OO e UML

Rosângela Penteado

DC – UFSCar
rosangela@dc.ufscar.br

Roteiro

- Características de OO
- Diagrama de Classe
- Diagrama de Seqüência
- Diagrama de Estado
- Roteiro para elaboração do sistema em desenvolvimento
- Informações das próximas atividades

Orientação a Objetos

- Definição:
  - Uma nova maneira de pensar os problemas utilizando modelos organizados a partir de conceitos do mundo real. O componente é o "objeto" que combina estrutura e comportamento em uma única entidade.
- Orientação a Objetos é uma metodologia de programação, assim como a procedimental, orientada a eventos, etc.
- Tudo é objeto

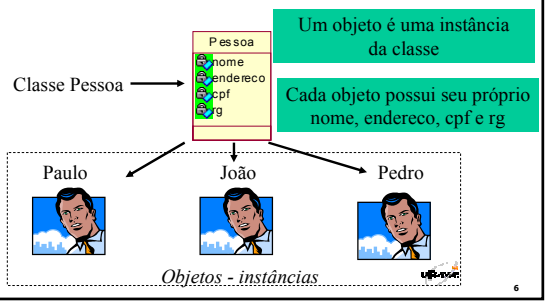
Conceitos OO - Abstração

- Capacidade de ignorar aspectos de um problema não relevantes para o entendimento/resolução desse;
- Sistema Caixa Preta
- A abstração pode ser:
  - Procedimento → operações que são decompostas em várias outras
  - Objeto.
    - Atributos
    - serviços

Conceitos da OO → Objeto

- Um objeto é uma ocorrência específica de uma **classe** e é similar a uma entidade/tabela no modelo relacional;
- Por exemplo: "Maria" é um objeto da classe "Pessoa". Uma classe define os dados e o comportamento de seus objetos.
- Valter é uma instância da classe "Pessoa"
- Uma classe pode definir o comportamento de vários objetos.

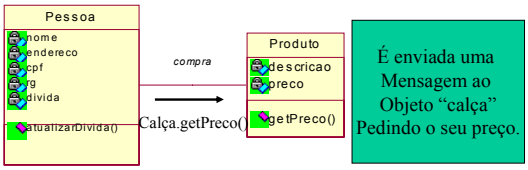
Exemplo: Classes e Objetos



Conceitos → Mensagem

- Objetos se comunicam por meio de mensagens
- Um mensagem é um sinal enviado de um objeto a outro requisitando um serviço através da execução de uma operação
- Essa operação é executada dentro da classe com base nos dados de seus alcance na hierarquia de classes e uma mensagem resultante é retornada ao solicitante

Exemplo→ Mensagem

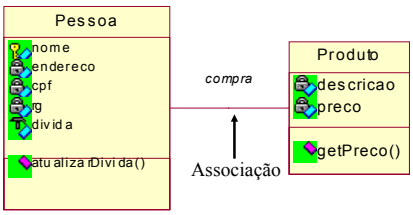


Suponha que em determinado momento da execução do programa um objeto "João" necessite atualizar seu atributo "divida". Para isso há necessidade de saber o preço do produto que o "João" comprou. Sendo assim, o método getPreco() da classe Produto deve ser invocado (mensagem) para obter o preço do produto. Após obter o preço, outras computações são executadas a fim de atualizar a dívida do "João".

Conceitos OO - Associação

- Associação é um relacionamento estrutural que ocorre entre classes;
- Esse relacionamento existe porque um objeto necessita de outros para cumprir certas responsabilidades;
- Por exemplo, quando um cliente compra um produto, tem-se uma associação entre o objeto cliente e o objeto produto

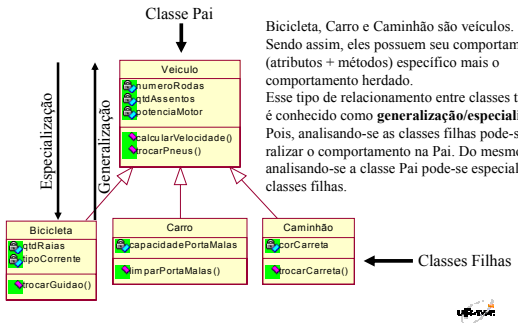
Associação - exemplo



Conceitos OO - herança

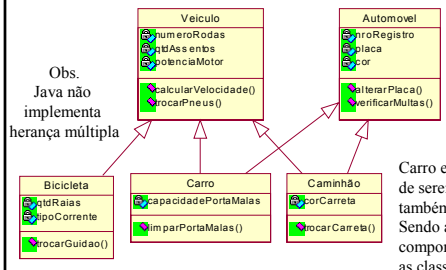
- Esse conceitos refere-se ao fato de que uma classe pode herdar o comportamento de outra.
- Geralmente identifica-se uma herança quando diz-se a palavra "é um"
- Por exemplo:
  - Bicicleta é um veículo
  - Carro é um veículo
  - Caminhão é um veículo

Exemplo herança



Bicicleta, Carro e Caminhão são veículos. Sendo assim, eles possuem seu comportamento (atributos + métodos) específico mais o comportamento herdado.
Esse tipo de relacionamento entre classes também é conhecido como **generalização/especialização**. Pois, analisando-se as classes filhas pode-se generalizar o comportamento na Pai. Do mesmo modo analisando-se a classe Pai pode-se especializar classes filhas.

Exemplo → herança múltipla



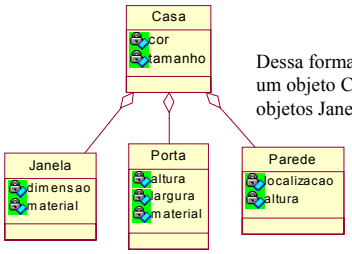
Obs.
Java não implementa herança múltipla

Carro e Caminhão, além de serem Veículos também são automóveis. Sendo assim, herdam o comportamento de ambas as classes

Conceitos OO → Todo-Parte (Agregação)

- Esse conceitos permite a construção de uma classe agregada a partir de outras classes componentes.
- Usa-se dizer que um objeto da classe Agregada (Todo) tem objetos da classe componente (Parte)
- Por exemplo: Pode-se imaginar esse tipo de relacionamento como uma casa, que é composta por portas, janelas, paredes, etc.
- A casa é a classe Todo e os constituintes as Partes.
- A pergunta a ser feita para identificar um relacionamento de agregação é: "é parte de ?"

Exemplo Agregação

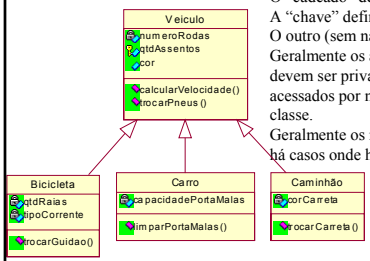


Dessa forma representa-se que um objeto Casa é composto pelos objetos Janela, Porta e Parede

Conceitos OO - Encapsulamento

- Com o advento da Internet e a exposição de sistemas nessa grande rede, a segurança tornou-se algo fundamental.
- Esse conceito está relacionado a proteger os dados da classe
- Há três tipos de encapsulamento
  - Private → só pode ser acessado por métodos da própria classe
  - Protected → pode ser acessado por métodos da própria classe e de classes herdeiras
  - Public → pode ser acessado por qualquer classe

Exemplo → encapsulamento

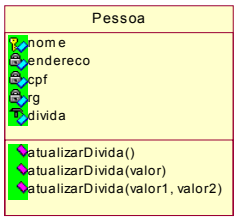


O "cadeado" define como privado (nroRodas).
A "chave" define como protegido (qtdAssentos).
O outro (sem nada) define como público (cor).
Geralmente os atributos de uma classe devem ser privados para que só possam ser acessados por meio de métodos da própria classe.
Geralmente os métodos são públicos, porém, há casos onde há necessidade de protegê-los.

Conceitos OO → Polimorfismo

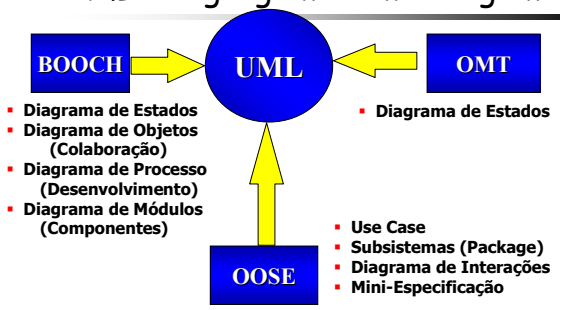
- Polimorfismo refere-se a capacidade de uma mesma operação realizar funções diferentes dependendo do objeto que a chama e dos parâmetros que lhes são passados.
- Por exemplo, pode-se ter em uma classe uma operação denominada "calcularDivida()". Caso essa operação seja invocada sem parâmetros ela realizará algo, caso seja invocada passando um determinado parâmetro realizará algo diferente.

Exemplo – Polimorfismo



Três métodos com o mesmo nome, porém, são diferenciados devido a quantidade de parâmetros passados

UML – linguagem de modelagem



- **Diagrama de Estados**
- **Diagrama de Objetos (Colaboração)**
- **Diagrama de Processo (Desenvolvimento)**
- **Diagrama de Módulos (Componentes)**

- **Diagrama de Estados**

- **Use Case**
- **Subsistemas (Package)**
- **Diagrama de Interações**
- **Mini-Especificação**

UML – Unified Modeling Language

- É uma linguagem para especificação, construção, visualização e documentação de sistemas de software;
- É a união da sintaxe gráfica de vários métodos, com vários símbolos removidos e vários adicionados;

Classes

- Classe é uma descrição de um conjunto de objetos com os mesmos atributos, relacionamentos, operações e semântica (informação transmitida do e para o sistema).
  - Um objeto é uma entidade conceitual ou física do mundo real que fornece um entendimento do mundo real e assim forma a base para uma solução de software.
- Classes = "Cliente", "Banco", "Conta"
- Toda classe deve ter um nome que a distinga das outras classes, que pode ser simples ou precedido pelo nome do pacote em que a classe está contida.

Diagramas Conceituais

Diagramas Conceituais

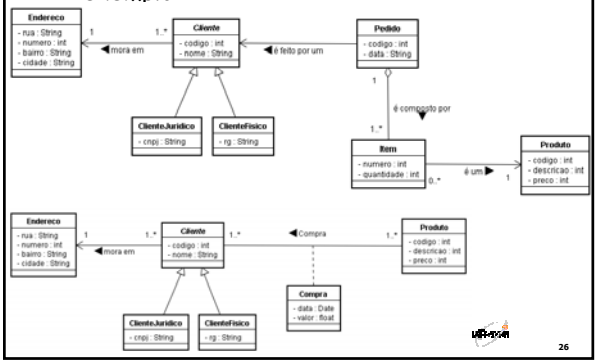
- São montados com os diagramas de classe da UML
- Cada classe representa um "conceito" do domínio
  - Nome dos conceitos no singular
- Não é o objetivo descrever detalhes de projeto
  - Tipos, métodos
  - Relacionamentos de dependência, interfaces
- O que "pode" estar presente nesses diagramas:
  - Todos os conceitos envolvidos nos casos de uso da iteração atual
  - Todos os outros tipos de relacionamentos
  - Nomes nos relacionamentos
  - Multiplicidades
  - Navegabilidade nos relacionamentos
  - Classes de associação

Diagramas Conceituais

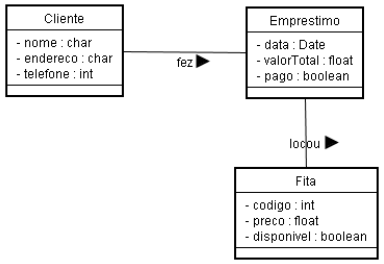
- Conceitos importantes
  - Classes
    - Atributos
  - Relacionamentos
    - Generalização/especialização
    - Associação
    - Agregação
    - Dependência
  - Nomes nos relacionamentos
  - multiplicidades

Diagramas Conceituais

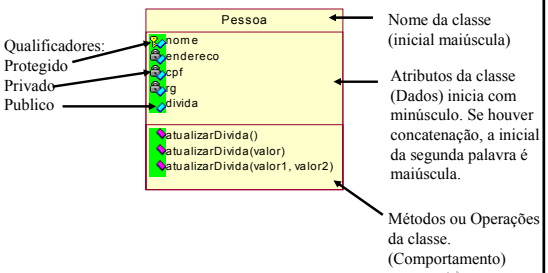
exemplo



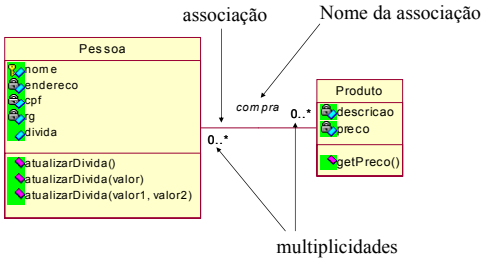
Exemplo de Diagrama de Classe



UML -- Classe



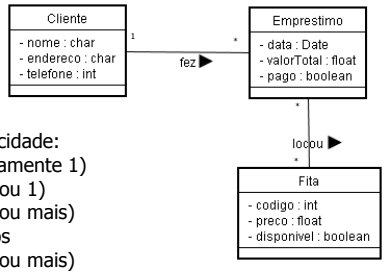
Relacionamentos - Associação



Como encontrar associações?

- A informação representada por um conceito está completa?
  - Se não, deve-se criar uma associação entre esse conceito e outro(s) conceito(s) de forma a complementar a informação necessária para que o conceito faça sentido.
- Não se deve colocar no modelo conceitual os atributos que representam chaves estrangeiras como acontece com as tabelas de banco de dados relacional.

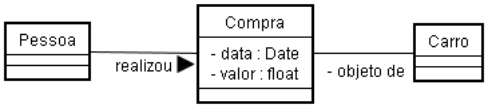
Associação



Multiplicidade:
1 (exatamente 1)
0..1 (0 ou 1)
0..* (0 ou mais)
* muitos
1..* (1 ou mais)

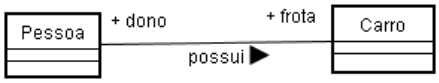
Transação não deve ser modelada como associação

Como fazer, quando necessário?



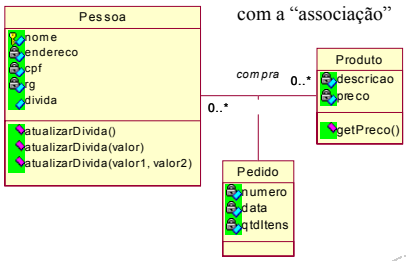
Papéis na associação

- Nomes de papéis são necessários para associação entre dois objetos de mesma classe.

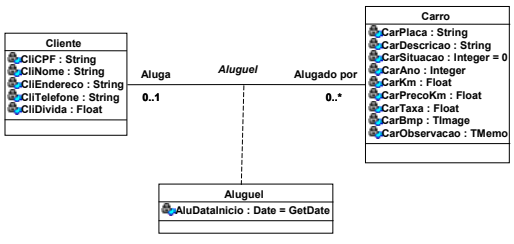


Relacionamentos – classe de associação

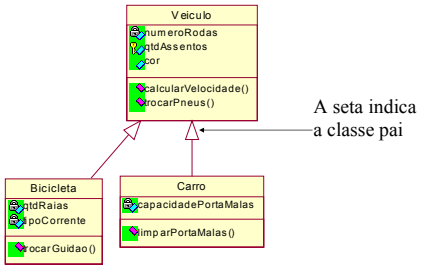
Relacionamento de uma classe com a "associação"



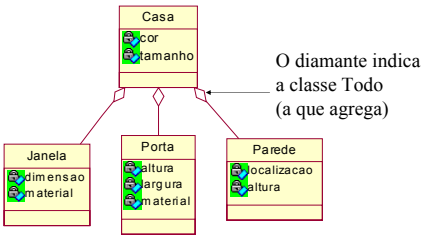
Classe associada à relação



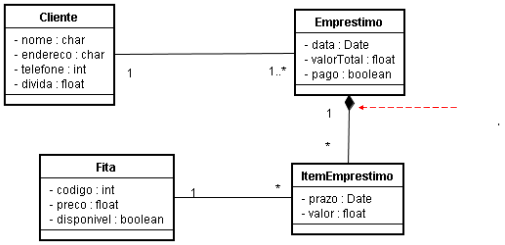
Relacionamentos - herança



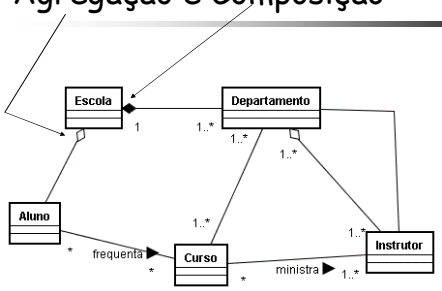
Relacionamentos - Agregação



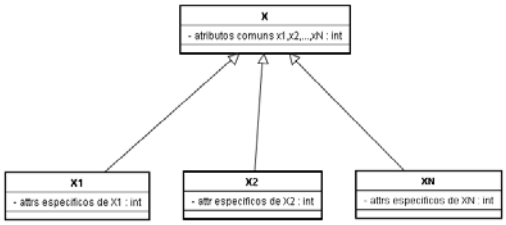
Relacionamento de Composição



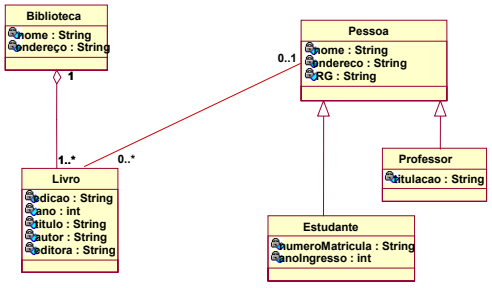
Agregação e Composição



Generalização



Exemplo: agregação/herança



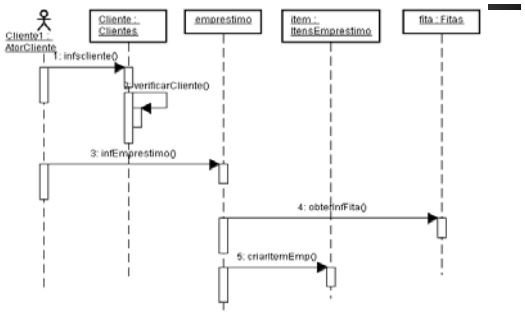
Relacionamentos entre Classes Dependência

- relacionamento de uso, no qual uma mudança na especificação de um elemento (que a seta aponta) pode alterar a especificação do elemento dependente

Diagramas de seqüência
(Diagrama de Interação)

- Modelagem do comportamento do sistema – descrição do que um sistema faz sem explicar como ele faz (seqüência de ações).
- Enfatizam a ordenação das mensagens trocadas entre os objetos e atores.
- São construídos a partir de um caso de uso.
- Representação do tempo: de cima para baixo

Exemplo de Diagrama de Seqüência



Diaarama de Seaüência- Análise

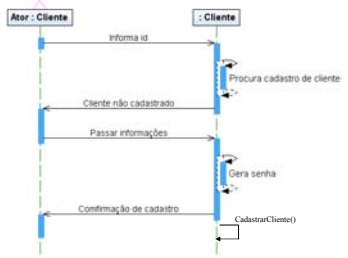


Diagrama de Seqüência de Projeto

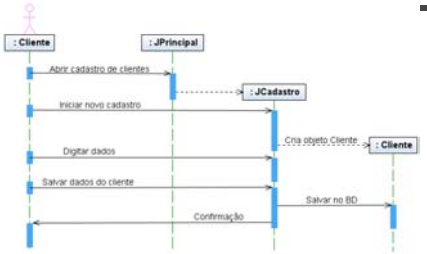


Diagrama de Estados

Diagrama de Estado

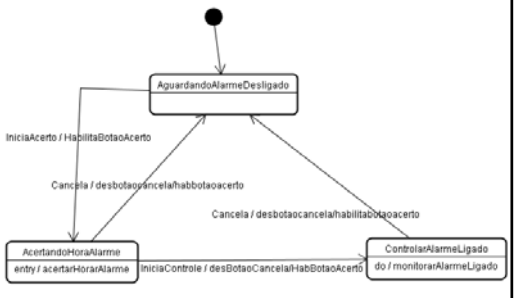
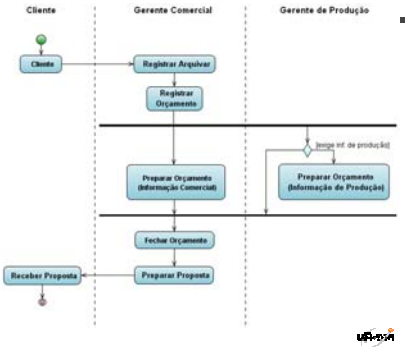


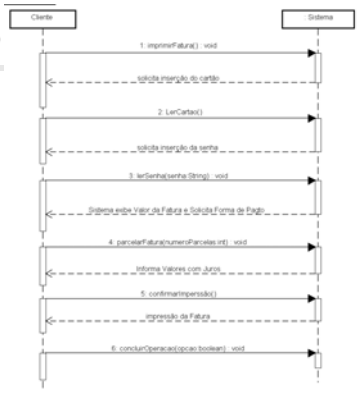
Diagrama de Atividades



Exercício

- Elabore o diagrama conceitual para o seguinte caso de uso e DSS:
  - O **cliente** informa que deseja imprimir a **fatura** de seu **cartão de crédito**.
  - O sistema exibe a tela para impressão da fatura do cartão de crédito.
  - O sistema solicita que o ator insira seu cartão.
  - O ator insere o cartão no **Caixa Eletrônico**.
  - O sistema solicita que o usuário informe a **senha do cartão**.
  - O sistema calcula o **valor da fatura** do cartão e informa ao ator.
  - O sistema solicita que o ator informe a forma de **pagamento: parcelada ou à vista**.
  - O ator informa que deseja parcelar a fatura.
  - O sistema informa a quantidade de parcelas disponíveis.
  - O ator seleciona a **quantidade de parcelas** desejadas.
  - O sistema calcula os juros inerentes a sua escolha e informa ao ator.
  - O ator confirma a operação.
  - O sistema imprime as faturas referente ao número de parcelas escolhido pelo ator.
  - O ator finaliza a operação

Exercício



Referências Bibliográficas

- Utilizando UML e Padrões – Craig Larman, Bookman Editora, 2003 (tradução)
- Guia de Consulta Rápida UML – Douglas Marcos da Silva, Novatec Editora, 2001
- UML – Booch, Rumbaugh, Jacobson, Editora Campus, 1999.
- Desenvolvendo Software Com UML 2.0 – Ernani Medeiros, Pearson, 2004