SA (Atribuição-CompartilhaIgual)