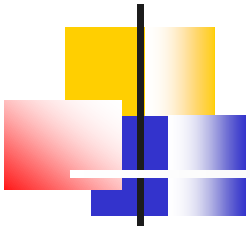
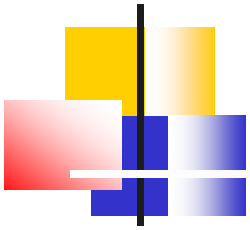


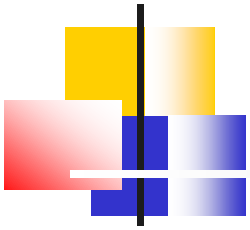
INF 213 – Estrutura de Dados
Prof. Marcus V. A. Andrade

Introdução à UML



Por que orientação a objetos?

- A orientação a objetos é um conceito de longa data:
  - **1962-67** Linguagem Simula - 1ºs conceitos OO
  - **1968** Eng. Software / Program. Estruturada
  - **~1975** Surgimento métodos OO
  - **1978** Popularização Análise Estruturada
  - **1980** SmallTalk - 1ª linguagem OO

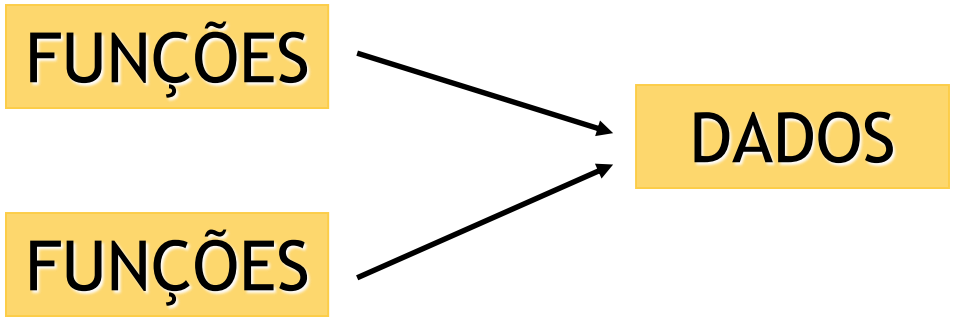


Por que orientação a objetos?

- O que ocorreu a partir de então:
  - Utilização de OO em Sistemas Operacionais
  - Evolução de linguagens, como:
    - Object Pascal, C++, Java
  - Passamos a ser usuários de OO:
    - Delphi, Visual Basic, ...

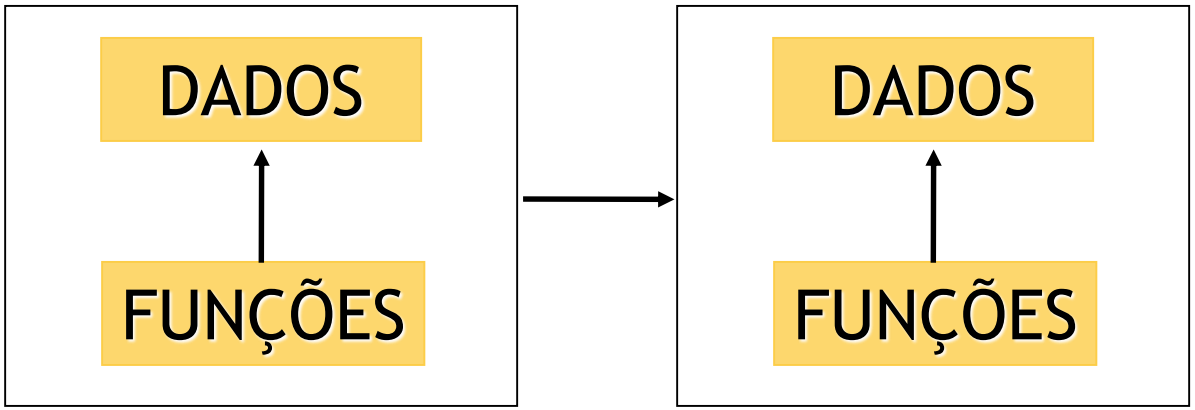
Por que orientação a objetos?

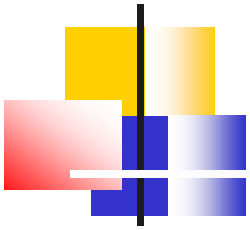
- O que buscamos ao modelar orientado a objetos ?
  - ANÁLISE ESTRUTURADA : Foco principal ⇨ Funções



Por que orientação a objetos?

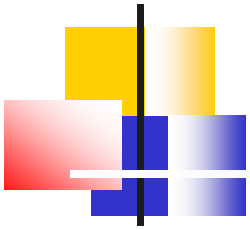
- O que buscamos ao modelar orientado a objetos ?
  - OO : Foco principal ⇨ Objetos (mundo real)





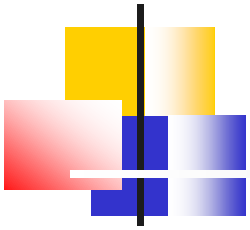
Por que orientação a objetos?

- O que buscamos ao modelar orientado a objetos ?
  - Diminuição do tempo e custo de desenvolvimento
  - Atendimento da demanda gerada pela evolução tecnológica (celular, Palm, eletrodomésticos, ...)
  - Reutilização de código, facilidade de manutenção



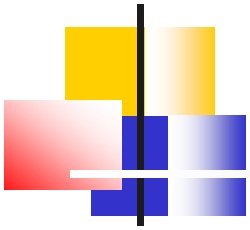
O que é UML?

- UML (*Unified Modeling Language*), é a linguagem padrão definida *OMG* (*Object Management Group*) para representar sistemas orientados a objeto
- A UML "surgiu" em meados dos anos 90 a partir da junção das 3 metodologias de modelagem orientada a objetos mais populares da época:
  - o método de Booch
  - o método OMT (Object Modelling Technique) de Jacobson
  - o método OOSE (Object-Oriented Software Engineering) de Rumbaugh.



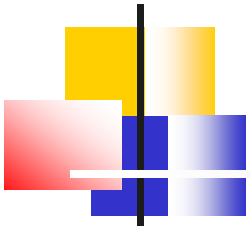
O que é UML?

- UML é uma linguagem para se descrever sistemas orientados a objeto e **não** um método para desenvolvimento de software
- Não inclui a descrição de passos que se deve seguir para se desenvolver um sistema
- Nem mesmo quais são as etapas para se modelar um sistema



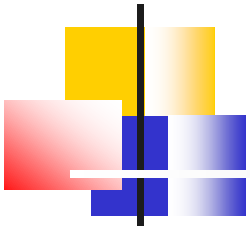
O que é UML?

- A UML se limita, exclusivamente, a representar um sistema através de um conjunto de diagramas, onde cada diagrama se refere a uma visão parcial do sistema, que em conjunto forma um todo integrado e coerente.



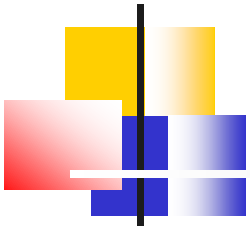
O que é UML?

- Desde a sua adoção pela OMG como linguagem padrão de modelagem, a UML tem passado por diversas evoluções
- A UML em sua versão 2.0 trouxe grandes novidades em relação sua estrutura geral principalmente com relação à abordagem de 4 camadas e à possibilidade de se desenvolver "perfis" particulares a partir da UML



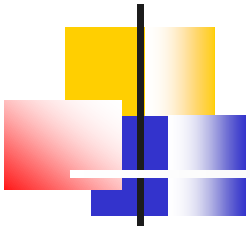
O que é UML?

- As principais camadas (visões) são:
  - Casos de uso: mostra as funcionalidades do sistema vistas pelos agentes externos
  - Lógica: mostra a visão interna do sistema descrevendo sua estrutura estática e dinâmica
  - Implementação: mostra a organização do código do sistema
  - Processo: mostra os principais componentes do sistema relacionados ao seu desempenho



O que é UML?

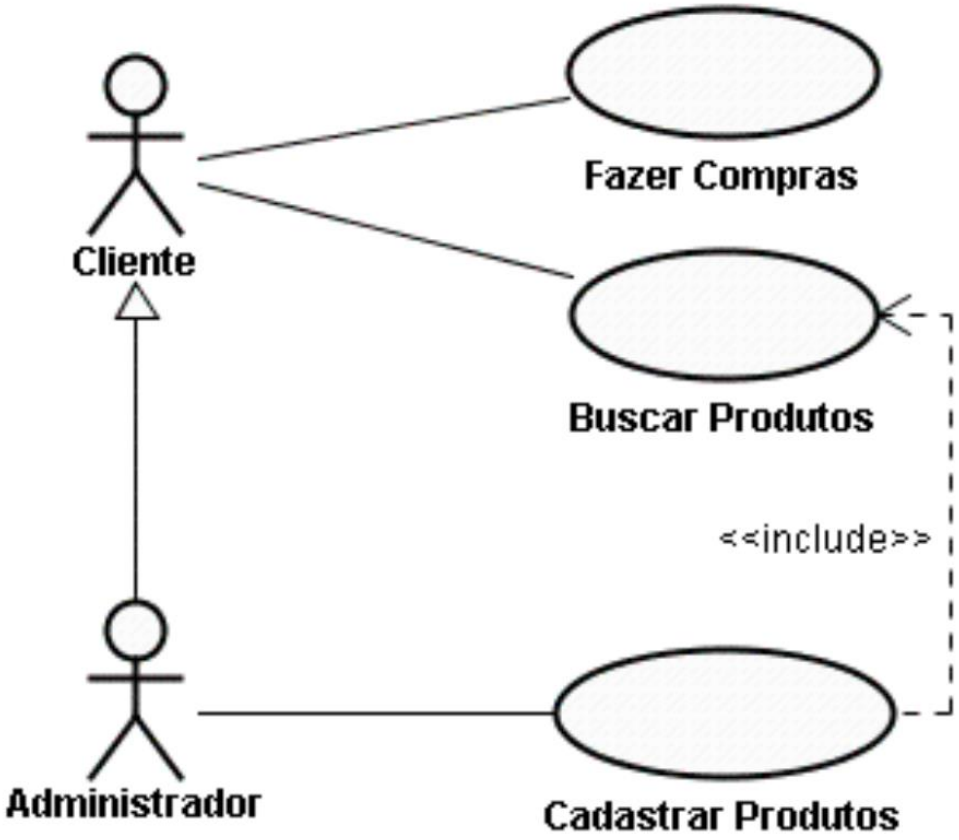
- Nos diagramas a seguir, a legenda no canto superior esquerdo se refere a:
  - **uc:** Use Case (Diagrama de Caso de Uso);
  - **class:** Class Diagram (Diagrama de Classes);
  - **object**: Object Diagram (Diagrama de Objetos);
  - **sd**: Sequence Diagram (Diagrama de Seqüência);
  - **stm**: State-Machine Diagram (Diagrama de Máquina de Estados)

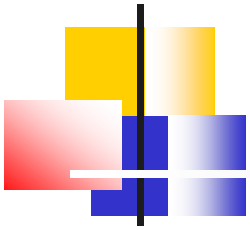


Diagramas de Casos de Uso

- Este é o diagrama mais geral e informal da UML, sendo utilizado principalmente para auxiliar no levantamento e análise dos requisitos
- Auxilia no levantamento das necessidades do usuário e na compreensão do sistema
- Serve de base para todos os outros diagramas.

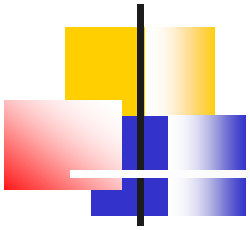
Diagramas de Casos de Uso





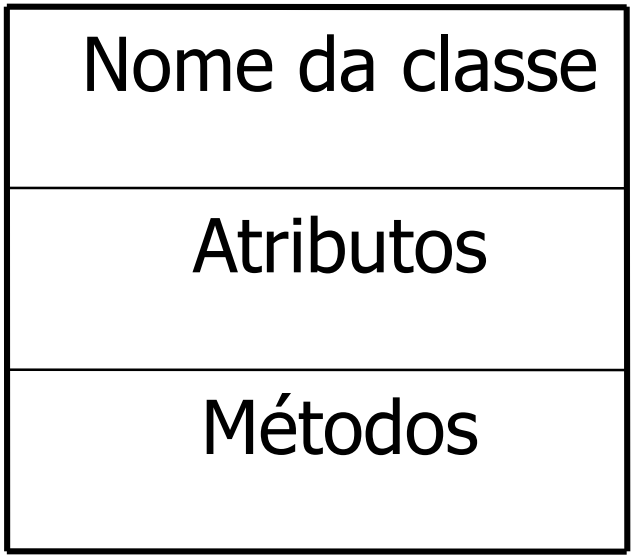
Diagramas de Classes

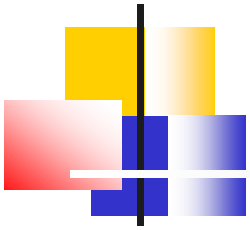
- É o diagrama mais utilizado e o mais importante da UML
- Serve de apoio para a maioria dos outros diagramas.
- Esse diagrama define a estrutura das classes utilizadas pelo sistema, determinando os atributos e métodos possuídos por cada classe, além de estabelecer como as classes se relacionam e trocam informações entre si.



Diagramas de Classes

- A representação de uma classe usa um retângulo dividido em 3 partes:

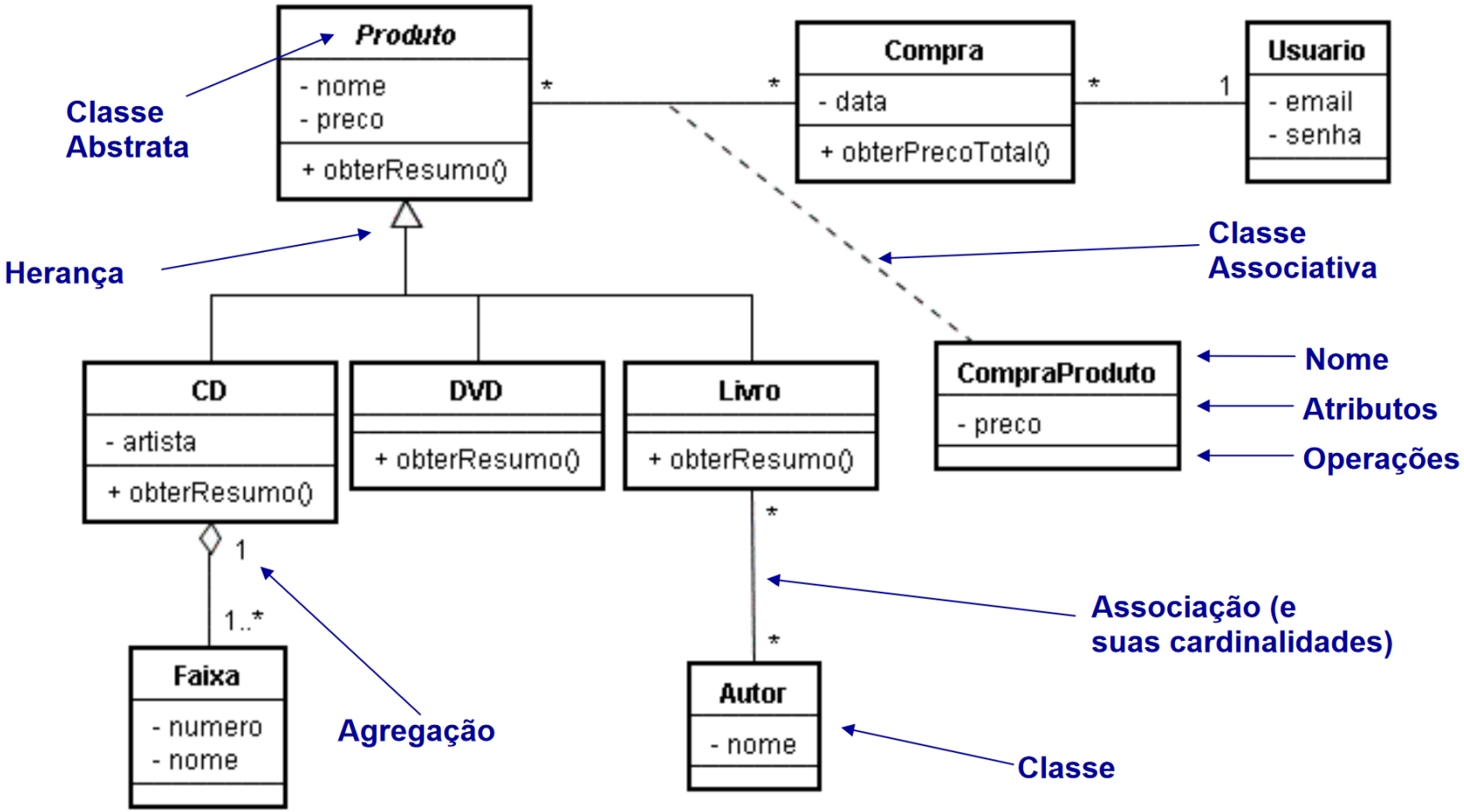


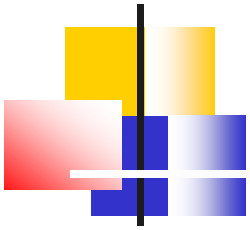


Diagramas de Classes

- O diagrama pode incluir o tipo e a *visibilidade* (público, privado ou protegido) dos atributos e métodos:
  - + indica público
  - - indica privado
  - # indica protegido

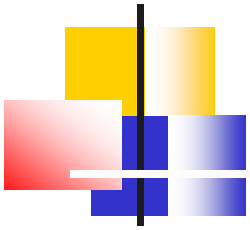
Diagramas de Classes





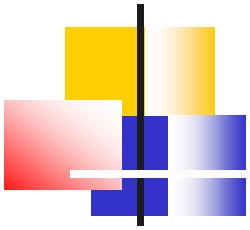
Diagramas de Classes

- As relações entre as classes podem ser associações, agregações ou heranças.
- Uma relação indica um tipo de dependência entre as classes, essa dependência pode ser forte como no caso da herança ou da agregação ou mais fraca como no caso da associação.



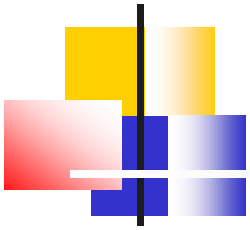
Diagramas de Classes

- Os objetos tem relações entre eles:
  - um professor *ministra* uma disciplina *para* alunos *numa* sala
  - um cliente *faz* uma reserva *de* alguns lugares *para* uma data, etc.
- Essas relações são representadas também no diagrama de classe



Diagramas de Classes

- A UML reconhece 3 tipos de relações:
  - associação
    - agregação
    - composição
  - dependência
  - generalização (ou herança).



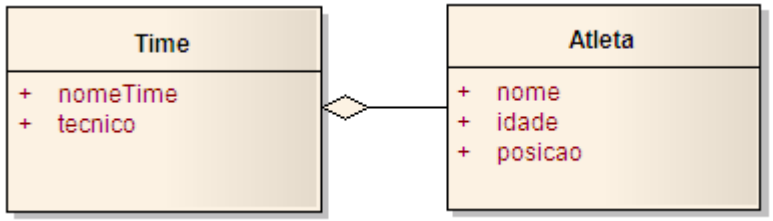
Diagramas de Classes

- **Associação** : São relacionamentos estruturais entre instâncias e especificam que objetos de uma classe estão ligados a objetos de outras classes. Podemos ter associação unária , binária , etc.

- A associação pode existir entre classes ou entre objetos:
  - Uma associação entre a classe *Professor* e a classe *Disciplina* (um professor ministra uma disciplina) significa que uma instância de Professor (um professor específico) vai ter uma associação com uma instância de Disciplina. Esta relação significa que as instâncias das classes são conectadas, seja fisicamente ou conceitualmente.

Diagramas de Classes

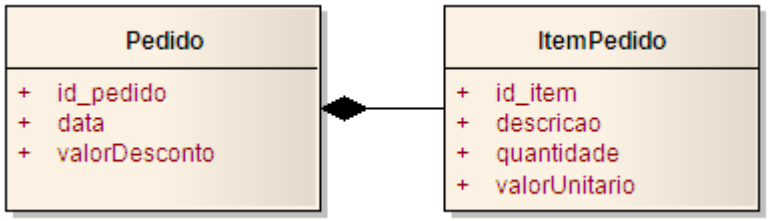
- Uma associação pode ser uma agregação ou uma composição
- Na **Agregação**, a existência do Objeto-Parte faz sentido, mesmo não existindo o Objeto-Todo. Vejamos o exemplo Time-Atleta:



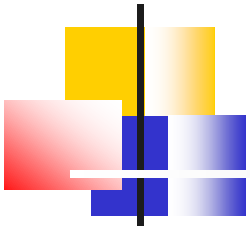
- Um time é formado por atletas, ou seja, os atletas são parte integrante de um time, mas os atletas existem independentemente de um time existir. Nesse caso, chamamos esse relacionamento de AGREGAÇÃO.

Diagramas de Classes

- Já a **Composição** é uma agregação mais forte; nela, a existência do Objeto-Parte NÃO faz sentido se o Objeto-Todo não existir. Vejamos o exemplo Pedido-ItemPedido:



- Nesse caso, um pedido é composto por um ou vários itens, mas um produto NÃO é item de um pedido se não existe pedido. Assim, chamamos esse relacionamento de COMPOSIÇÃO



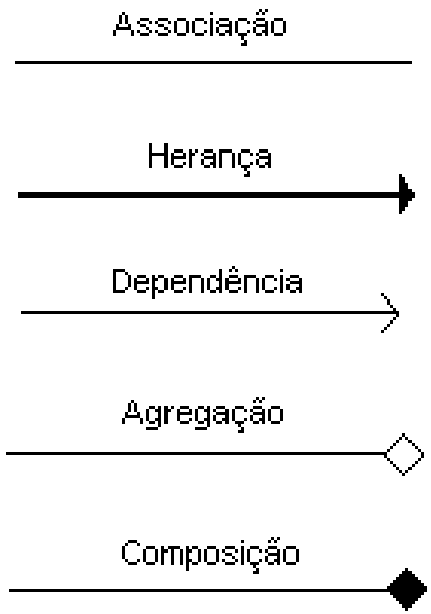
Diagramas de Classes

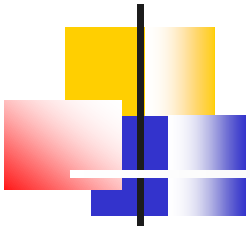
- **Dependência** - São relacionamentos de utilização no qual uma mudança na especificação de um elemento pode alterar a especificação do elemento dependente. A dependência entre classes indica que os objetos de uma classe usam serviços dos objetos de outra classe.

- **Generalização** (herança : simples ou composta) - Relacionamento entre um elemento mais geral e um mais específico. Onde o elemento mais específico herda as propriedades e métodos do elemento mais geral.

Diagramas de Classes

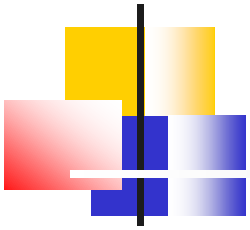
- As relações entre as classes são representadas da seguinte forma:





Diagramas de Classes

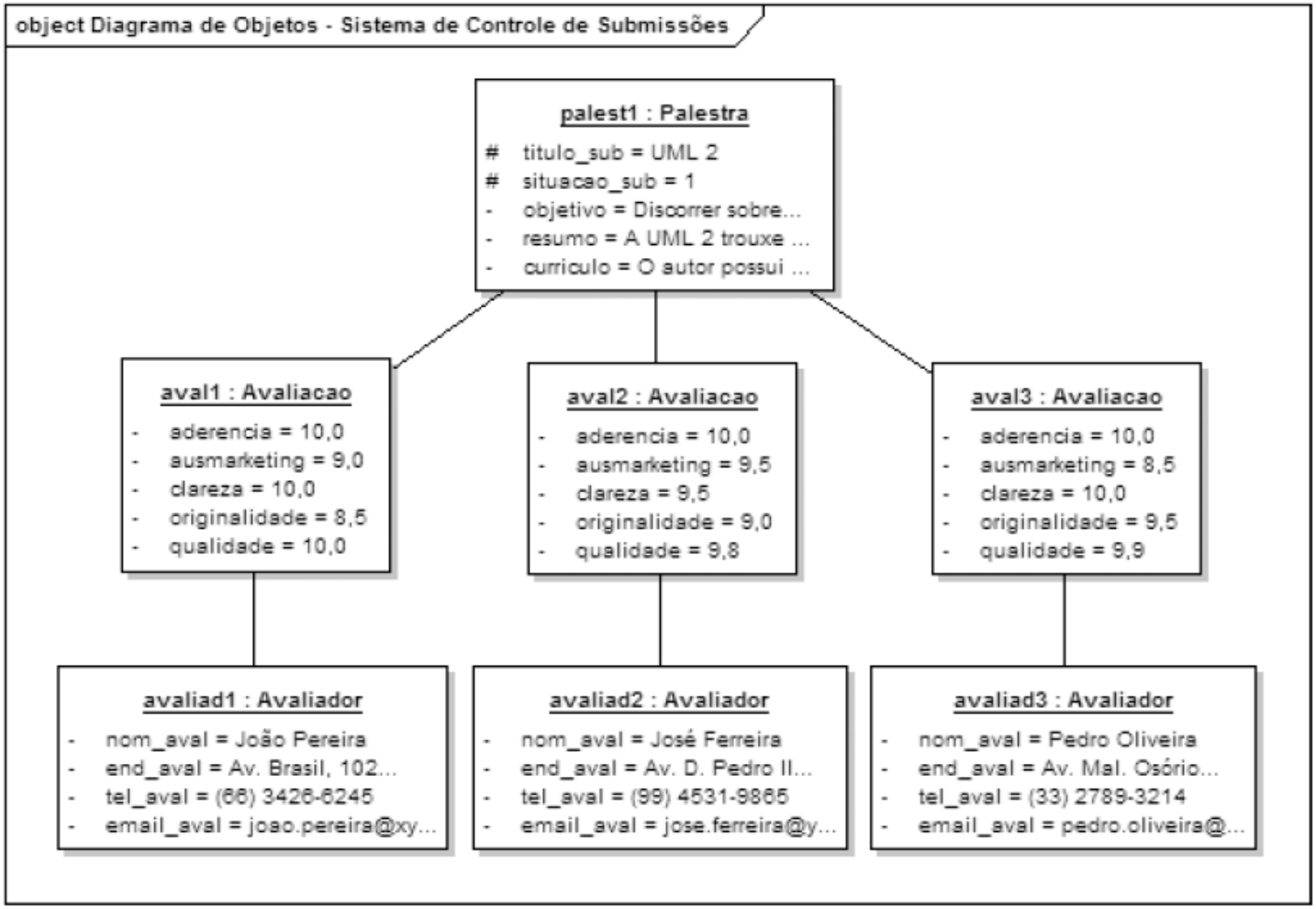
- Ao final do processo de modelagem, o diagrama de classes pode ser traduzido em uma estrutura de código que servirá de base para a implementação dos sistemas.
- No entanto, não existe no diagrama de classes informações sobre os algoritmos que serão utilizados nas operações e também sobre a dinâmica do sistema porque não há elementos sobre o processo ou a seqüência de processamento neste modelo.
- Estas informações são representadas em outros diagramas, como os diagramas de seqüência de eventos ou diagramas de estado

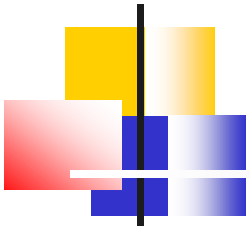


Diagramas de Objetos

- Este diagrama está amplamente associado ao Diagrama de Classes.
- Na verdade, o Diagrama de Objetos é praticamente um complemento do Diagrama de Classes que fornece uma visão dos valores armazenados pelos objetos de um Diagrama de Classes em um determinado momento da execução de um processo.

Diagramas de Objetos

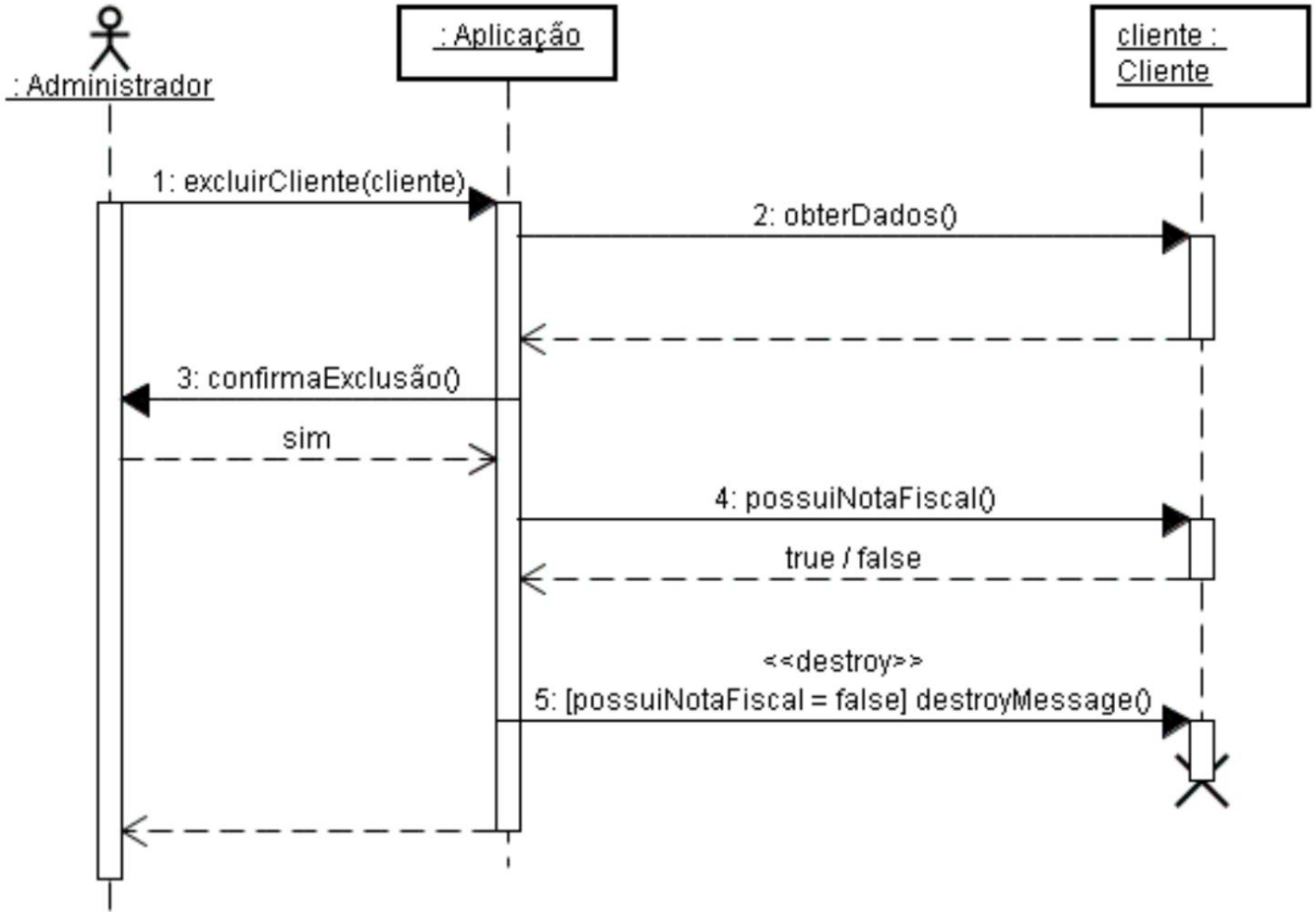




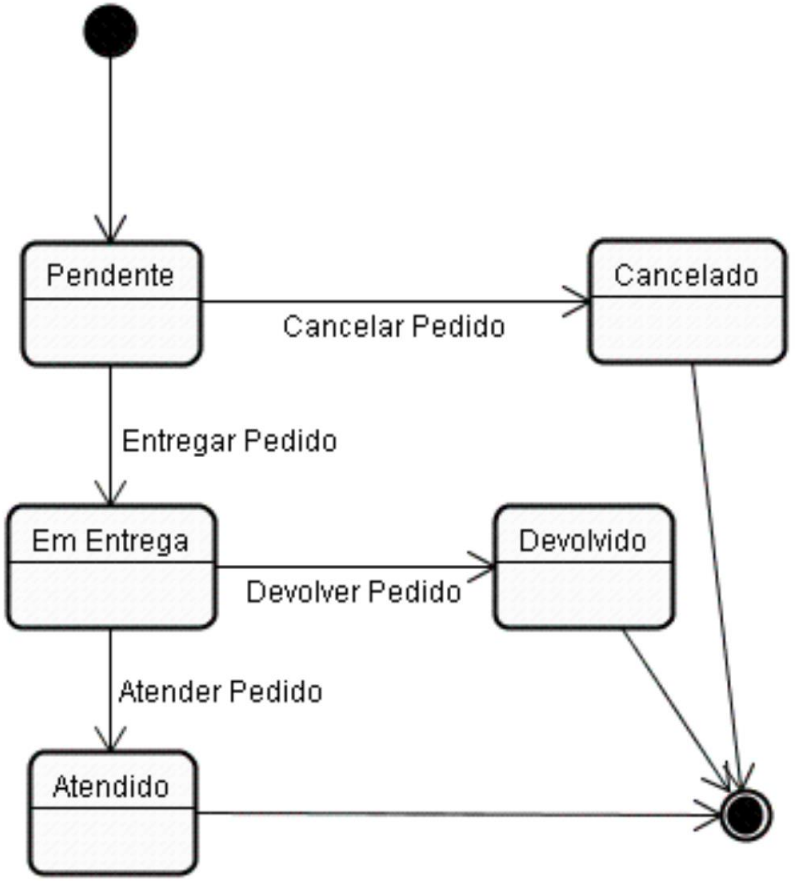
Diagramas de Seqüência

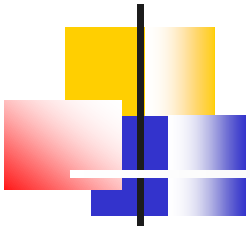
- Descreve a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos em um determinado processo.
- Em geral, baseia-se em um Caso de Uso definido pelo diagrama casos de uso e apóia-se no Diagrama de Classes para determinar os objetos das classes envolvidas em um processo, bem como os métodos disparados entre os mesmos.

Diagrama de Seqüência



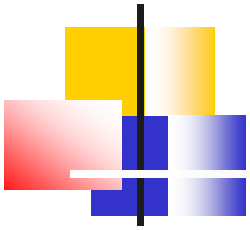
Diagramas de Máquina Estados





Outros diagramas

- O **Diagrama de Comunicação** contém informações semelhantes às apresentadas no Diagrama de Seqüência, porém com um enfoque diferente, visto que este diagrama não se preocupa com a temporalidade do processo, concentrando-se em como os objetos estão vinculados e quais mensagens trocam entre si durante o processo



Outros diagramas

- O **Diagrama de Atividade** descreve os passos a serem percorridos para a conclusão de uma atividade específica, muitas vezes representada por um método com um certo grau de complexidade, podendo, no entanto, modelar um processo completo.
- Concentra-se na representação do fluxo de controle e no fluxo de objeto de uma atividade