INSTITUTO DE CIÊNCIAS EXATAS

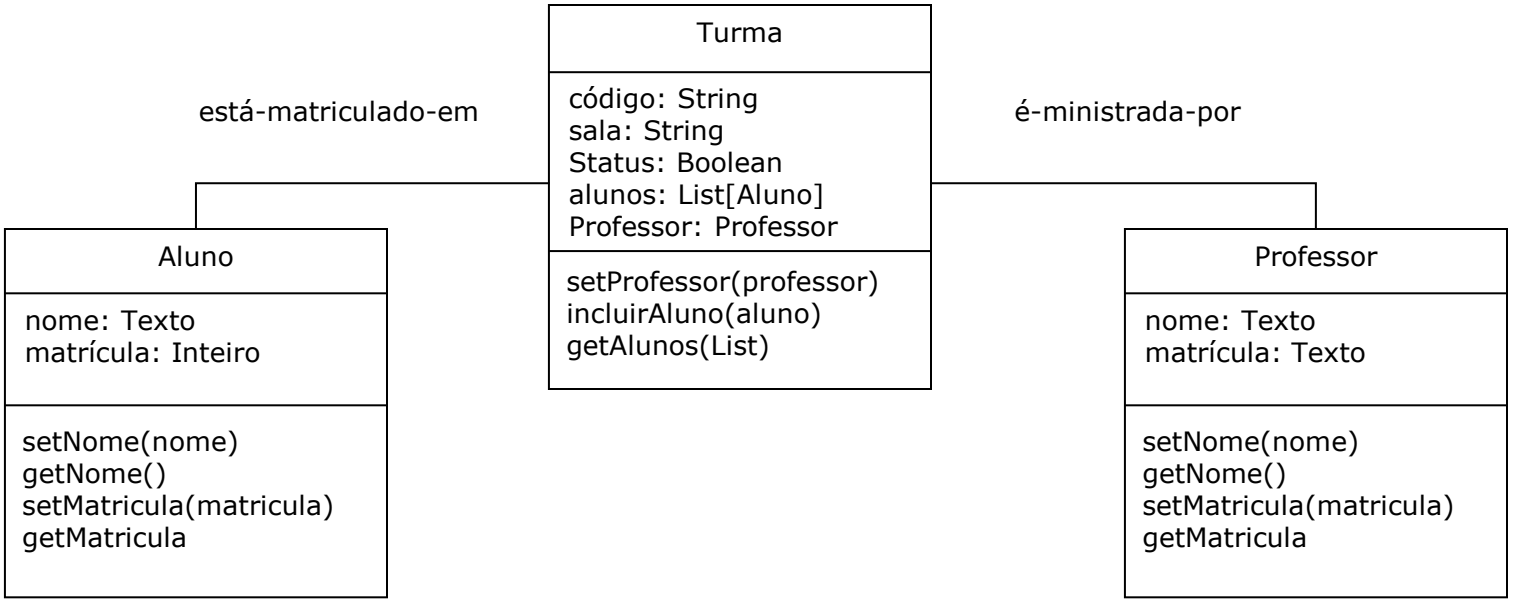
DEPARTAMENTO DE COMPUTAÇÃO

UML 2.0
Diagrama de Classes

Eduardo Kinder Almentero
ekalmentero@gmail.com

Introdução

- Permite a representação de conjunto de **classes**, suas **propriedades** (atributos), **operações** (métodos) e seus **relacionamentos**;
- É o diagrama central da modelagem orientada a objetos.

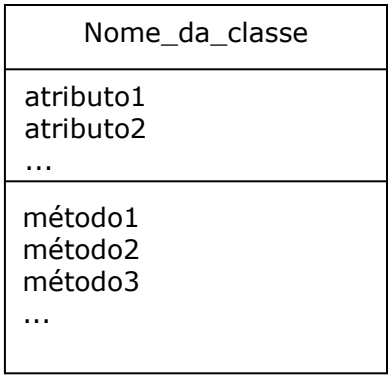


Elementos

- Classes
- Notas
- Estereótipo
- Relacionamentos
  - Associação
    - Agregação
    - Composição
  - Generalização
  - Dependência
  - Instanciação

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

- Classe abstrata
  - Uma classe que contém **pelo menos uma operação abstrata** é chamada de classe abstrata.
  - Esse tipo de classe **não pode criar instâncias**, visto que algumas de suas operações **não possuem implementação**.
  - As classes abstratas **devem possuir subclasses**, que implementem as operações abstratas.
  - A classe abstrata **pode possuir métodos concretos**.

Classes

- Interface (disponível em algumas linguagens, como Java)
  - Classe que possui **todos os métodos abstratos**;
  - É diferente de uma classe abstrata com todos os métodos abstratos, pois permite a **herança múltipla**;
  - Versões mais recentes do Java (a partir do Java 8) permitem a criação de um **método *default***.
    - E se duas interfaces possuírem um método, definido como *default*, com o mesmo nome, e uma classe herdas as duas interfaces?
      - O método precisa ser sobrescrito pela classe herdeira.
      - E para invocar o método original da interface?
        - » *minhaClasse.super.metodoDaInterface();*

Notas e Estereótipo

- Notas
  - Contem **comentários** ou **restrições** ligados a um **elemento.**
- Estereótipo
  - **Estende o vocabulário do UML** permitindo a criação de **novos tipos de elementos semelhantes aos já existentes**, porém **específicos para o seu problema**
  - Ex.:
    - <<abstract>>;
    - <<interface>>;
    - <<liderTeste>>;
    - <<testadorJunior>>.

Atributos

- Representam **o conjunto de características (estado) dos objetos** da classe;
- Visibilidade:

  "+" público: visível em qualquer classe de qualquer pacote;

  "#" protegido: visível para classes do mesmo pacote;

  "-" privado: visível somente para os métodos da própria classe.
- Exemplo:
  - + nome : String
  - - nome : String
  - # nome : String

Métodos

- Representam o conjunto de **operações (comportamento) que a classe fornece**
- Visibilidade:

  "+" público: visível em qualquer classe de qualquer pacote.

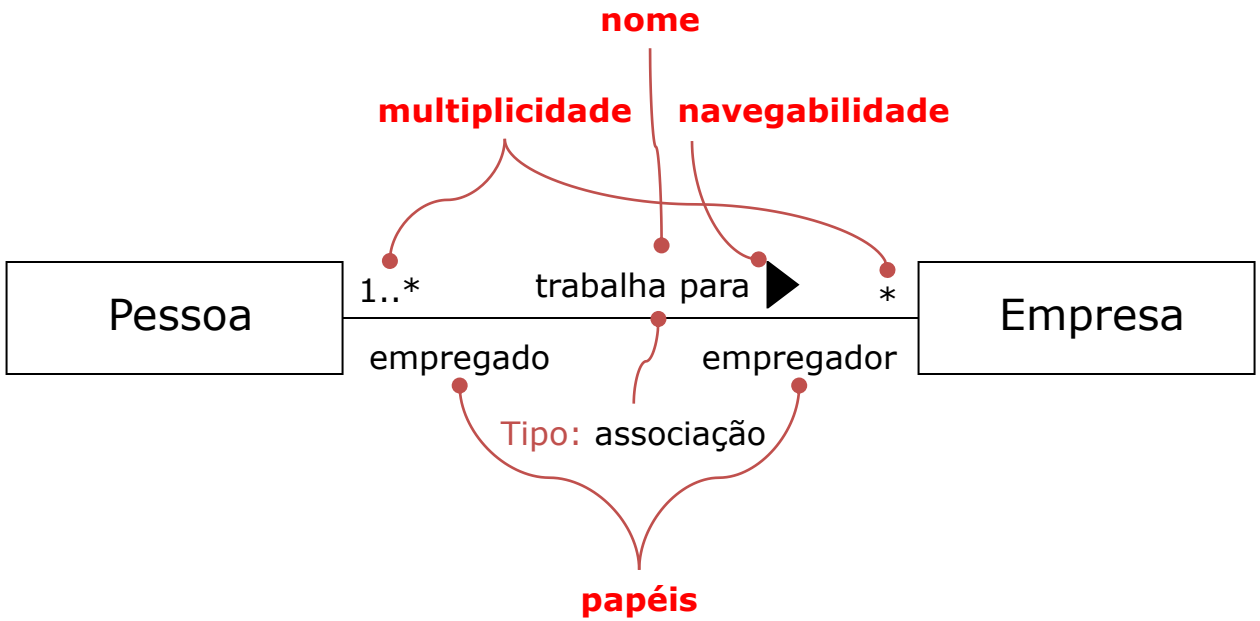
  "#" protegido: visível para classes do mesmo pacote.

  "-" privado: visível somente para classe.
- Exemplo:
  - + getNome() : String
  - # getId() : String
  - - getEmail() : String

Relacionamentos

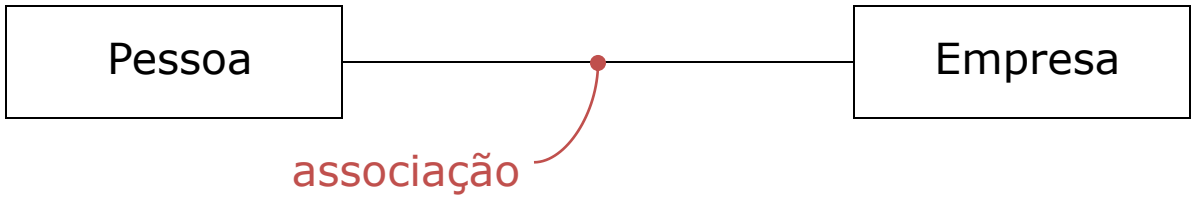
- Os relacionamentos possuem:
  - **Nome:** descrição dada ao relacionamento (faz, tem, possui, etc.);
  - **Navegabilidade:** indicada por uma seta no fim do relacionamento.
  - **Multiplicidade:** 0..1, 0..*, 1, 1..*, 2, 3..7
  - **Tipo:** associação (agregação, composição), generalização e dependência;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

Relacionamentos



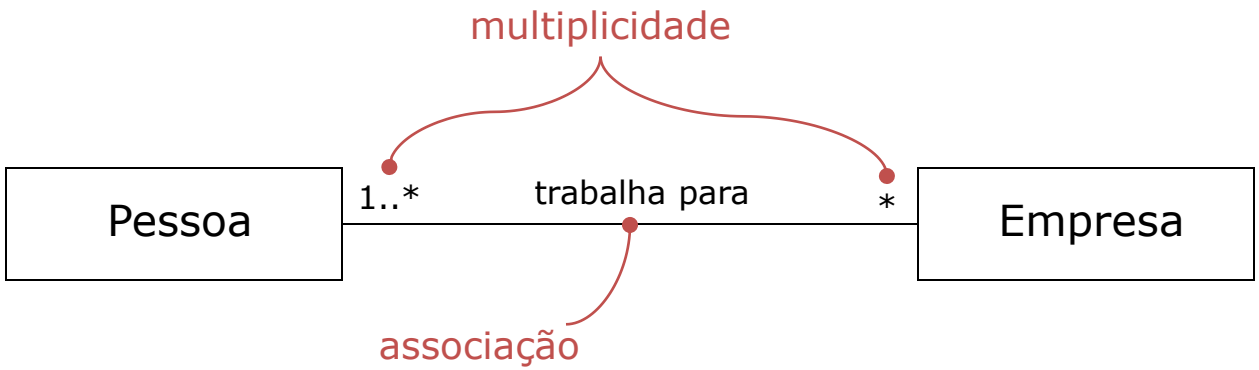
Relacionamento - Associação

- 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.



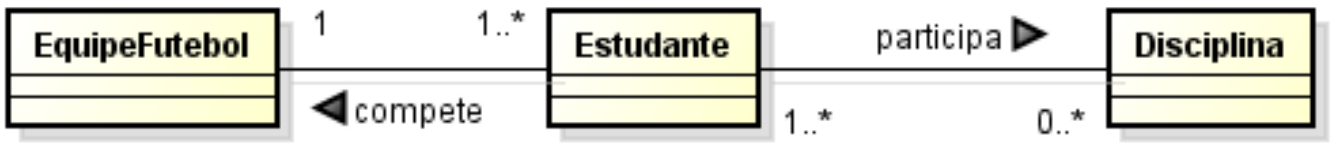
Relacionamento - Associação

- 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)

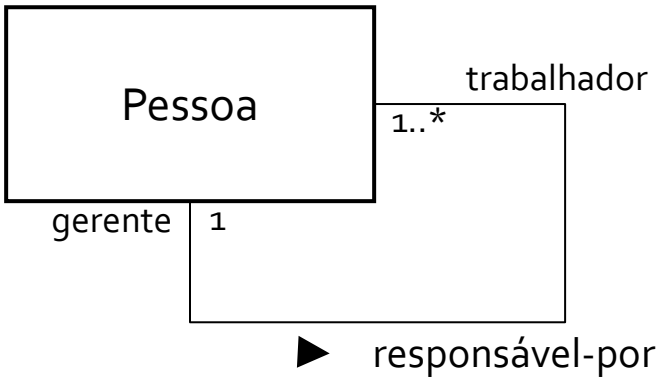


Relacionamento - Associação

- 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

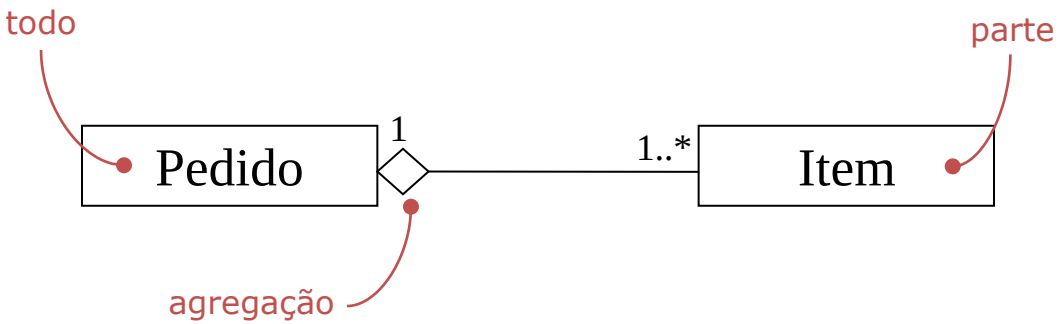


Relacionamento - Associação



Relacionamento - Agregação

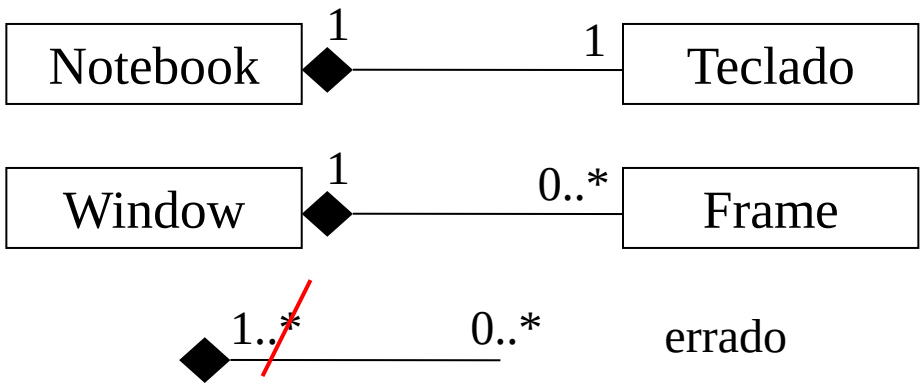
- É um **tipo especial de associação**
- Utilizada para indicar **"todo-parte"**



- Um objeto **"parte"** pode fazer parte de **vários objetos "todo"**

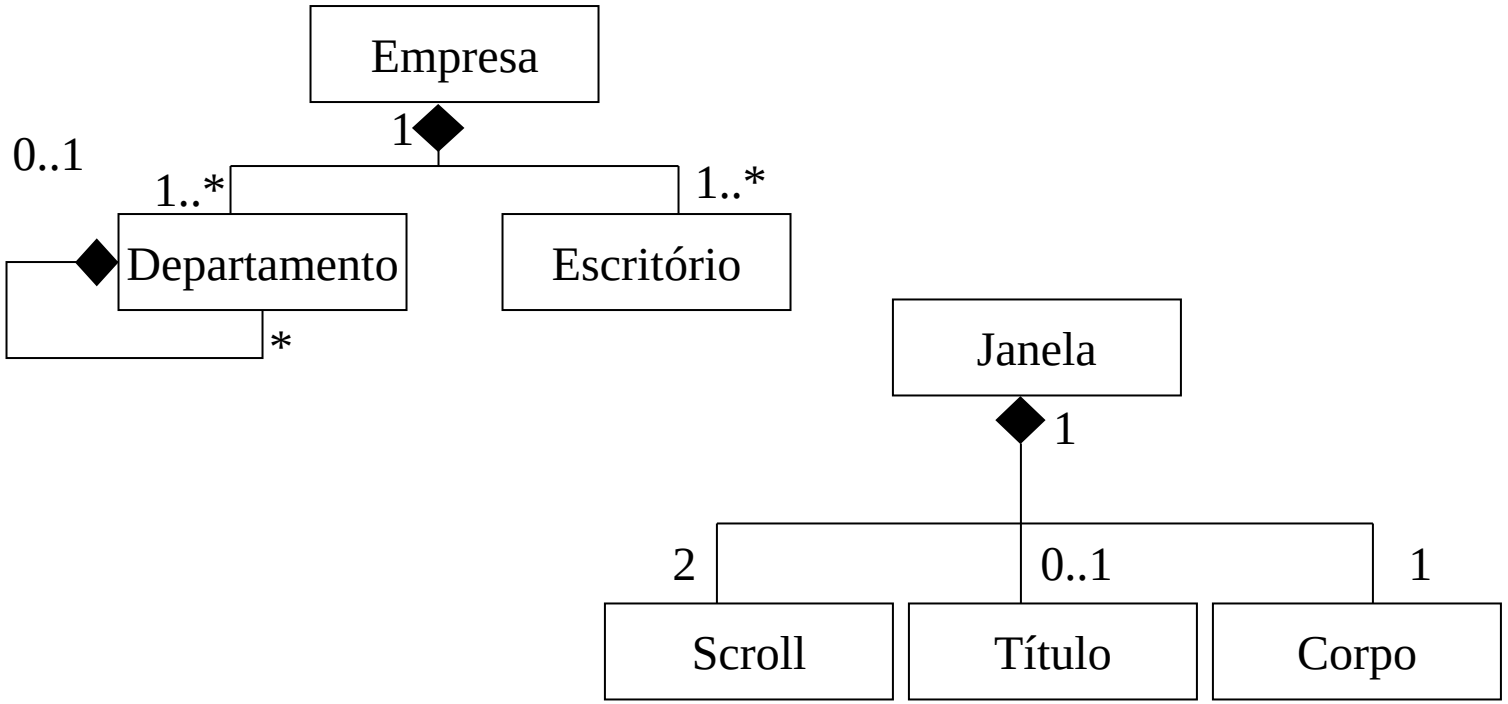
Relacionamento - Composição

- 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.

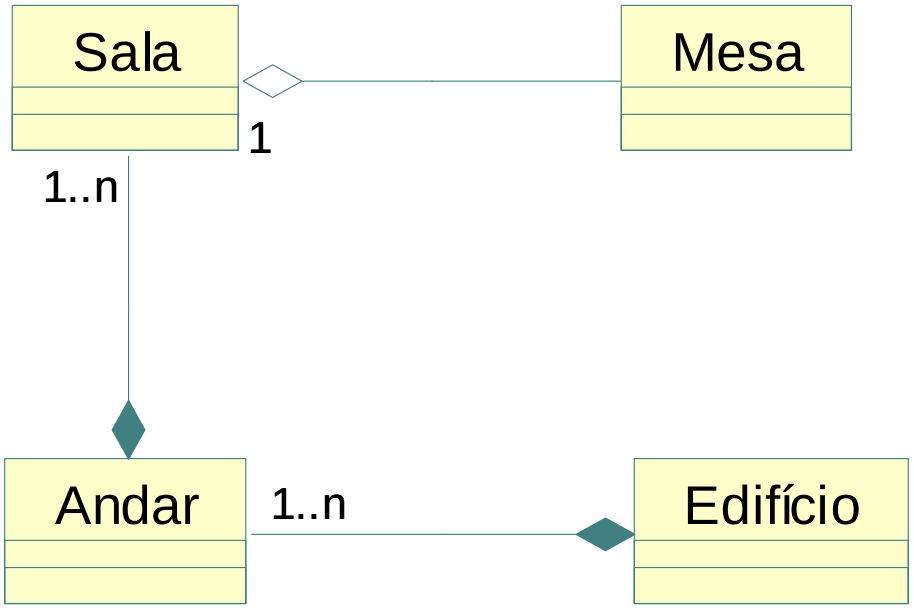


Relacionamento - Composição

- Outros exemplos:

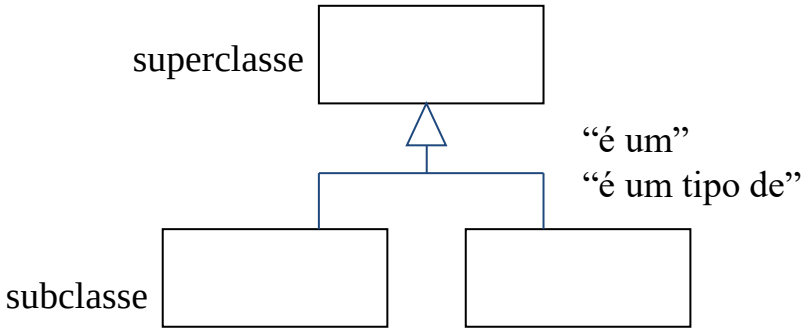


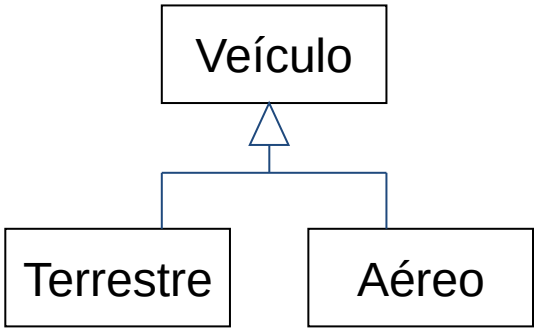
Agregação x Composição



Relacionamento - **Generalização**

- É um relacionamento **entre itens gerais** (superclasses) e **itens mais específicos** (subclasses).





Relacionamento - Dependência

- É o relacionamento mais **informal** do diagrama de classes.
- Significa que a classe (cliente) depende **de alguma forma** da outra (fornecedor).
  - A classe que depende (cliente) **não contém** uma instância da classe de que se depende (fornecedor).
  - Uma mudança na interface da classe da qual se depende (fornecedor) irá **acarretar uma mudança** na classe que depende (cliente).
    - Exemplo (correto):
      - A classe cliente recebe como parâmetro de um de seus métodos a classe fornecedor.
      - A classe cliente cria uma instância da classe fornecedor em um de seus métodos.
    - **Exemplo (errado)**:
      - A classe cliente declara uma instância da variável fornecedor. **Neste caso deve-se utilizar associação!**
- Podemos construir um **diagrama de classes apenas com relacionamentos de dependência?**
  - Teoricamente sim, mas o leitor (do diagrama) precisaria de mais informações para determinar exatamente a relação, fazendo com que o diagrama perca o sentido.

Como construir um diagrama de classes

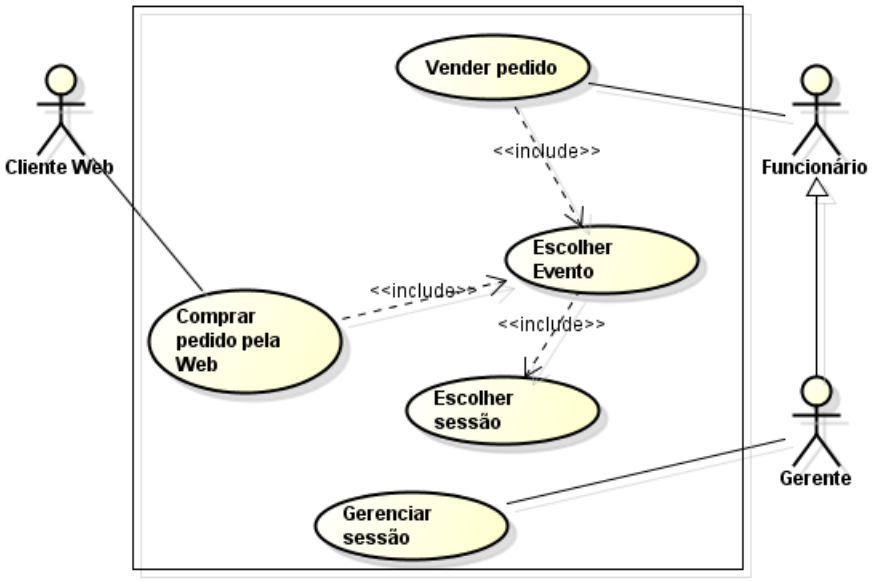
- **Identificar todas as classes** que participam da solução em software;
- Identificar os **relacionamento entre as classes** listadas
  - Associação
    - Agregação
    - Composição
  - Dependência
- Adicionar **nome e sentido aos relacionamentos**;
- Adicionar **multiplicidade aos relacionamentos**;
- Adicionar **atributos às classes**, respeitando os relacionamentos identificados;
- Adicionar **visibilidade dos atributos**
  - Protegido, público ou privado
- **Adicionar métodos às classes**, respeitando os relacionamentos identificados.;
- Adicionar **visibilidade dos métodos**
  - Protegido, público ou privado

Exemplo – Sistema de Venda de Ingressos

- O Estabelecimento XYZ deseja informatizar seu sistema de venda de ingressos:
  - O estabelecimento exibe filmes e peças de teatro
  - O **cliente** pode comprar ingressos pela Web.
  - O **cliente** pode comprar ingressos no estabelecimento através de um **funcionário**.
  - As peças e filmes são exibidos em salas com horários (sessões) específicas.
  - O cliente pode comprar combo de produtos ou produtos individuais tanto pela Web quanto no estabelecimento (através de um **funcionário**).
  - As compras são realizadas através de pedidos, que podem conter produtos, combos e ingressos.
  - As salas possuem assentos numerados.
  - O gerente executa as mesmas ações do funcionário e gerencia sessões.

Exemplo

- Estabelecimento XYZ



Exemplo

- Descrição do Caso de Uso "Comprar Ingresso "
  - Esse caso de uso se inicia quando o cliente acessa o sistema Web do estabelecimento e informa seu login e senha.
  - O sistema apresenta lista com filmes e peças disponíveis.
  - O cliente escolhe o filme ou peça que deseja assistir.
  - O sistema apresenta as sessões do evento escolhido.
  - O cliente escolhe a sessão desejada.
  - O sistema apresenta o mapa de assentos da sala onde a sessão ocorrerá.
  - O cliente escolhe o assento desejado.
  - O sistema emite o pedido para o cliente.

Exemplo Cinema/Teatro

- Identificando as classes

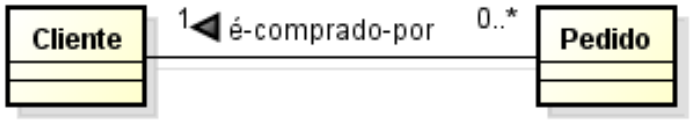
Exemplo

- Identificando 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 Cinema/Teatro

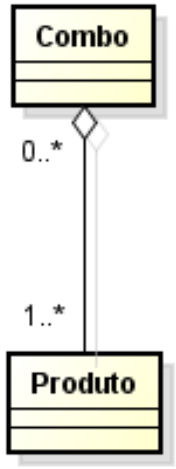
- Associação
  - Um pedido é vendido pelo funcionário do cinema.

  - Um pedido é comprado pelo cliente da Web.



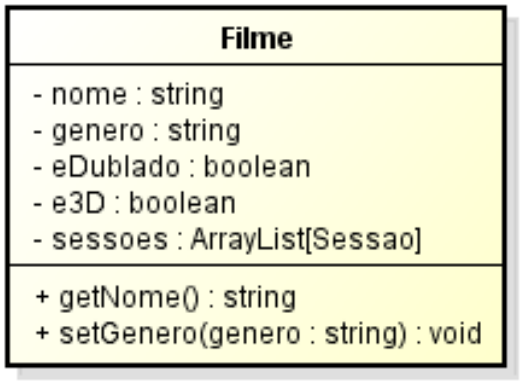
Exemplo Cinema/Teatro

- Agregação
  - Um combo é composto de vários produtos
  - Um produto não precisa, necessariamente, pertencer a um combo



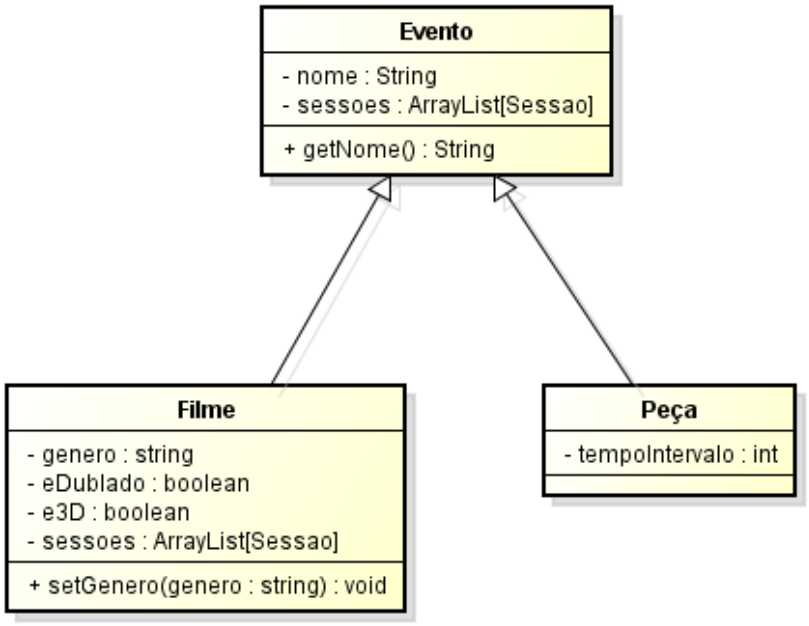
Exemplo Cinema/Teatro

- Identificando atributos
  - Os atributos podem ser encontrados examinando-se as descrições dos casos de uso e também pelo conhecimento do domínio do problema.
    - Os Filmes tem nome, gênero (comédia, suspense, etc.), podem ser dublados ou legendados e 3D ou normal.
    - Cada filme tem sessões onde são exibidos



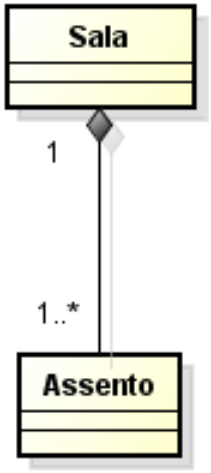
Exemplo Cinema/Teatro

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



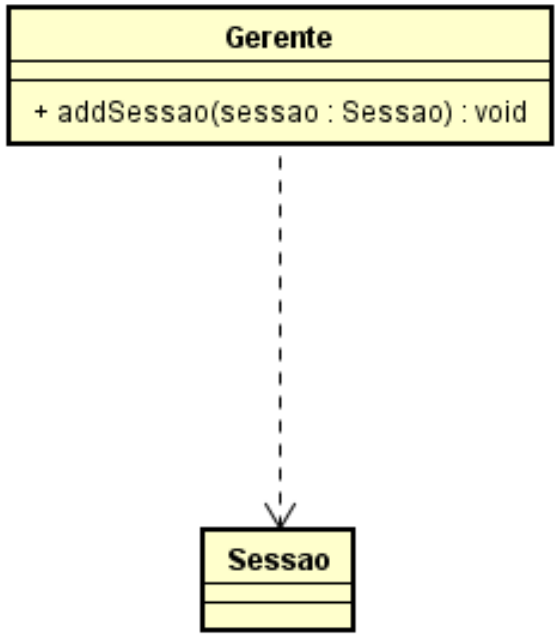
Exemplo Cinema/Teatro

- Composição
  - Uma sala possui assentos identificados
    - Assentos não fazem sentido sem sala.

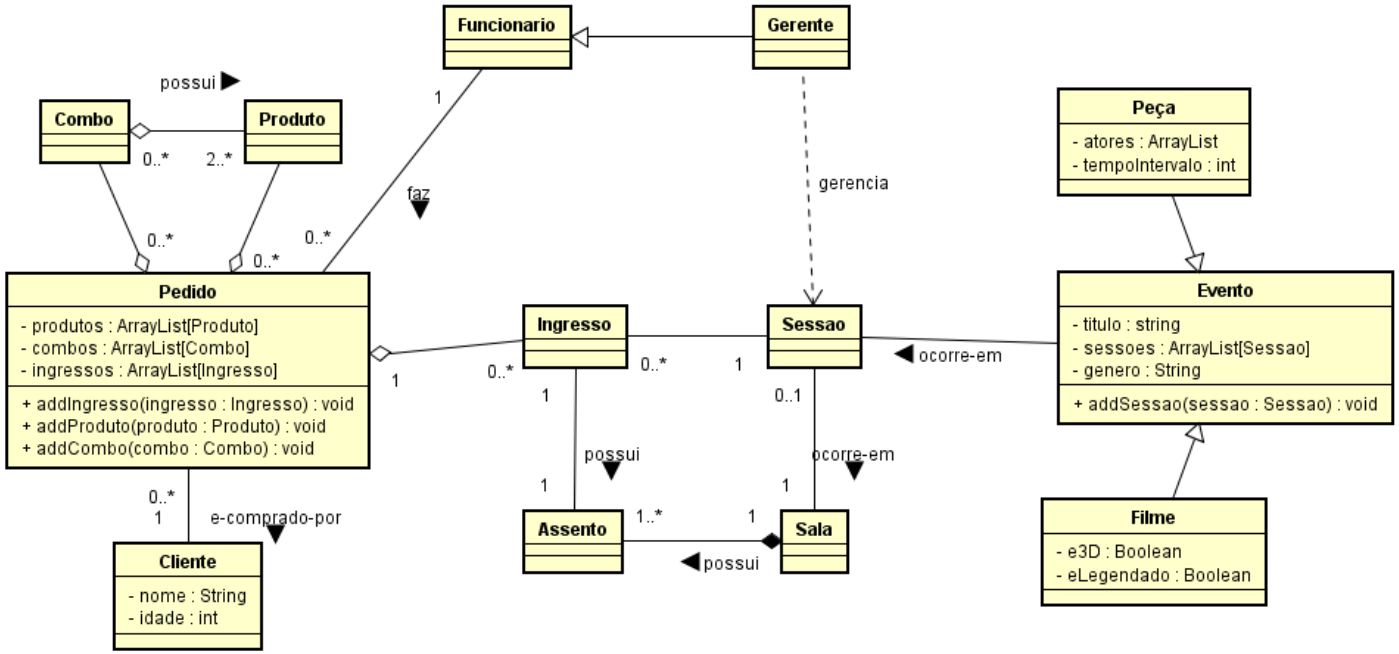


Exemplo Cinema/Teatro

- Dependência
  - A Gerente possui um método (addSessao) que utiliza a classe Sessao como parâmetro



Exemplo Cinema/Teatro



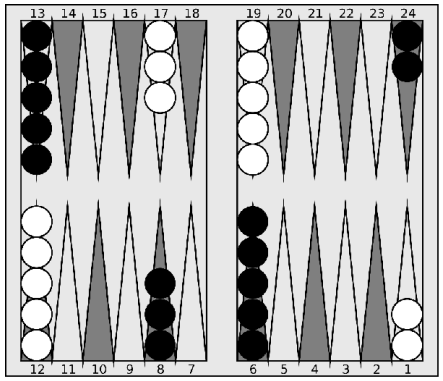
Exercício – Sistema para jogos de tabuleiro

- A princípio, o sistema irá contemplar as seguintes modalidades de jogos de tabuleiro: xadrez, damas.
- Os usuários podem começar a jogar de três formas distintas: iniciar uma partida, procurar por uma partida ou retomar partida pausada.
- Existem dois perfis distintos de usuário: sênior e iniciante.
  - Apenas um usuário sênior pode iniciar uma partida. Usuários iniciantes podem apenas procurar por partidas ou retomar uma partida pausada.
- Ao iniciar uma partida a partir de solicitação de um usuário sênior, o sistema organiza o tabuleiro, incluindo as peças necessárias, e inicia o jogo de acordo com a modalidade escolhida.
- Ao iniciar a partida, o usuário pode selecionar se o seu adversário será o computador (uso de inteligência artificial) ou outro jogador. O jogador computador tem associado a ele seu nível de dificuldade.
  - Caso o usuário escolha jogar contra um computador, ele deve informar o nível de dificuldade que deseja enfrentar.
  - Caso a escolha do usuário seja por um outro jogador, o sistema irá selecionar o adversário automaticamente.
- O sistema deve permitir que o jogador (seja ele usuário ou computador) realize jogadas. Ao fim de cada jogada, o sistema deve sempre verificar se a partida chegou ao fim (empate ou vitória de um dos lados). Caso a partida tenha se encerrado, o histórico do usuário é atualizado com os dados da partida e um relatório final é emitido.
- O sistema deve permitir que o usuário desista da partida que está jogando. Neste caso, ele deve informar o motivo e seu histórico é atualizado com a desistência.
- O sistema deve permitir que o usuário pause a partida. Neste caso, se o jogador adversário for o computador, a partida é pausada imediatamente. Se o outro jogador for outro usuário, o sistema solicita seu consentimento para pausar a partida. O histórico do usuário é atualizado com a partida pausada.
- O sistema deve permitir que o usuário verifique seu histórico de partidas.
- O administrador do sistema pode realizar as mesmas ações que um jogador sênior e ainda pode visualizar todas as partidas que estão sendo realizadas no momento.
- Observações:
  - Uma modalidade é composta de peças, tabuleiro e regras.
  - Um tabuleiro é composto de "casas", que são os locais onde as peças podem ser colocadas pelos jogadores.
  - Cada peça tem movimentos específicos associados a ela.
  - Uma partida é relacionada a uma modalidade e é composta de várias jogadas.
  - Uma partida é realizada por dois ou mais jogadores.
  - Uma jogada pode ser feita por um jogador, ocorre em um tabuleiro e é caracterizada pelo movimento de uma peça de uma casa para outra.
  - Os relatórios contêm as partidas jogadas pelos usuários e os movimentos realizados com as peças durante as partidas.

Exercício – Sistema para jogos de tabuleiro

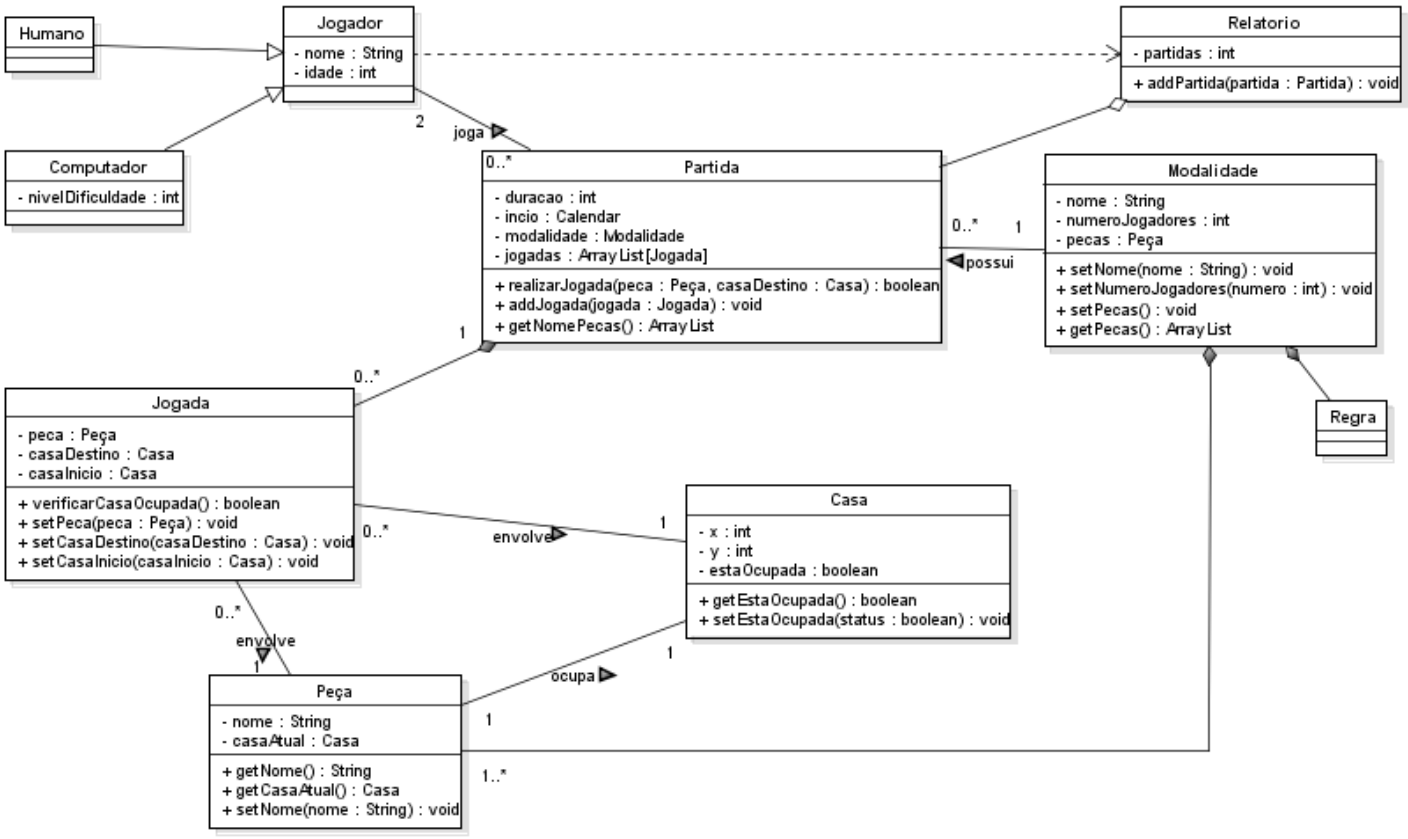
Xadrez

Damas



Gamão

Exercício – Sistema para jogos de tabuleiro



Ferramentas de Modelagem

- StarUML (avaliação por tempo indefinido)
  - https://staruml.io/

- Draw.io
  - https://app.diagrams.net/
  - Web

Referências

- Boock, G. and Rumbaugh, J. The Unified Modeling Language User Guide . Addison-Wesley, 1999
- Arlow, J. and Neustadt, I. UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design, 2nd Edition, The Addison-Wesley Object Technology Series, 2005.
- Rumbaugh, J.; Jacobson, I. and Booch , G. The Unified Modeling Language Reference Manual, 2nd Edition, The Addison-Wesley Object Technology Series, 2004.
- Boock, G.; Rumbaugh, J. and Jacobson, I; Unified Modeling Language User Guide, 2nd Edition, The Addison-Wesley Object Technology Series, 2005.
- Jacobson, I; Boock, G. and Rumbaugh, J., Unified Software Development Process, Addison-Wesley, Janeiro 1999.
- Larman, C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design Prentice-Hall, New Jersey - USA, 1997

INSTITUTO DE CIÊNCIAS EXATAS

DEPARTAMENTO DE COMPUTAÇÃO

Perguntas?