Universidade POSITIVO

DIAGRAMA DE CLASSES E MODELO DE DADOS

ANÁLISE E PROJETO DE SISTEMAS

Profª Dra. Cassiana Fagundes da Silva

O que é UML?

A Unified Modelling Language (UML) é uma linguagem ou notação de diagramas para especificar, visualizar e documentar modelos de 'software' orientados por objetos.

O UML não é um método de desenvolvimento, o que significa que não diz o que fazer primeiro ou o que fazer depois ou como desenhar o sistema, mas ajuda a visualizá-lo e a comunicá-lo para os outros.

O UML é controlado pelo Object Management Group (OMG) e é considerado "a norma da indústria de software" para descrever graficamente o 'software'.

http://www.uml.org/

O que é UML?

A UML é composta por muitos elementos de modelo que representam as diferentes partes de um sistema de software.

Os elementos UML são usados para criar diagramas, que representam uma determinada parte, ou um ponto de vista do sistema.

Alguns tipos de diagramas da UML::

Diagrama de Caso de Uso => mostra atores (pessoas ou outros usuários do sistema), casos de uso (os cenários onde eles usam o sistema), e seus relacionamentos

Diagrama de Classe => mostra classes e os relacionamentos entre elas

Diagrama de Sequência => mostra objetos e uma sequência de entradas e saídas entre eles.

O que é UML?

Universidade POSITIVO

Diagrama de Colaboração => mostra objetos e seus relacionamentos, colocando ênfase nos objetos que participam na troca de mensagens

Diagrama de Estado => mostra estados, mudanças de estado e eventos num objeto ou uma parte do sistema

Diagrama de Atividade => mostra atividades e as mudanças de uma atividade para outra com os eventos ocorridos em alguma parte do sistema

Diagrama de Componente => mostra os componentes de programação de alto nível (como KParts ou Java Beans)

Diagrama de Distribuição => mostra as instâncias dos componentes e seus relacionamentos.

Diagramas de Entidade-Associação => mostram os dados e as relações e as restrições entre os dados.

Diagrama de Casos de Uso

O Diagrama de *Casos de Uso* serve para auxiliar a comunicação entre os analistas e o cliente.

Um diagrama de Caso de Uso descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário.

O cliente deve ver no diagrama de Casos de Uso as principais funcionalidades de seu sistema.

O diagrama de Caso de Uso é representado por:

- atores;
- casos de uso;
- relacionamentos entre estes elementos.

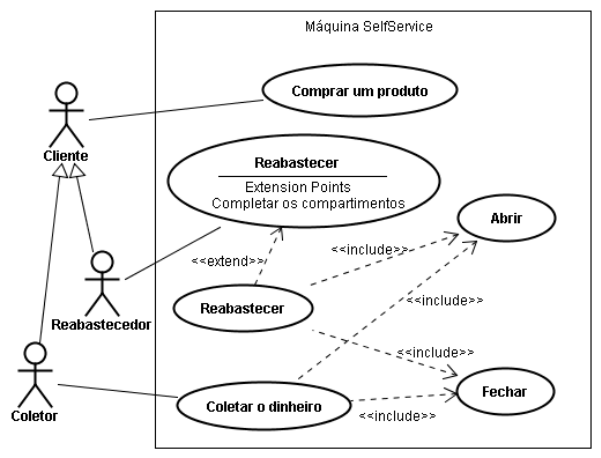
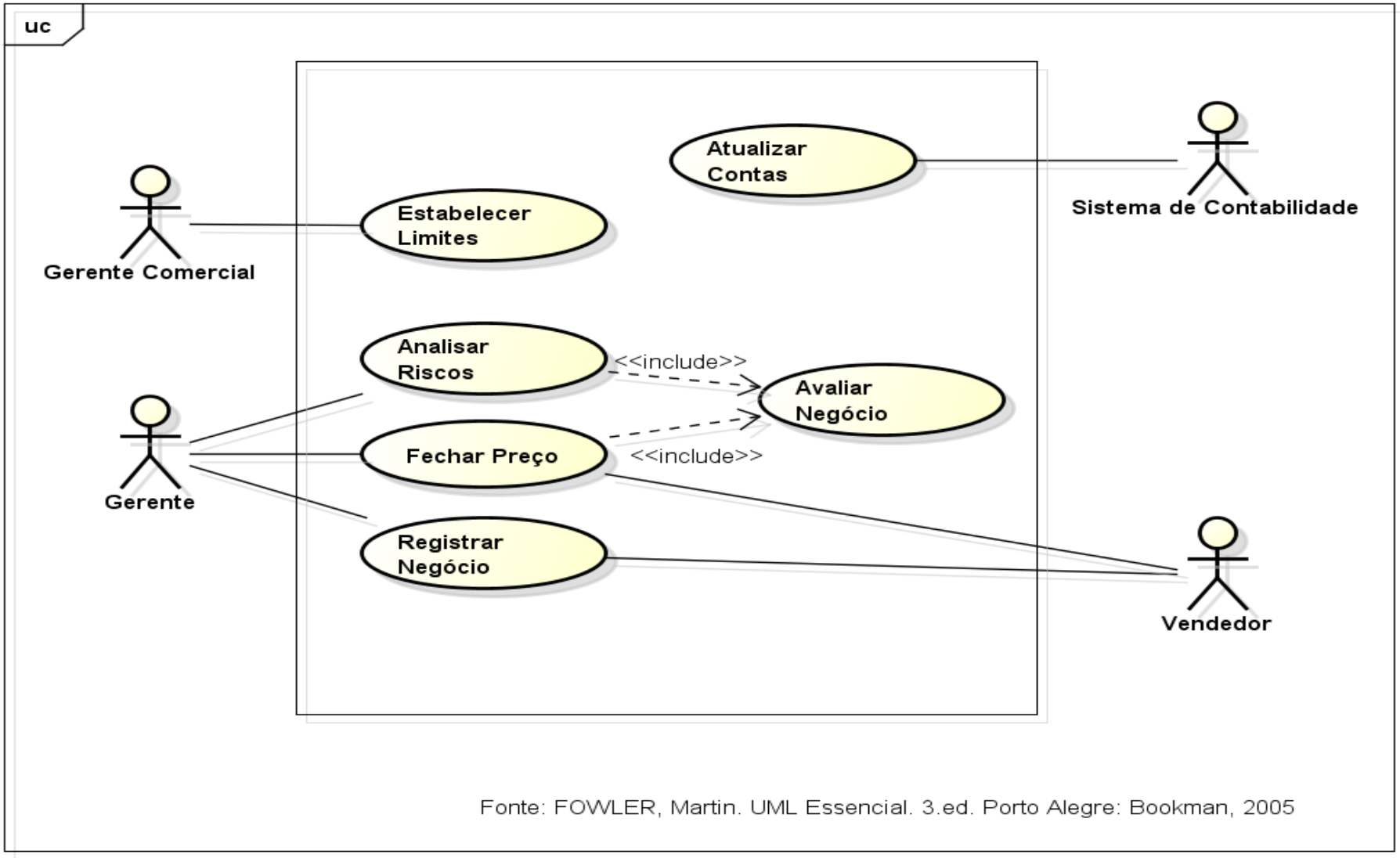
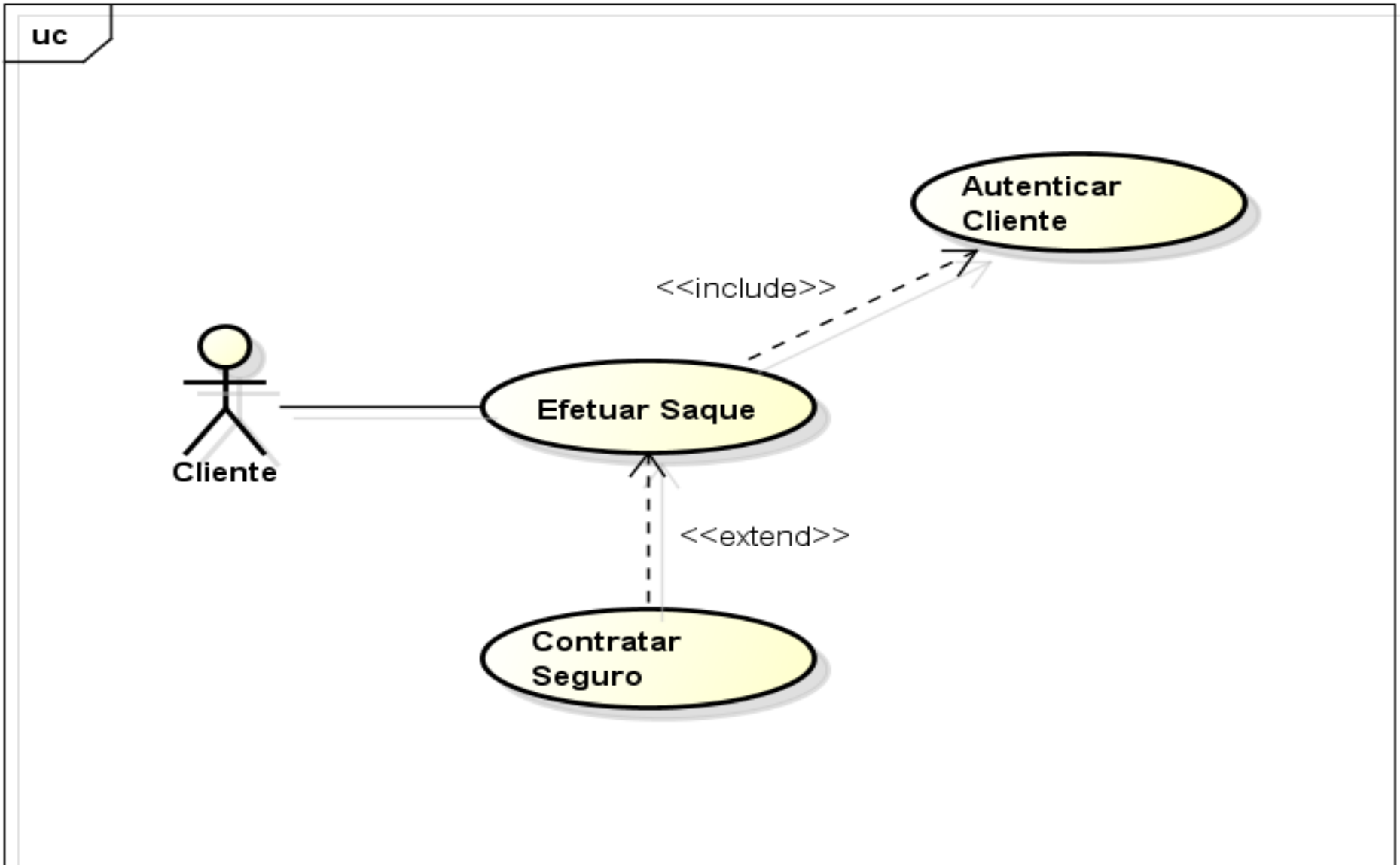


DIAGRAMA DE CASOS DE USO



Fonte: FOWLER, Martin. UML Essencial. 3.ed. Porto Alegre: Bookman, 2005

DIAGRAMA DE CASOS DE USO



ORIENTAÇÃO A OBJETOS

- Processo conceitual independente de uma linguagem de programação até as etapas finais.

- O resultado de um projeto orientado a objetos uma hierarquia de classes.

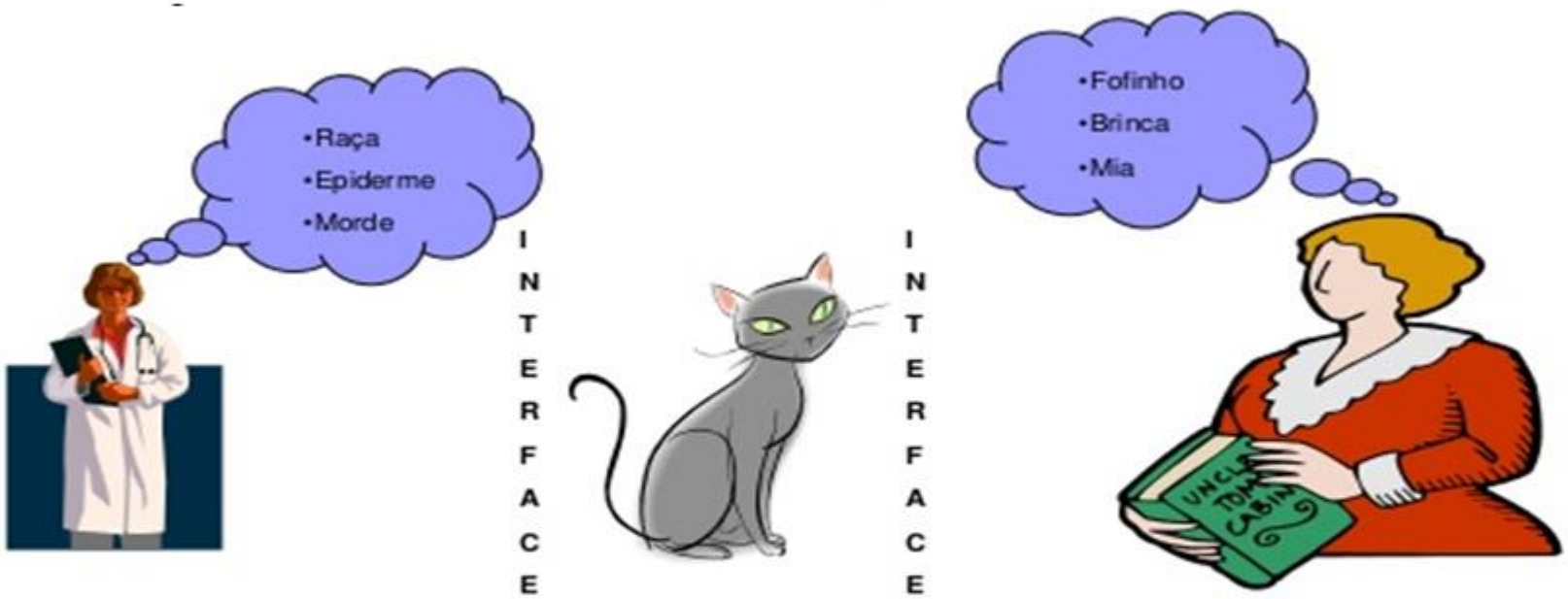
- O domínio do problema pode ser visualizado como uma coleção de objetos e métodos associados.

ORIENTAÇÃO A OBJETOS

- Objeto é a representação de elementos do mundo real, sob o ponto de vista do problema.
- Todo objeto é identificável.
- As coisas do mundo real são denominadas objetos.

ORIENTAÇÃO A OBJETOS

- **Abstração:**
  - **Consiste na concentração nos aspectos essenciais, próprios de uma entidade e em ignorar suas propriedades acidentais.**
  - **Foco em aspectos relevantes para um determinado propósito.**



Nota : Duas ou mais pessoas podem ver características distintas em um mesmo objeto. O objeto comporta ambas as características porém cada uma das pessoas visualiza aquilo que mais atendem a sua realidade, mediante aos seus próprios filtros internos. Isso chama-se **abstração.**

ORIENTAÇÃO A OBJETOS

- **Encapsulamento:**
  - **consiste na separação dos aspectos externos de um objeto, acessíveis por outros objetos, dos detalhes internos da implementação daquele objeto, que ficam ocultos dos demais objetos.**
  - **significa separar o programa em partes, o mais isoladas possível. A ideia é tornar o software mais flexível, fácil de modificar e de criar novas implementações.ee**

ORIENTAÇÃO A OBJETOS

- **Encapsulamento:**
  - **é uma forma de restringir o acesso ao comportamento interno de um objeto. Cada objeto possui uma interface que é o que ele conhece e o que ele sabe fazer, sem descrever como o objeto faz.**

Para dirigir não precisamos conhecer o funcionamento interno do acelerador e da embreagem, só precisamos saber que para andar é preciso pisar na embreagem, passar a marcha e pisar no acelerador....esse conhecimento já basta – a interface embreagem, marcha e acelerador já são suficientes para fazer o carro andar, não importa como isso tudo funciona internamente no carro.

ORIENTAÇÃO A OBJETOS

- Herança:
  - é o compartilhamento de atributos e operações entre classes com base em um relacionamento hierárquico.
  - Permite que a estrutura comum seja compartilhada por diversas subclasses semelhantes sem redundâncias.
  - Cada classe em um nível de hierarquia herda as características das classes nos níveis acima.

ORIENTAÇÃO A OBJETOS

- Uma abstração útil resulta de uma decomposição inteligente da descrição do problema em elementos independentes e intuitivamente corretos.

- Bibliotecas de classe pré-definidas simplificam muito o processo do projeto.

CLASSES

- É uma abstração de um conjunto de coisas que possuem características e operações em comum.
- Surge da união de vários objetos que possuem coisas em comum.
- São a estrutura de dados que darão ao programador a noção do domínio do problema.
- O modelo de classes tem os dados e o comportamento destes.

CLASSES - ATRIBUTOS

- Correspondem à descrição dos dados armazenados pelos objetos.

- A cada atributo de uma classe está associado um conjunto de valores que esse atributo pode assumir.

- Cada instância de classe assume valores diferentes para cada atributo.

CLASSES - OPERAÇÕES

• Correspondem à descrição das ações que os objetos de uma classe sabem realizar.

• Objetos de uma classe compartilham as mesmas operações.

• A operação especifica um serviço que pode ser requerido de um objeto de uma classe, um método é a implementação de uma operação.

CLASSES - RESPONSABILIDADES

• Contrato ou obrigação de uma classe.

• Especificar, inicialmente, as responsabilidades dos itens do vocabulário.

•Identifique os itens que usuários ou programadores usam para descrever o problema ou a solução.

• Para cada abstração, identifique uma gama de responsabilidades.

CLASSES - RESPONSABILIDADES

- Identifique os itens que usuários ou programadores usam para descrever o problema ou a solução.

- Para cada abstração, identifique uma gama de responsabilidades.

CLASSES - VOCABULÁRIO

- Certifique-se que cada classe está definida e que existe um bom balanceamento entre as responsabilidades entre elas.
- Identifique os atributos e operações que são necessárias para o cumprimento das responsabilidades de cada classe.
- A identificação das classes tem como objetivo saber quais objetos irão compor o sistema.
- O modelador analisa cada Caso de Uso para identificar as classes candidatas a desempenhar responsabilidades.

CLASSES - IDENTIFICAÇÃO

- O nome do ator deve ser removido da lista de classes candidatas se não for necessário que o sistema mantenha informações sobre o mesmo.
- Uma única classe não deve ser sobrecarregada com responsabilidades demais.
- Não se deve concentrar a inteligência do sistema em uma única classe.

CLASSES – IDENTIFICAÇÃO - MÉTODOS

- No método **dirigido a dados**, a ênfase está na identificação da estrutura dos conceitos relevantes para um domínio de negócio.

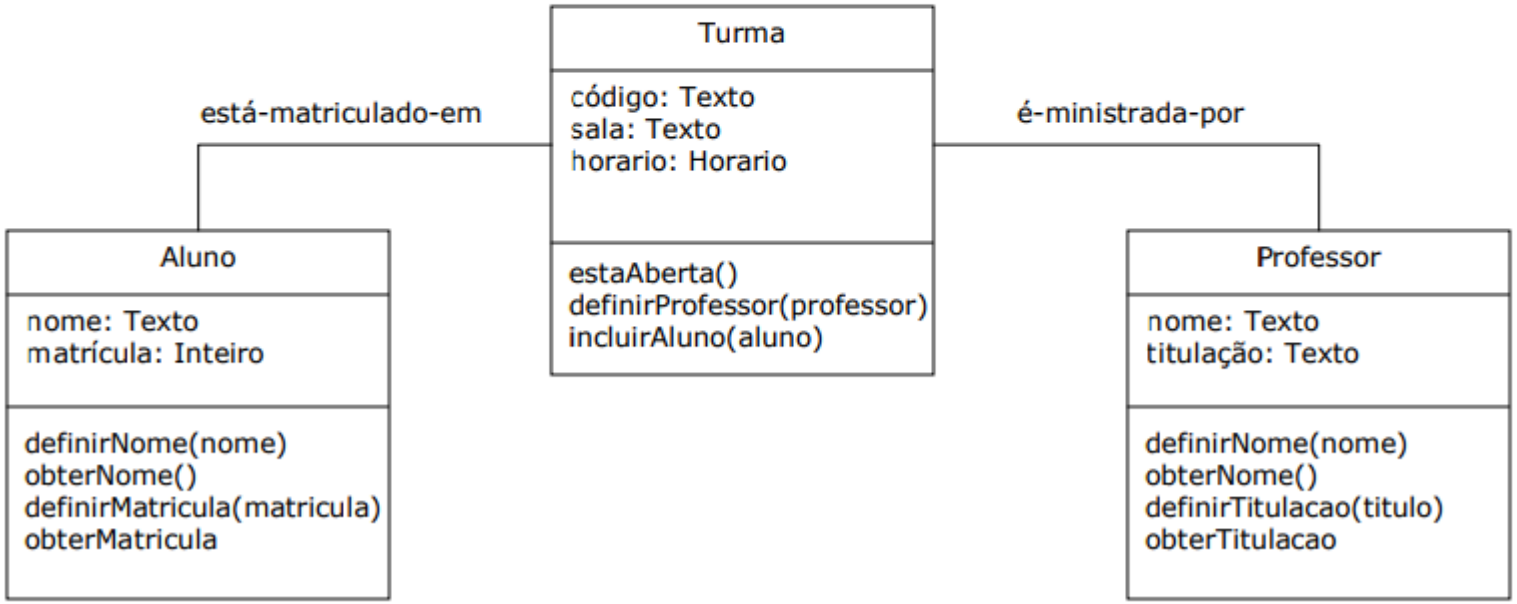
- Resulta em um modelo conceitual do sistema.

- No método **dirigido a responsabilidades**, a ênfase está na identificação de classes a partir de seus comportamentos relevantes para o sistema.

- Enfatiza o encapsulamento da estrutura e do comportamento dos objetos.

DIAGRAMA DE CLASSE

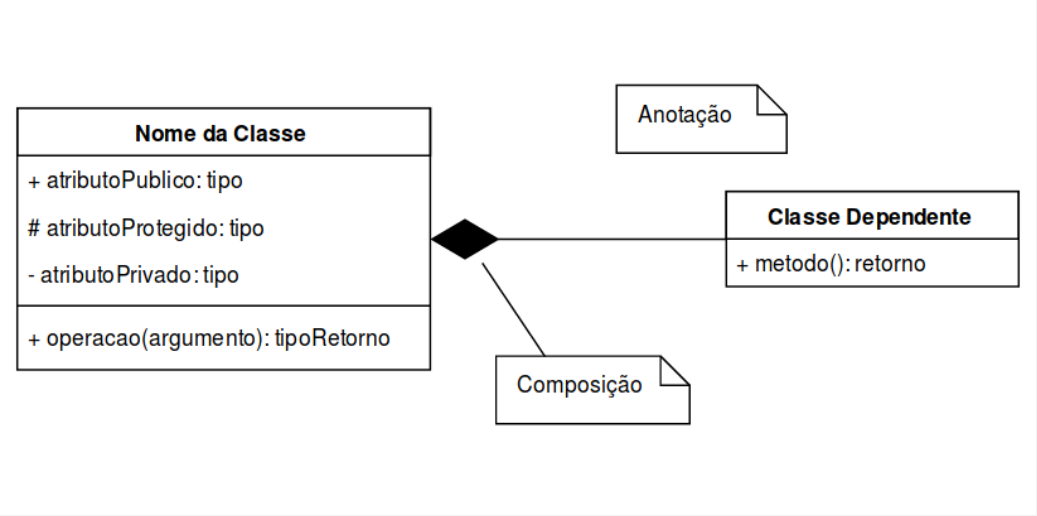
- Mostra um conjunto de classes e seus relacionamentos.
- É o diagrama central da modelagem orientada a objetos.



ELEMENTOS

- Classes

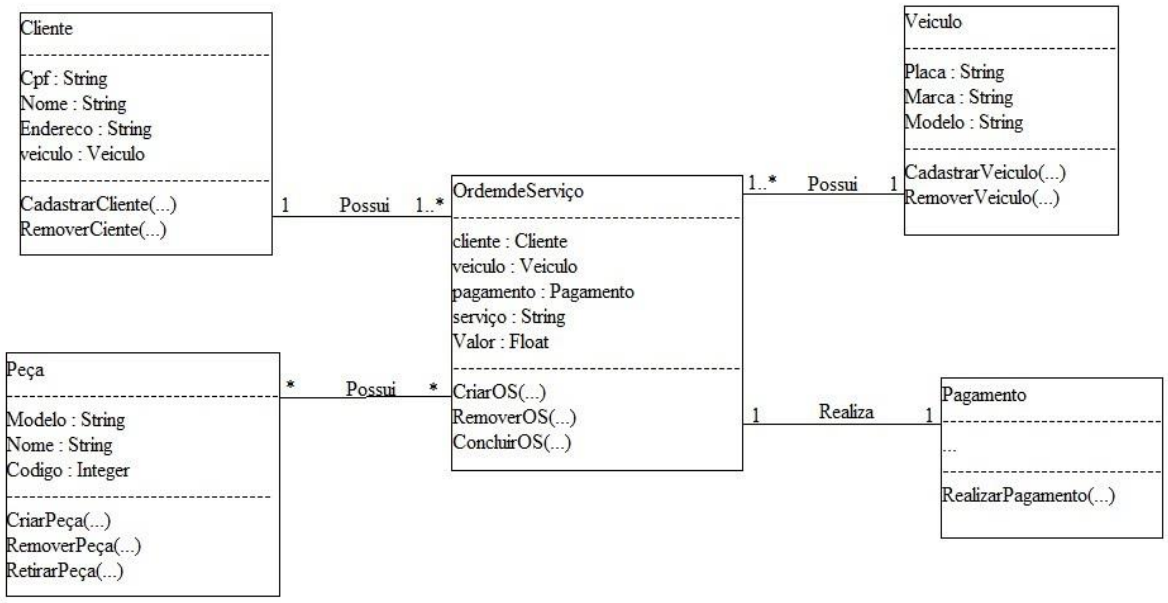
- Relacionamentos
  - Associação
  - Agregação
  - Composição
  - Generalização
  - Dependência



- Métodos relacionados a cada classe

CLASSES

Graficamente, as classes são representadas por retângulos incluindo nome, atributos e métodos.



- Devem receber nomes de acordo com o vocabulário do domínio do problema.
- É comum adotar um padrão para nomeá-las Ex: todos os nomes de classes serão substantivos singulares com a primeira letra maiúscula

CLASSES

Atributos:

– Representam o conjunto de características (estado) dos objetos daquela classe

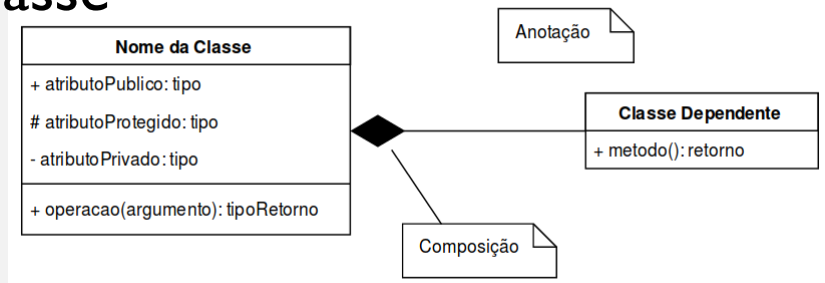
– Visibilidade:

+ público: visível em qualquer classe de qualquer pacote

protegido: visível para classes do mesmo pacote

- privado: visível somente para classe

Exemplo: + nome : String



CLASSES

Métodos:

– Representam o conjunto de operações (comportamento) que a classe fornece

– Visibilidade:

+ público: visível em qualquer classe de qualquer pacote

protegido: visível para classes do mesmo pacote

- privado: visível somente para classe

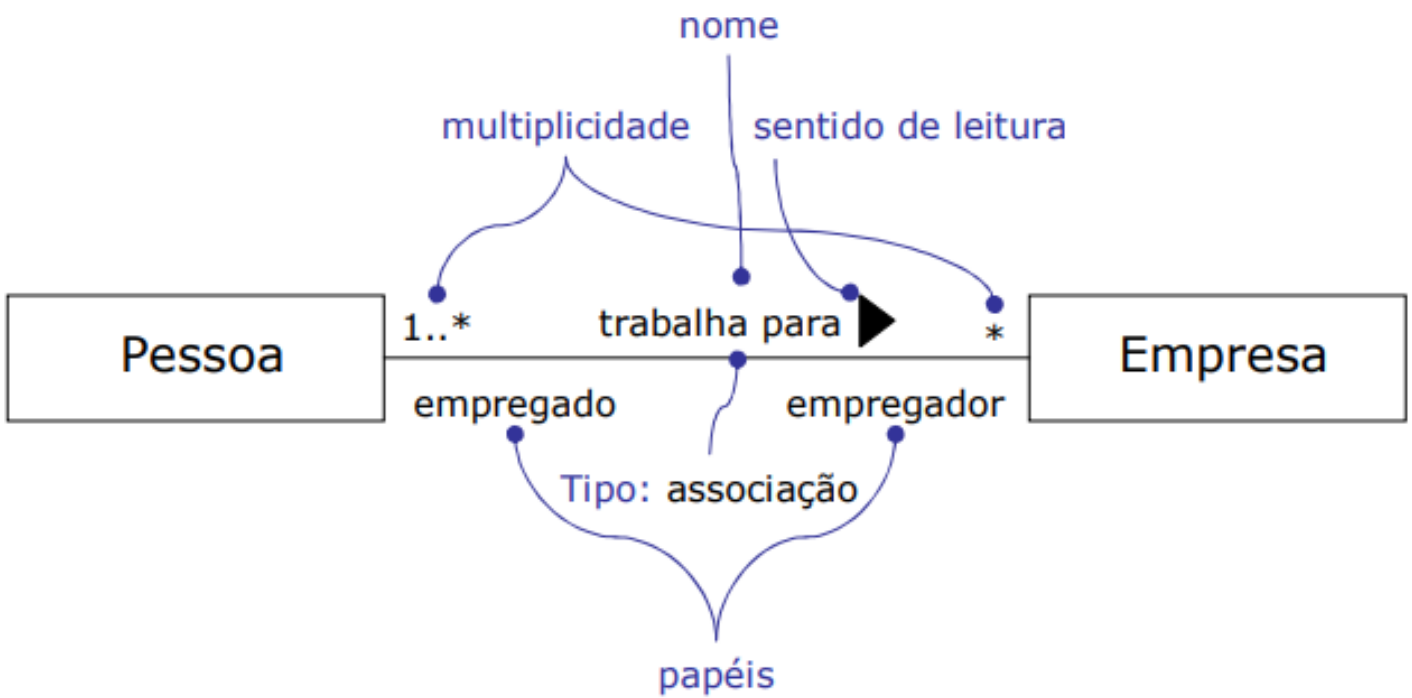
Exemplo: - getNome() : String

CLASSES

Os relacionamentos possuem:

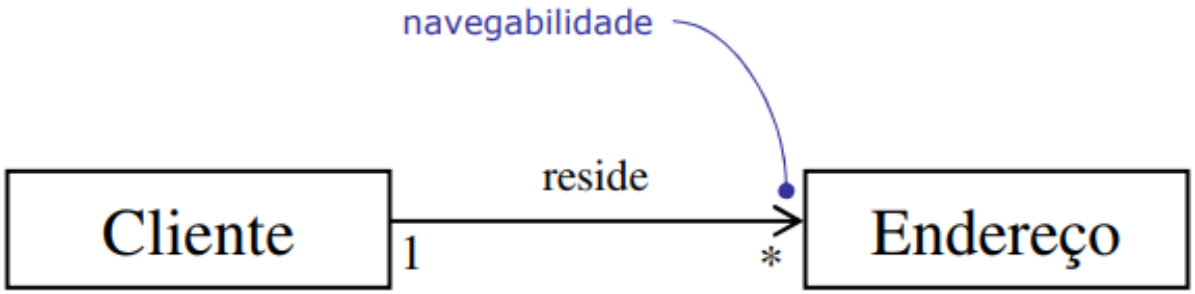
- Nome: descrição dada ao relacionamento (faz, tem, possui,...)
- Sentido de leitura
- Navegabilidade: indicada por uma seta no fim do relacionamento
- Multiplicidade: 0..1, 0..*, 1, 1..*, 2, 3..7
- Tipo: associação (agregação, composição), generalização e dependência
- Papéis: desempenhados por classes em um relacionamento

CLASSES - RELACIONAMENTOS



E a navegabilidade?

CLASSES - RELACIONAMENTOS



- O cliente sabe quais são seus endereços, mas o endereço não sabe a quais clientes pertence

Classes – Relacionamentos – Associação

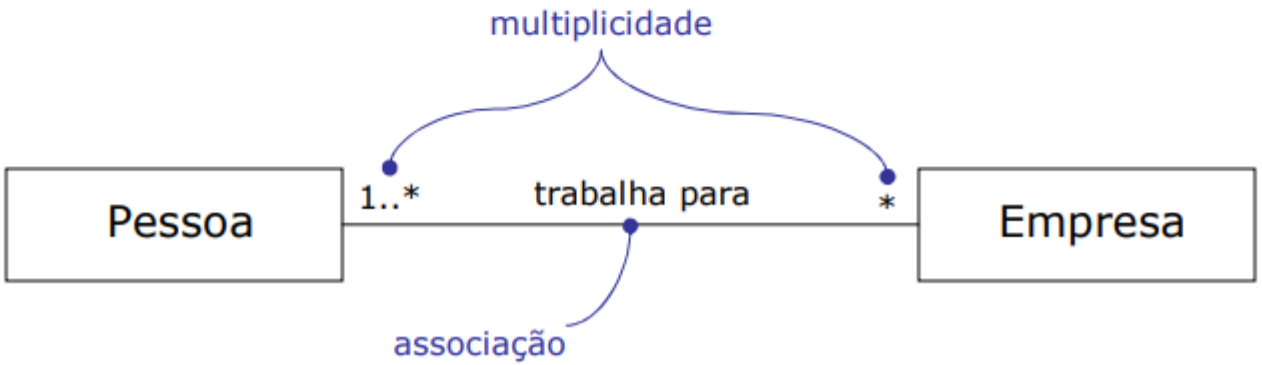
Relacionamentos: Associação

- Uma **associação** é um relacionamento estrutural que indica que os objetos de uma classe estão vinculados a objetos de outra classe.
- Uma associação é representada por uma linha sólida conectando duas classes.

Classes – Relacionamentos – Associação

Relacionamentos: Associação

- Indicadores de multiplicidade:
  - 1 Exatamente um
  - 1..* Um ou mais
  - 0..* Zero ou mais (muitos)
  - * Zero ou mais (muitos)
  - 0..1 Zero ou um
  - m..n Faixa de valores (por exemplo: 4..7)

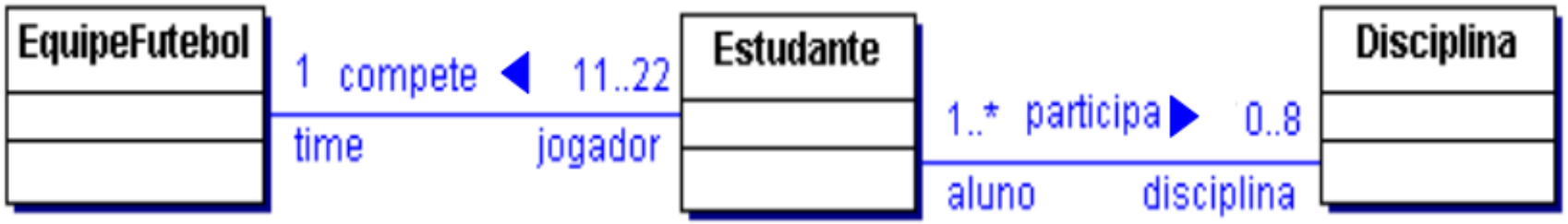


Classes – Relacionamentos – Associação

Relacionamentos: Associação

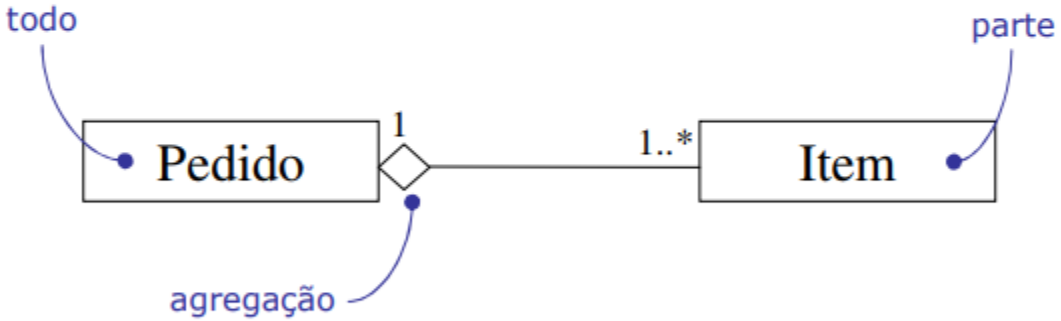
Exemplo:

- Um **Estudante** pode ser
  um aluno de uma Disciplina e
  um jogador da Equipe de Futebol
- Cada Disciplina deve ser cursada por no mínimo 1 aluno
- Um aluno pode cursar de 0 até 8 disciplinas



Classes – Relacionamentos – Agregação

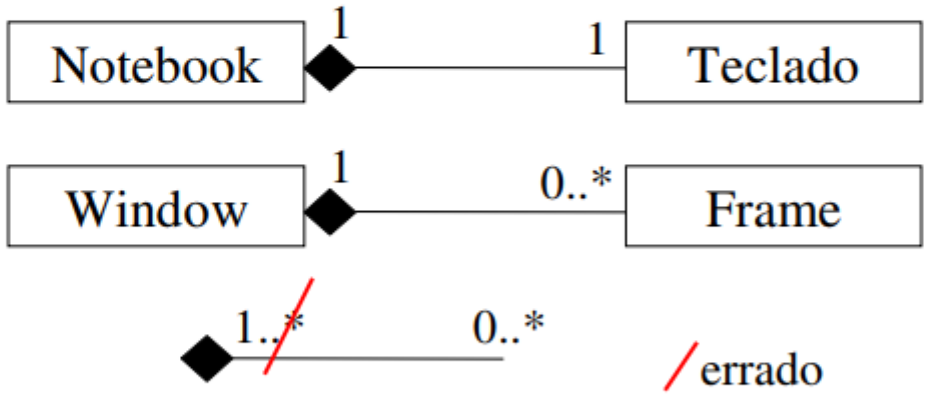
- Relacionamento: Agregação
  - É um tipo especial de associação
  - Utilizada para indicar "todo-parte"



  - um objeto "parte" pode fazer parte de vários objetos "todo"

Classes – Relacionamentos – Composição

- Relacionamento: Composição
  - É uma variante semanticamente mais "forte" da agregação
  - Os objetos "parte" só podem pertencer a um único objeto "todo" e têm o seu tempo de vida coincidente com o dele

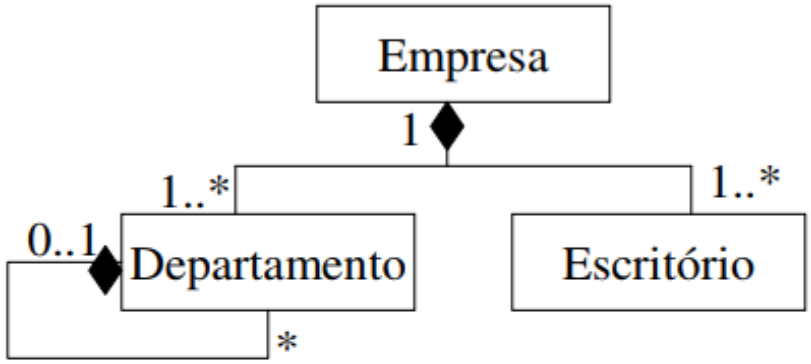


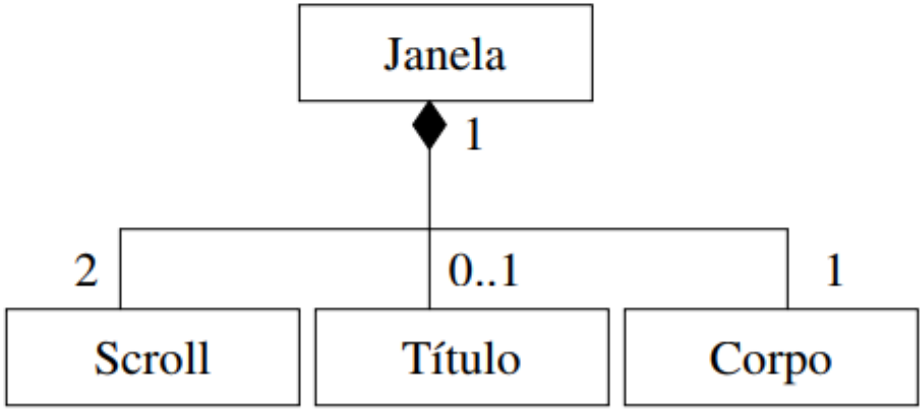
  - Quando o "todo" *morre* todas as suas "partes" também *morrem*

Classes – Relacionamentos – Composição

- Relacionamento: Composição

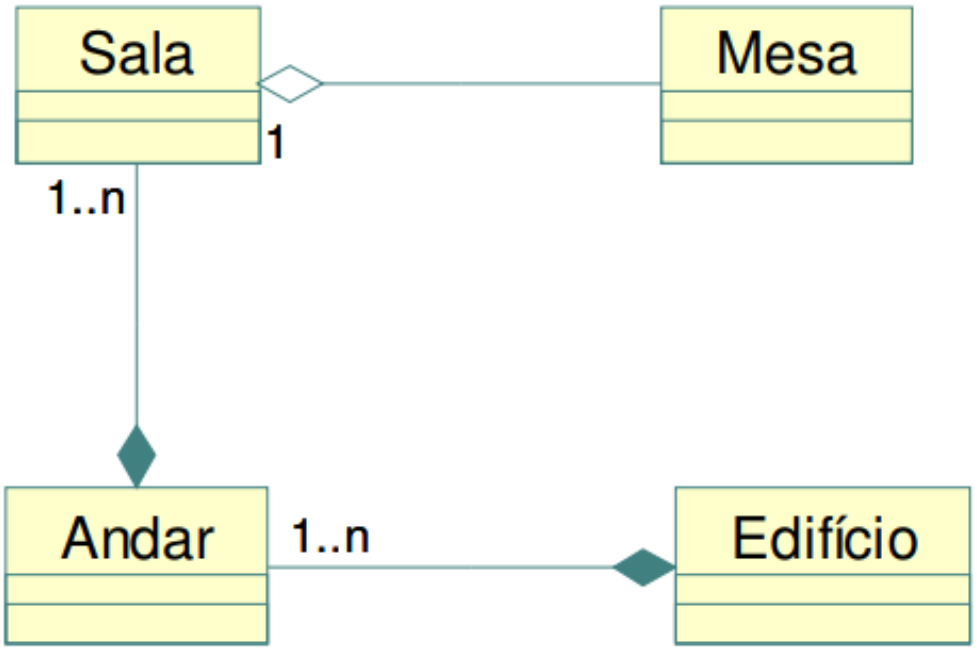
  Ex:





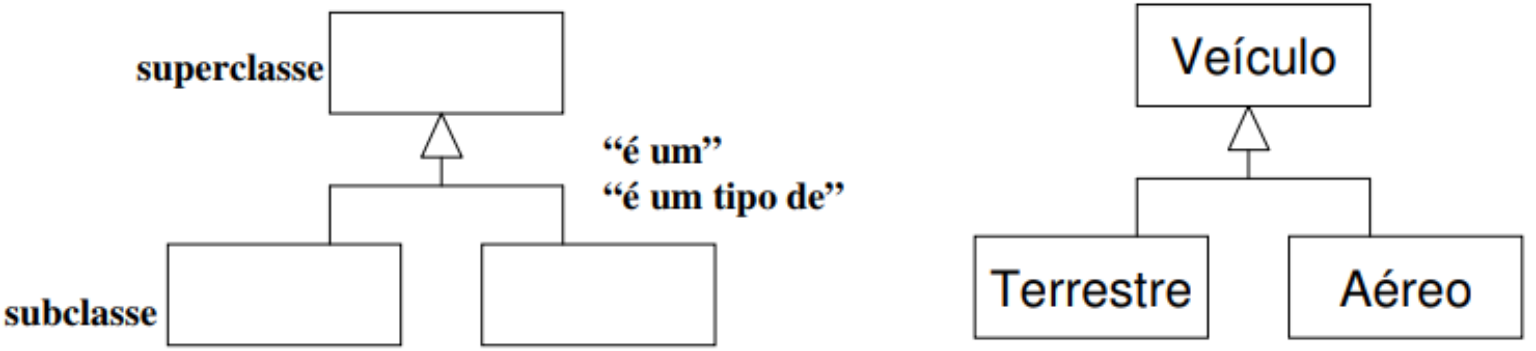
Classes – Relacionamentos

- Agregação **X** Composição



Classes – Relacionamentos – Generalização

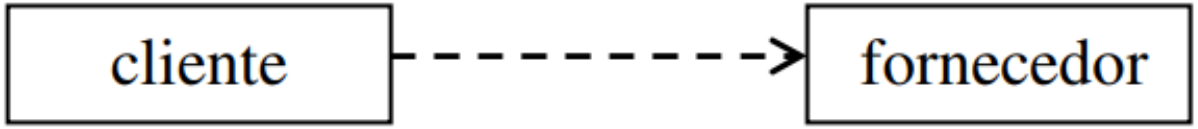
- Relacionamento: Generalização
  - É um relacionamento entre itens gerais (superclasses) e itens mais específicos (subclasses)



Classes – Relacionamentos – Dependência

- Relacionamento: Dependência
  - Representa que a alteração de um objeto (o objeto indepedendente) pode afetar outro objeto (o objeto dependente)

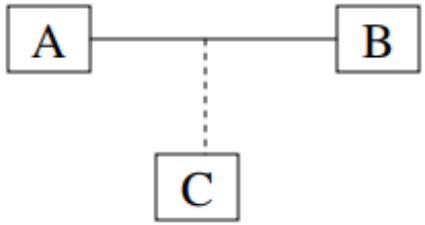
Ex:



Obs:

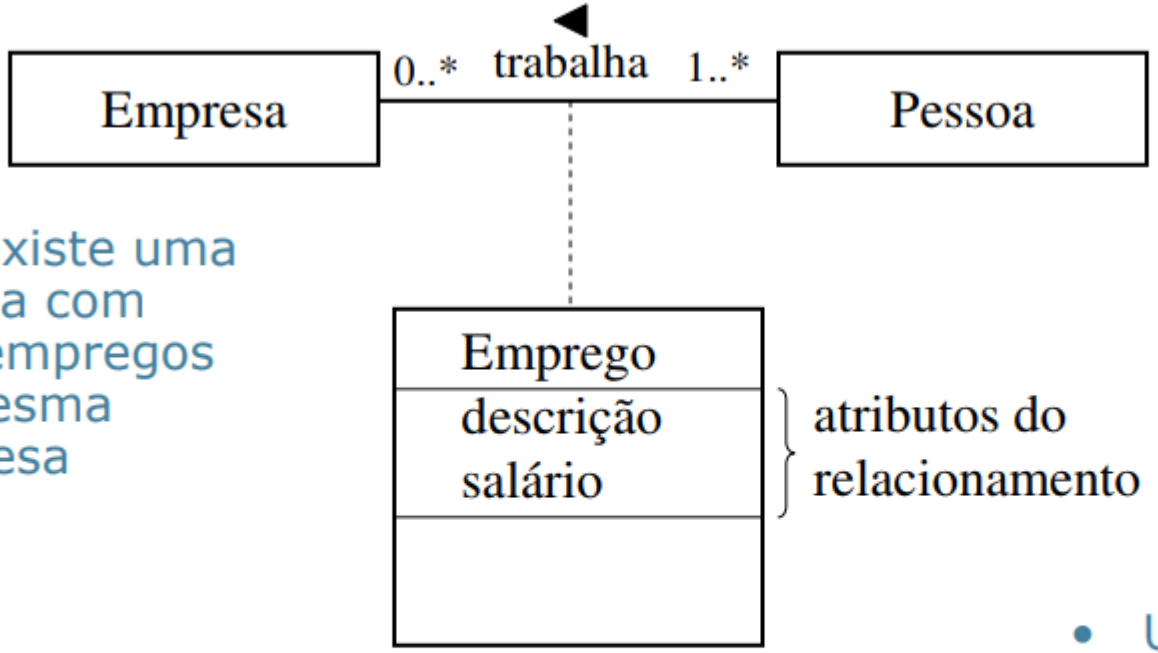
- A classe cliente depende de algum serviço da classe fornecedor
- A mudança de estado do fornecedor afeta o objeto cliente
- A classe cliente não declara nos seus atributos um objeto do tipo fornecedor
- Fornecedor é recebido por parâmetro de método

- Classe de associação
  - Usada quando uma associação entre duas classes contiver atributos da associação
    - Atributos farão parte da classe de associação
    - C existe para todo relacionamento de A com B



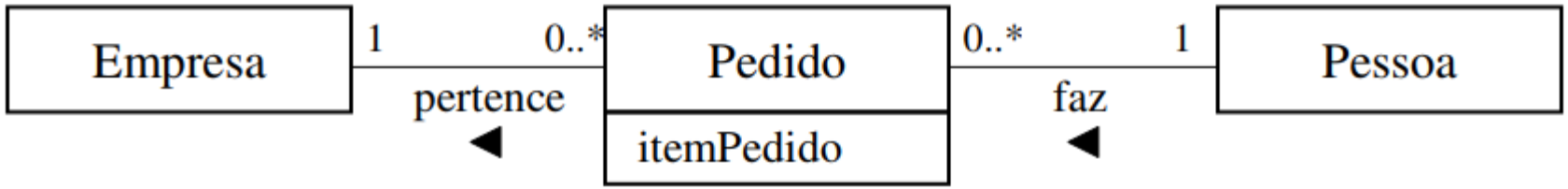
Classes

- Classe de associação



- Não existe uma pessoa com dois empregos na mesma empresa

- Uma pessoa pode fazer mais de um pedido na mesma empresa



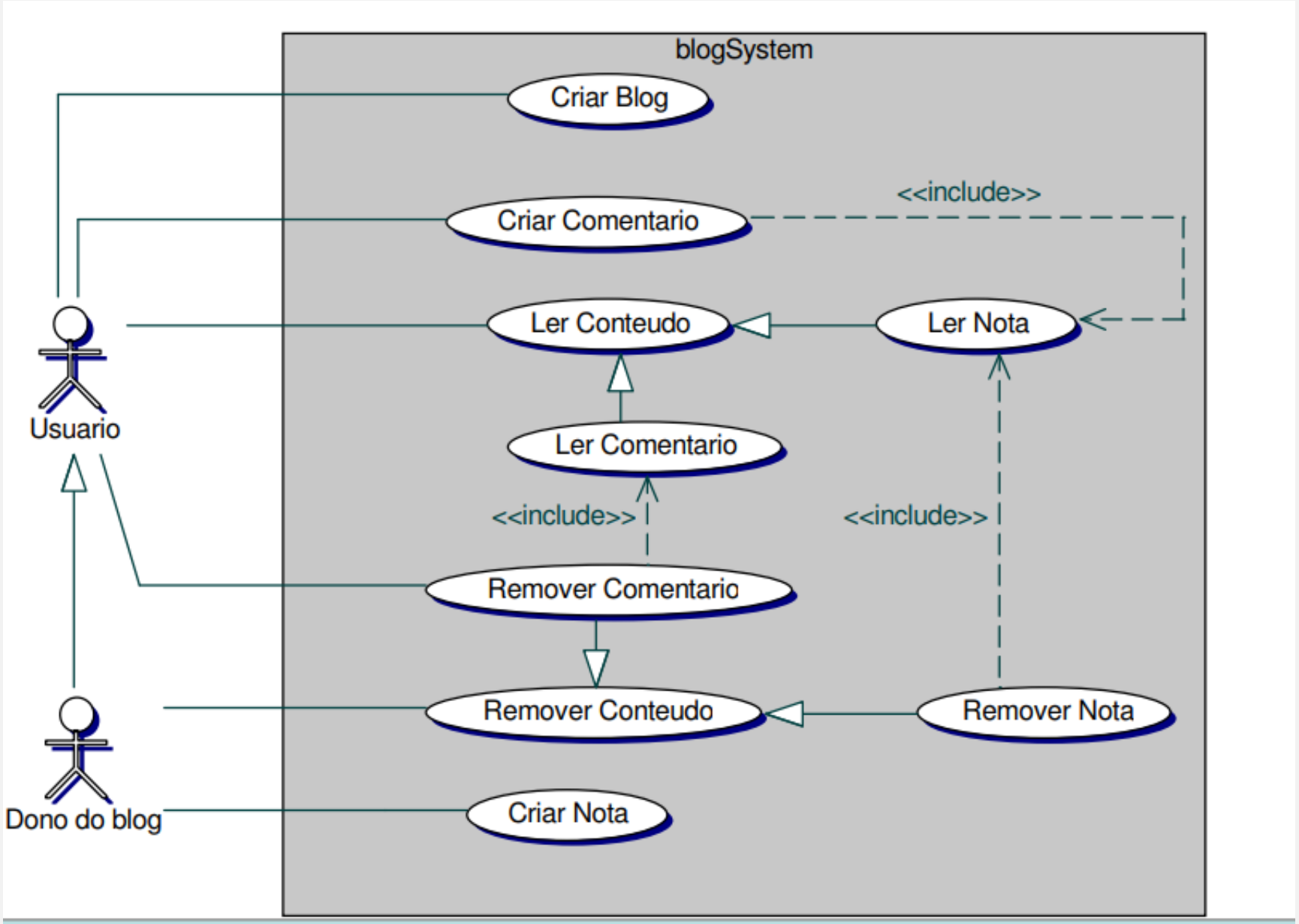
Exemplo: Blog

- Um *blog* tem um título e uma data de criação e além disso é um conjunto de conteúdos.
- Estes conteúdos (mensagens) podem ser notas ou comentários sobre as notas. Tanto notas quanto comentários têm características comuns como o texto e a data de sua criação.
- Todo usuário possui:
  - E-mail (deve ser único, ou seja, não há mais de um usuário com o mesmo e-mail)

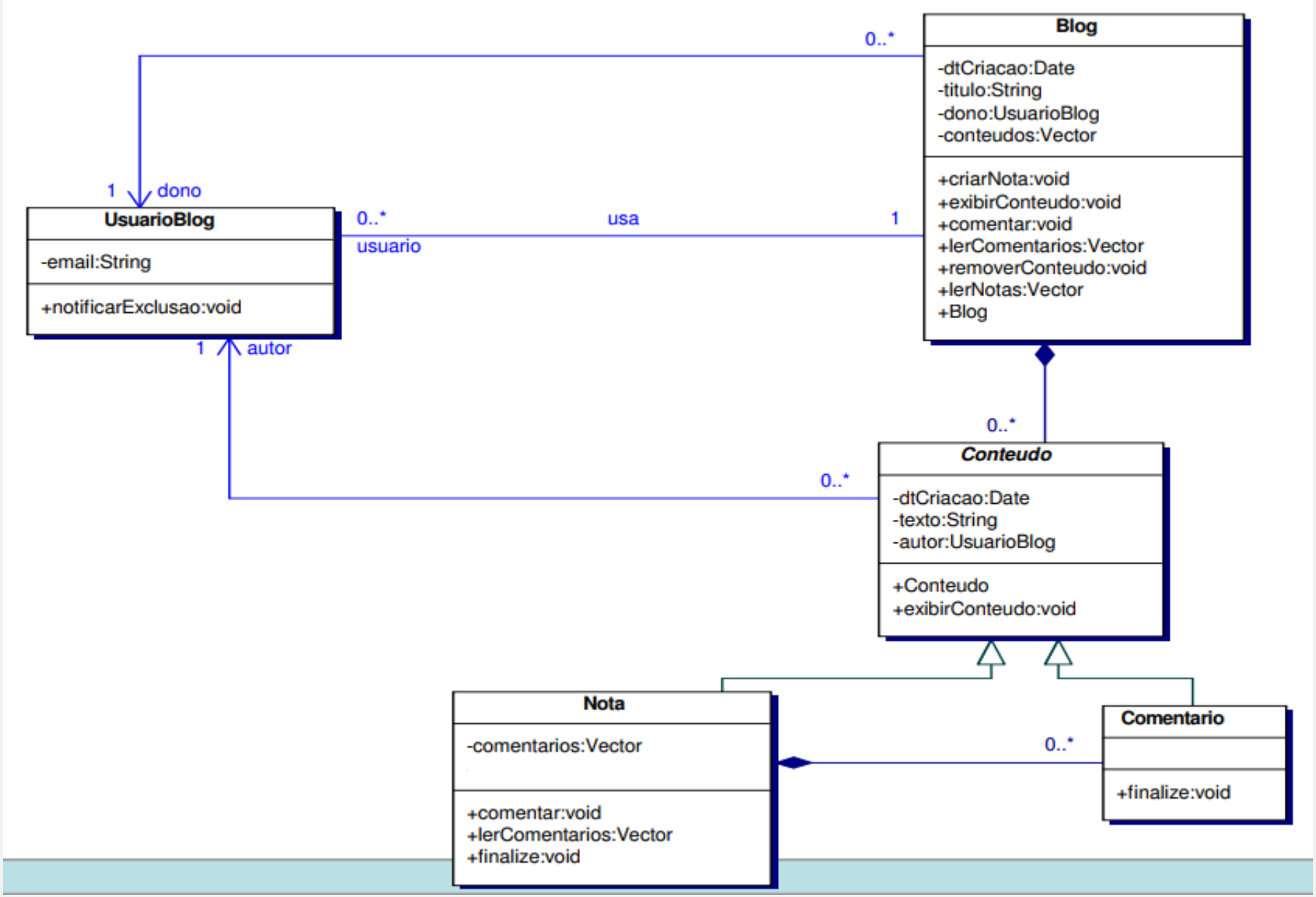
Exemplo: Blog - Requisitos

- Permitir a criação de blogs
- Permitir a utilização de blogs
  - Qualquer usuário pode ler conteúdos
  - Somente o dono do blog pode criar notas
  - Qualquer usuário pode criar comentários. Para criar um comentário o usuários precisa ler as notas.
  - Somente o dono do blog pode remover conteúdos. Para remover um conteúdo ele precisará ler o conteúdo. Caso ele remova um comentário, o autor do comentário deve ser notificado por e-mail.

Exemplo: Blog – Diagrama de Casos de Uso



Exemplo: Blog – Diagrama de Classe



EXEMPLO: SISTEMA DE MATRÍCULA

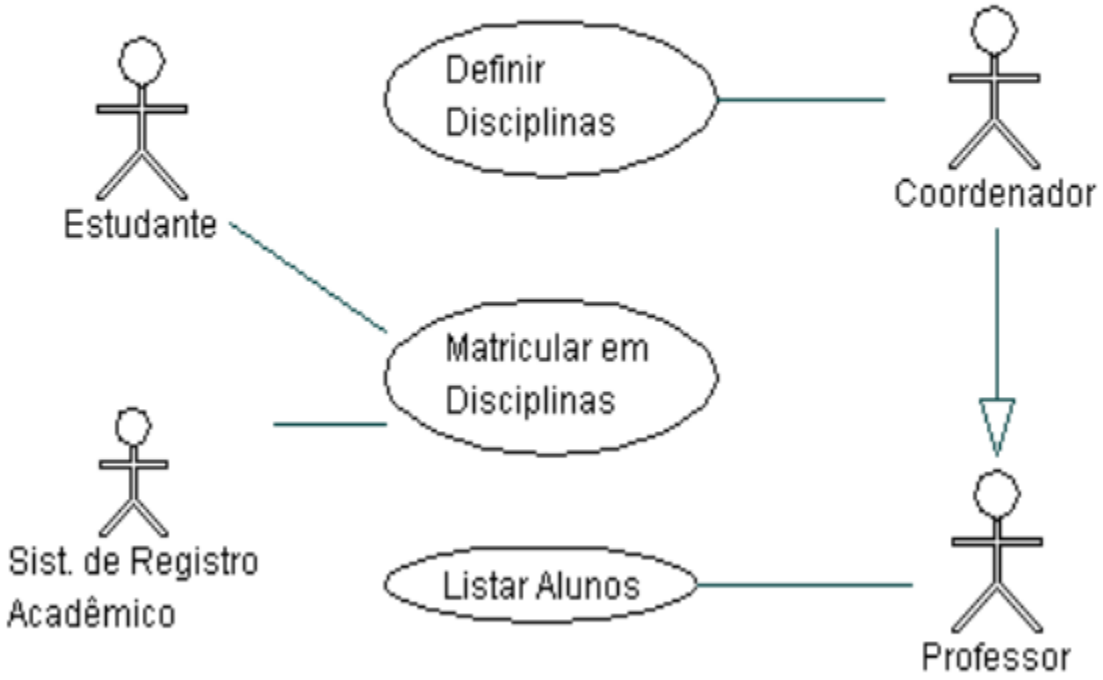
Descrição

A Universidade XYZ deseja informatizar seu sistema de matrículas:

- A universidade oferece vários cursos.
- O Coordenador de um curso define as disciplinas que serão oferecidas pelo seu curso num dado semestre.
- Várias disciplinas são oferecidas em um curso.
- Várias turmas podem ser abertas para uma mesma disciplina, porém o número de estudantes inscritos deve ser entre 3 e 10.
- Estudantes selecionam 4 disciplinas.
- Quando um estudante matricula-se para um semestre, o Sistema de Registro Acadêmico (SRA) é notificado.
- Após a matrícula, os estudantes podem, por um certo prazo, utilizar o sistema para adicionar ou remover disciplinas.
- Professores usam o sistema para obter a lista de alunos matriculados em suas disciplinas. O Coordenador também.
- Todos os usuários do sistema devem ser validados.

EXEMPLO: SISTEMA DE MATRÍCULA

Diagrama de Casos de Uso



EXEMPLO: SISTEMA DE MATRÍCULA

Descrição do Caso de Uso "Matricular em Disciplina"

- Esse caso de uso se inicia quando o Estudante de Curso inicia uma sessão no sistema e apresenta suas credenciais.
- O sistema verifica se a credencial é válida.
- O sistema solicita que o estudante realize sua matrícula, selecionando 4 disciplinas.
- O estudante preenche um formulário eletrônico de matrícula e o submete para uma análise de consistência.
- O sistema analisa as informações contidas no formulário.
  - Se as informações são consistentes, o estudante é incluído em turmas abertas de 4 disciplinas, iniciando pelas preferenciais.
  - Se as informações não são consistentes, o sistema informa o motivo da inconsistência e solicita que o formulário seja alterado.

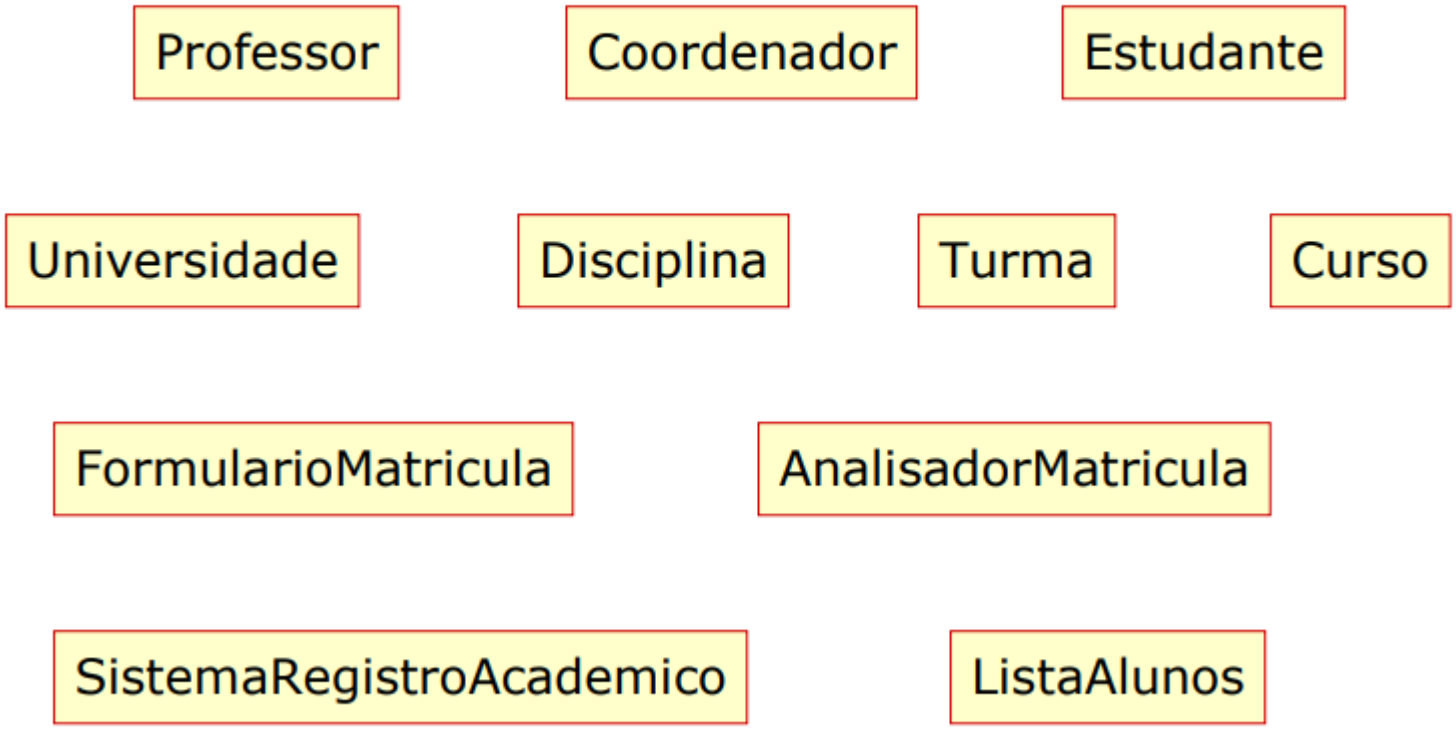
EXEMPLO: SISTEMA DE MATRÍCULA

Diagrama de Classes: identificando os relacionamentos

- Exemplos de candidatos a relacionamentos:
  - **A** é parte física ou lógica de **B**.
  - **A** está contido fisicamente ou logicamente em **B**.
  - **A** é uma descrição de **B**.
  - **A** é membro de **B**.
  - **A** é subunidade organizacional de **B**.
  - **A** usa ou gerencia **B**.
  - **A** se comunica/interage com **B**.
  - **A** está relacionado com uma transação **B**.
  - **A** é possuído por **B**.
  - **A** é um tipo de **B**.

EXEMPLO: SISTEMA DE MATRÍCULA

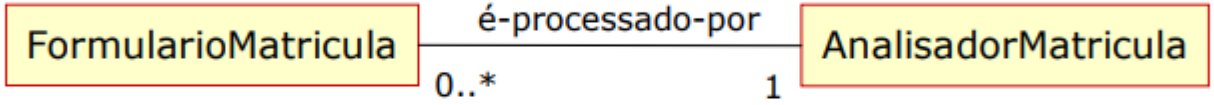
Diagrama de Classes: identificando as classes



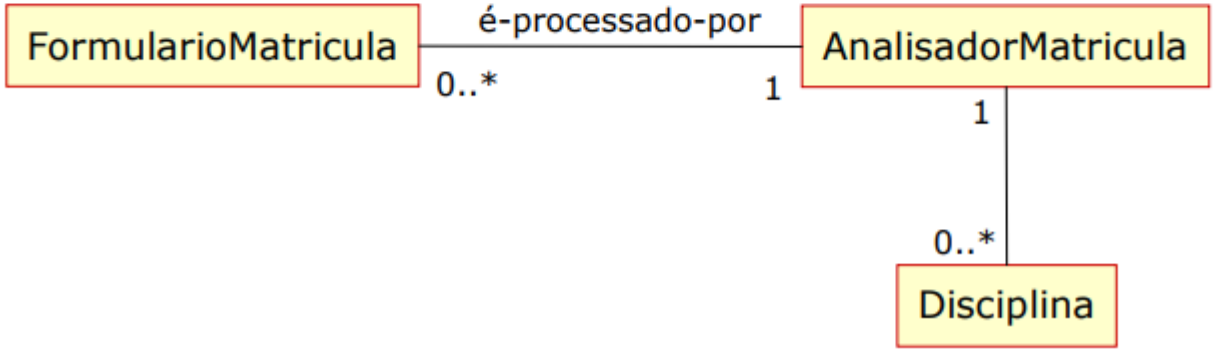
EXEMPLO: SISTEMA DE MATRÍCULA

Diagrama de Classes: identificando os relacionamentos

- O formulário de matrícula é processado por um analisador de matrícula

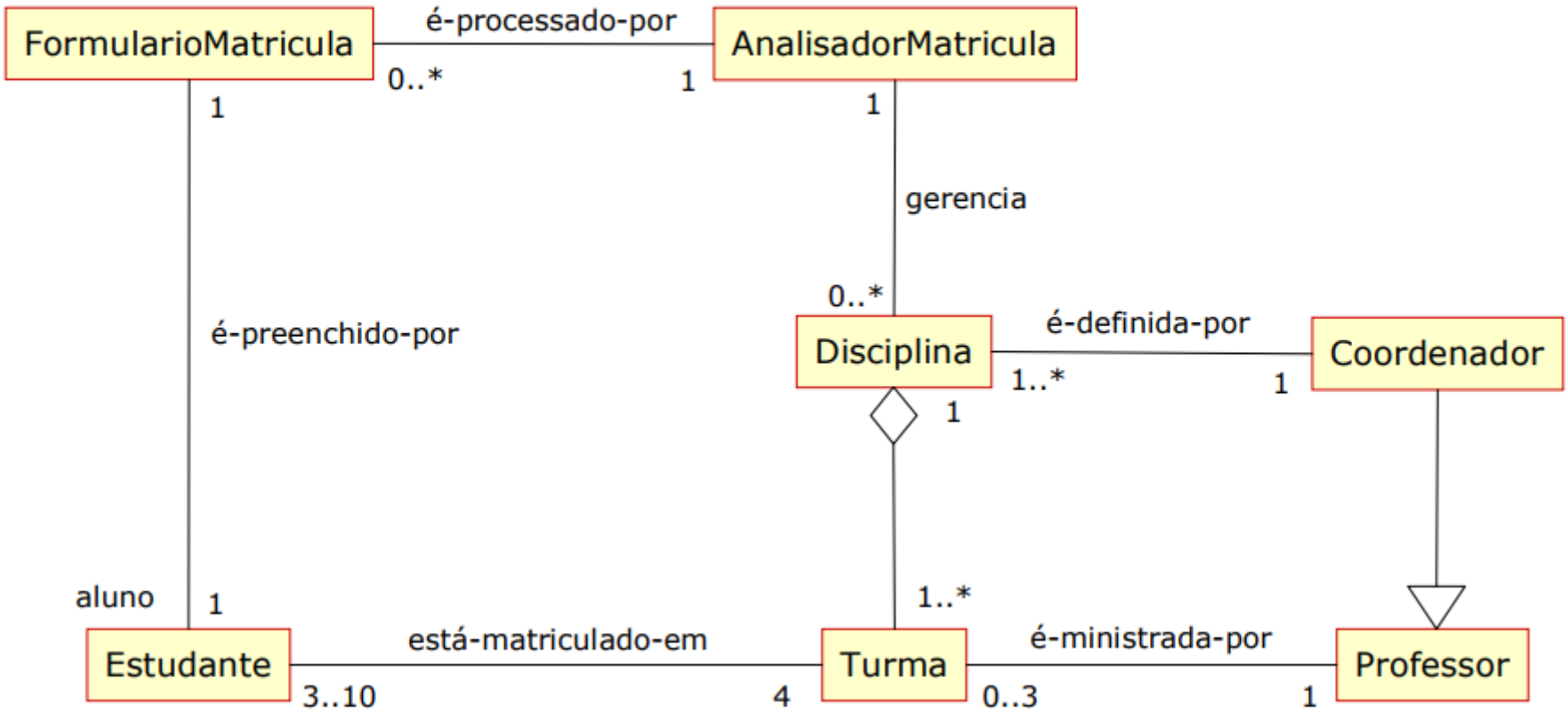


- O analisador de matrícula gerencia a disciplina



EXEMPLO: SISTEMA DE MATRÍCULA

Diagrama de Classes

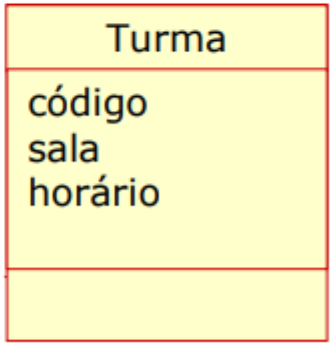


EXEMPLO: SISTEMA DE MATRÍCULA

Diagrama de Classes: identificando os atributos

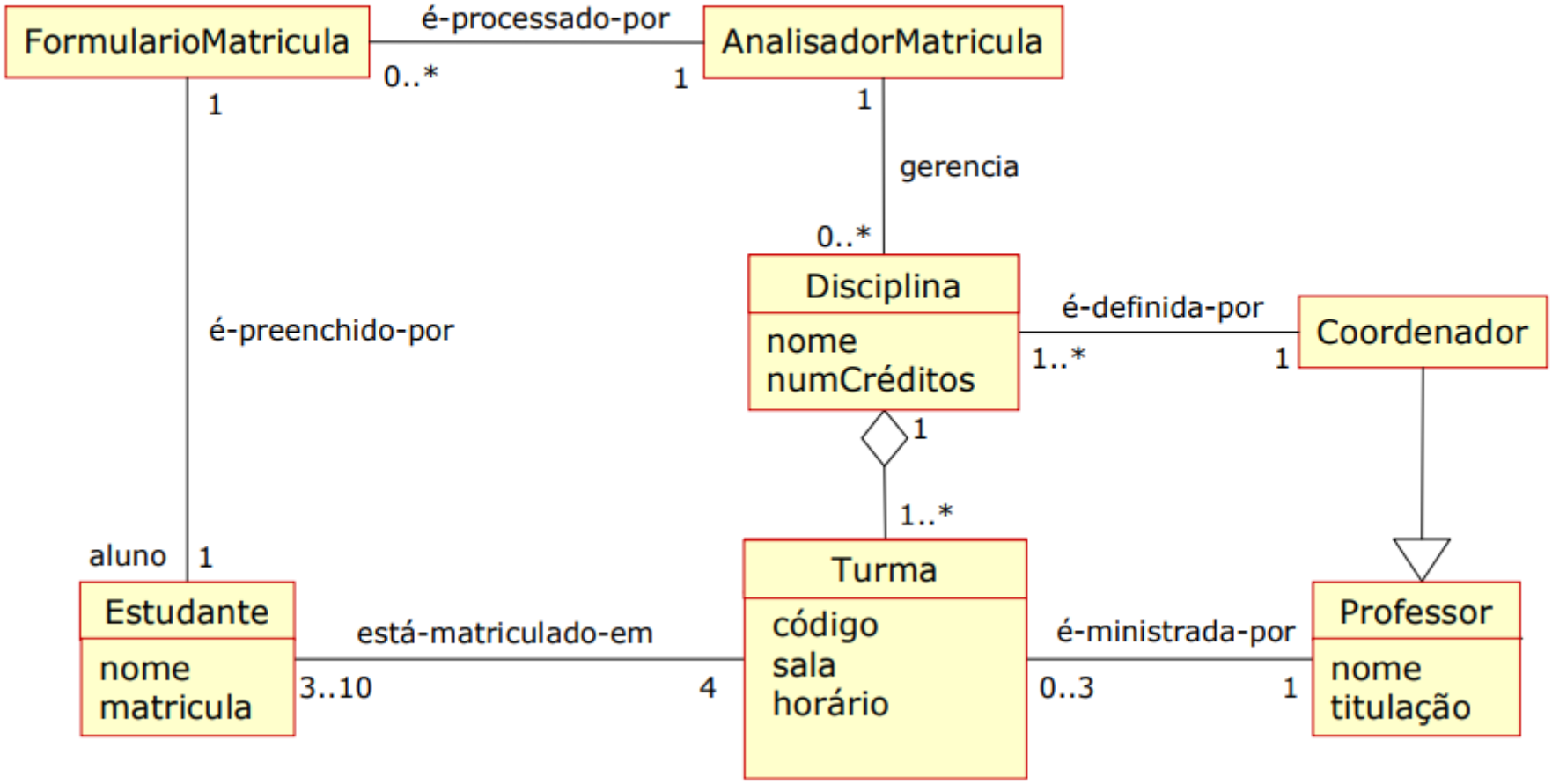
- Os atributos podem ser encontrados examinando-se as descrições dos casos de uso e também pelo conhecimento do domínio do problema.

- Cada turma oferecida possui um código, uma sala e um horário.



EXEMPLO: SISTEMA DE MATRÍCULA

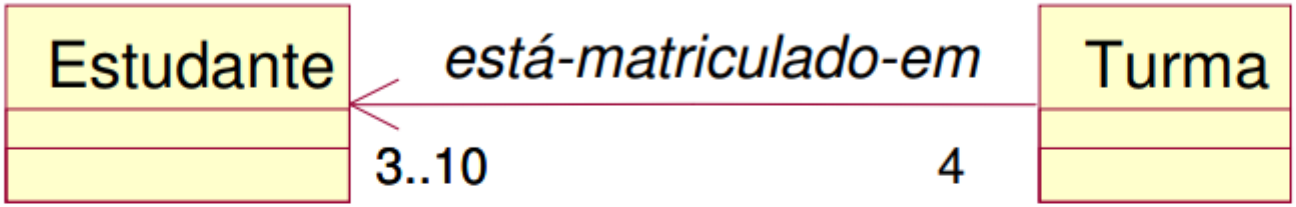
Diagrama de Classes



EXEMPLO: SISTEMA DE MATRÍCULA

Diagrama de Classes:

- E a navegabilidade?




```
public class Estudante {
   private String nome;
   private String matricula;
...
}
```

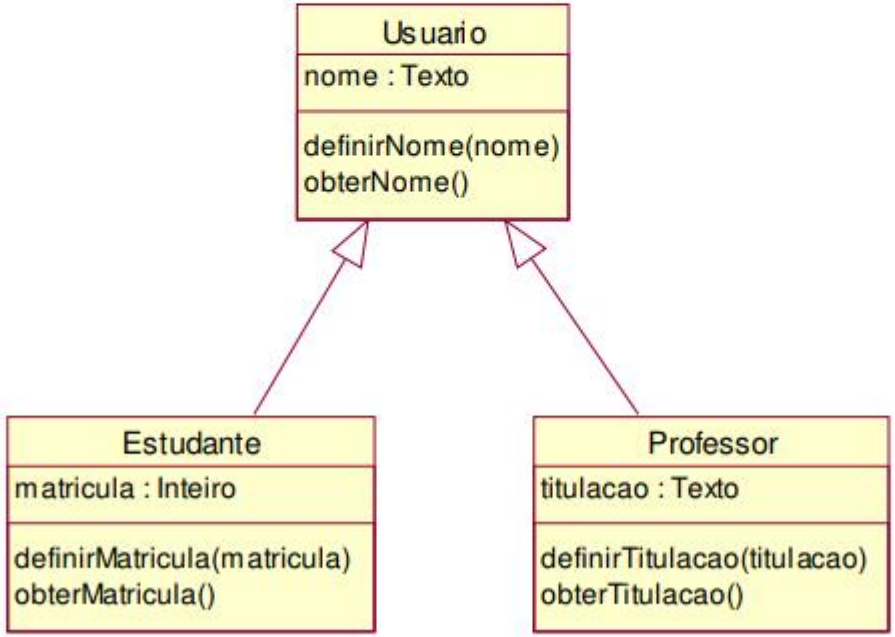
```
public class Turma {
   private String codigo;
   private String sala;
   private Estudante alunos[];
   ...
}
```

OBS: Turma não aparece como atributo de **Estudante!**

EXEMPLO: SISTEMA DE MATRÍCULA

Diagrama de Classes:

- Acrescentando generalizações:
  - Atributos, operações e/ou relacionamentos comuns podem ser movidos para uma classe mais geral.



EXEMPLO: SISTEMA DE MATRÍCULA