UML – Diagrama de Classes

Profa. Márcia H. Islabão Franco

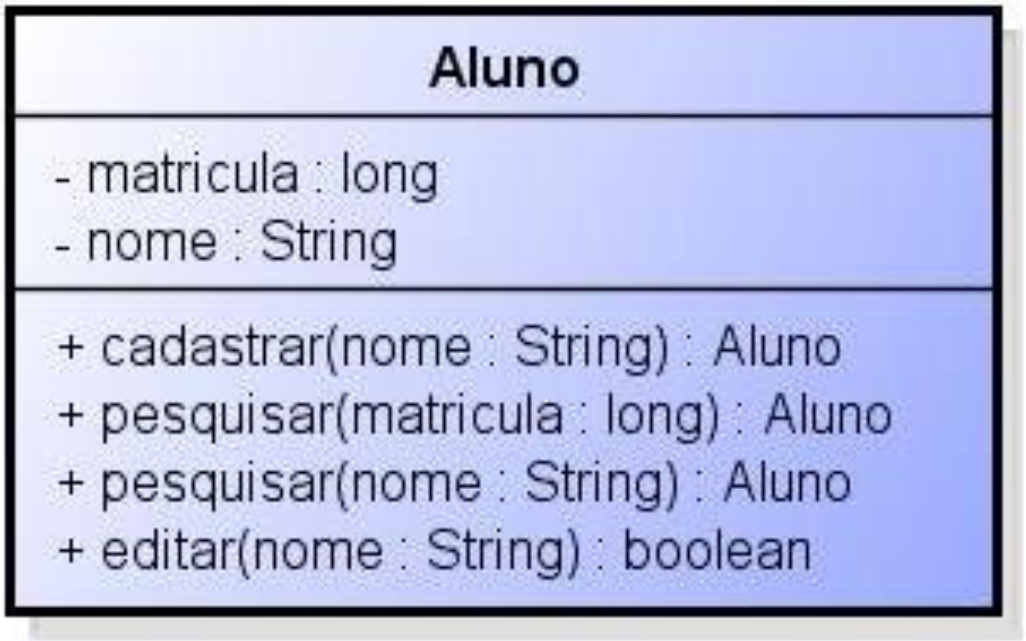
IFRS – Campus Porto Alegre

Sistemas para Internet - Engenharia de Software II

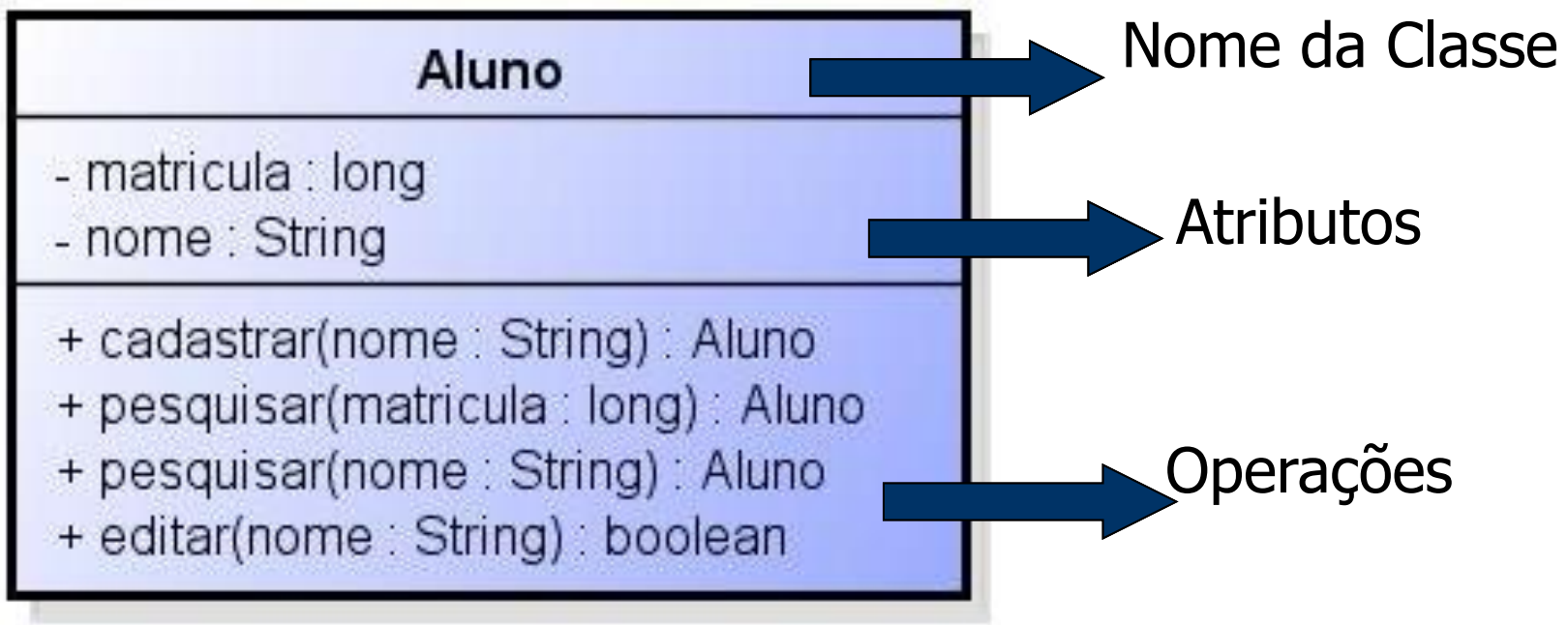
Diagrama de Classes

- Diagrama estrutural da UML que tem como objetivo apresentar uma visão estática de como as classes, que irão compor o sistema, se relacionam, complementam e transmitem informações entre si.
- Um dos diagramas mais importantes e mais utilizados da UML.
- Serve como base para a construção de outros diagramas da UML.
- Geralmente construído na fase de Projeto, podendo também ser utilizado na fase de Análise de Requisitos.

Classe


Aluno
- matricula : long
- nome : String
+ cadastrar(nome : String) : Aluno
+ pesquisar(matricula : long) : Aluno
+ pesquisar(nome : String) : Aluno
+ editar(nome : String) : boolean

Classe


Aluno
- matricula : long
- nome : String
+ cadastrar(nome : String) : Aluno
+ pesquisar(matricula : long) : Aluno
+ pesquisar(nome : String) : Aluno
+ editar(nome : String) : boolean
Nome da Classe
Atributos
Operações

Relacionamento entre Classes

- Representam de que forma as classes compartilham informações e como colaboram na execução das operações do sistema.
- Na UML, o relacionamento entre as classes determina diferentes vínculos entre os objetos. Esses vínculos são determinados pelo tipo de relacionamento, que podem ser:
  - Associação
    - Agregação
    - Composição
  - Generalização
  - Dependência
  - Realização

Associação Binária

- A associação binária é um tipo de relacionamento que indica a existência de um vínculo entre os objetos de uma classe com objetos de outra classe.

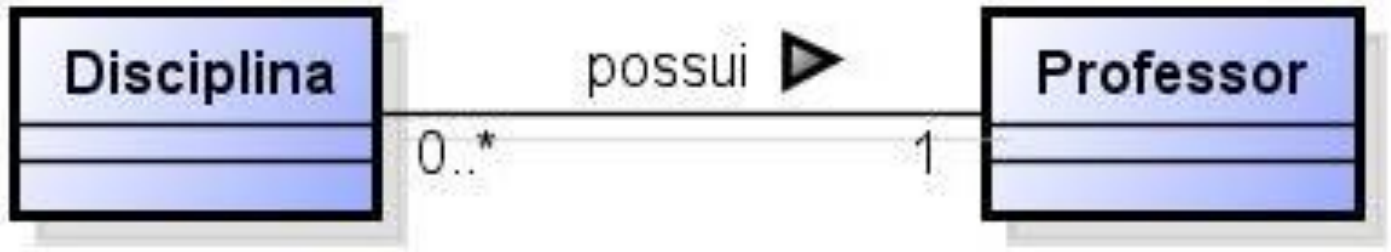
Multiplicidade

- A **multiplicidade** indica o número, mínimo e máximo, de objetos que podem estar associados.

Multiplicidade

1..1 ou 1	**Um e somente um** Um objeto da classe de origem relaciona-se com um e somente um objeto da classe de destino. Em muitos diagramas esse tipo de multiplicidade é omitido nos relacionamentos.
1..*	**No mínimo 1 e no máximo muitos** Um objeto da classe de origem relaciona-se com um ou vários objetos da classe destino.
0..* ou *	**No mínimo nenhum e no máximo muitos** Um objeto da classe de origem pode ou não se relacionar com objetos da classe de destino.
0..1	**No mínimo nenhum e no máximo um** Um objeto da classe de origem relaciona-se com um objeto da classe de destino ou com nenhum.
m..n	Faixa de valores que pode ser estabelecida, por exemplo **2..7**. Neste caso, um objeto da classe de origem relaciona-se com pelo menos dois e no máximo sete objetos da classe de destino.

Exemplo

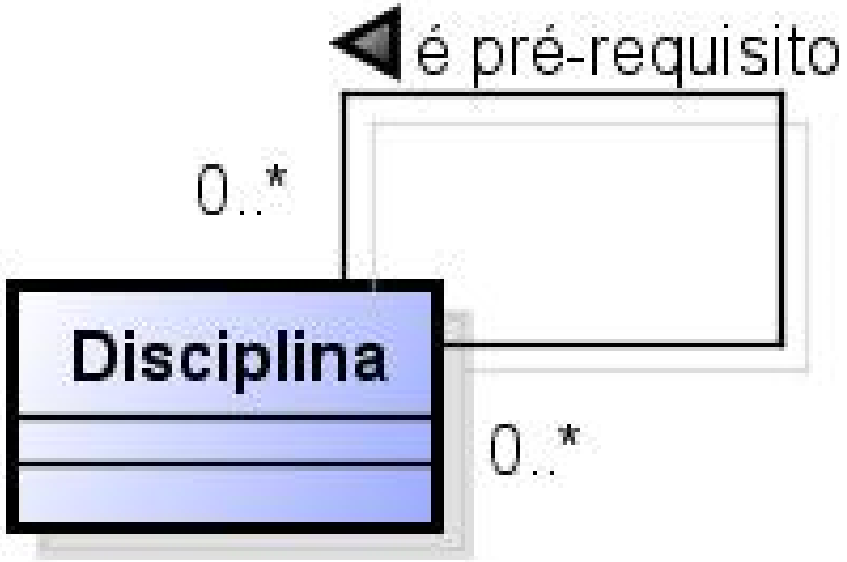


De acordo com a multiplicidade representada, um professor pode ministrar nenhuma ou várias disciplinas e uma disciplina é ministrada somente por um professor.

Por exemplo, podemos instanciar da classe Professor os objetos “Márcia Franco” e “Tanisi Carvalho” e da classe Disciplina podemos instanciar os objetos “Engenharia de Software I”, “Engenharia de Software II” e Banco de Dados II. Dessa forma, no cenário atual, o objeto “Márcia” possui vínculos com os objetos “Engenharia de Software I” e “Engenharia de Software II e o objeto “Tanisi” tem vínculo com o objeto “Banco de Dados II”.

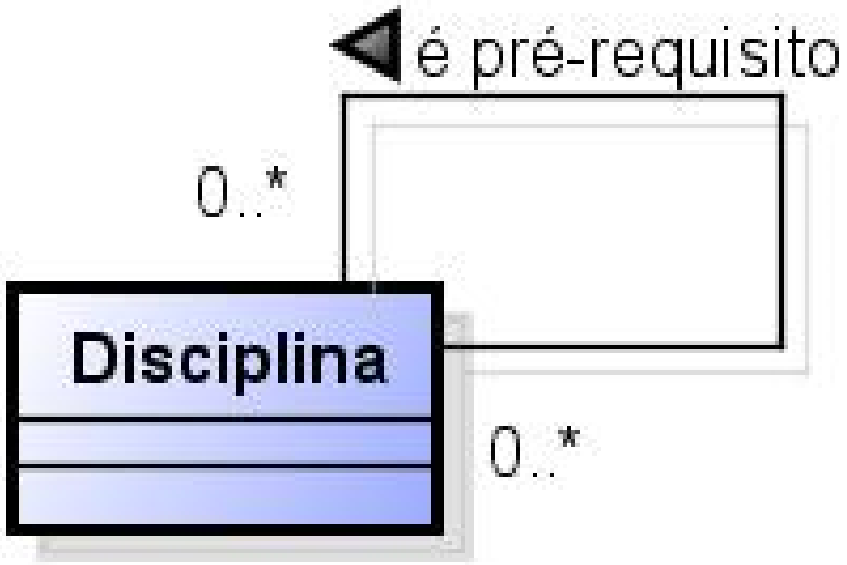
Associação Unária

- Usada quando se pretende representar a existência de um vínculo entre objetos da mesma classe.



Associação Unária

- Usada quando se pretende representar a existência de um vínculo entre objetos da mesma classe.



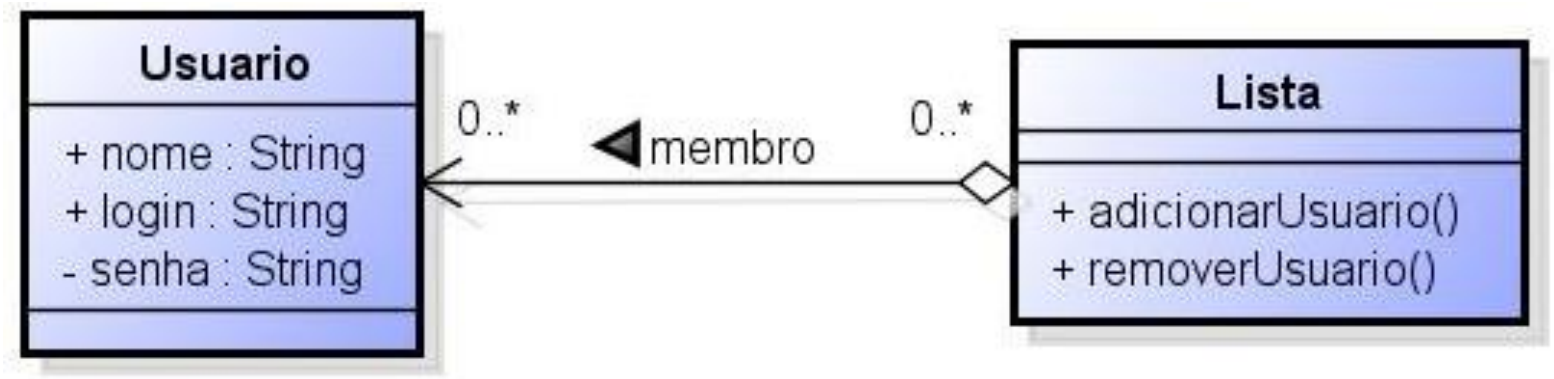
Por exemplo, o objeto "Engenharia de Software I", instanciado da classe Disciplina, é pré-requisito do objeto "Engenharia de Software II" também instanciado da classe Disciplina.

Agregação

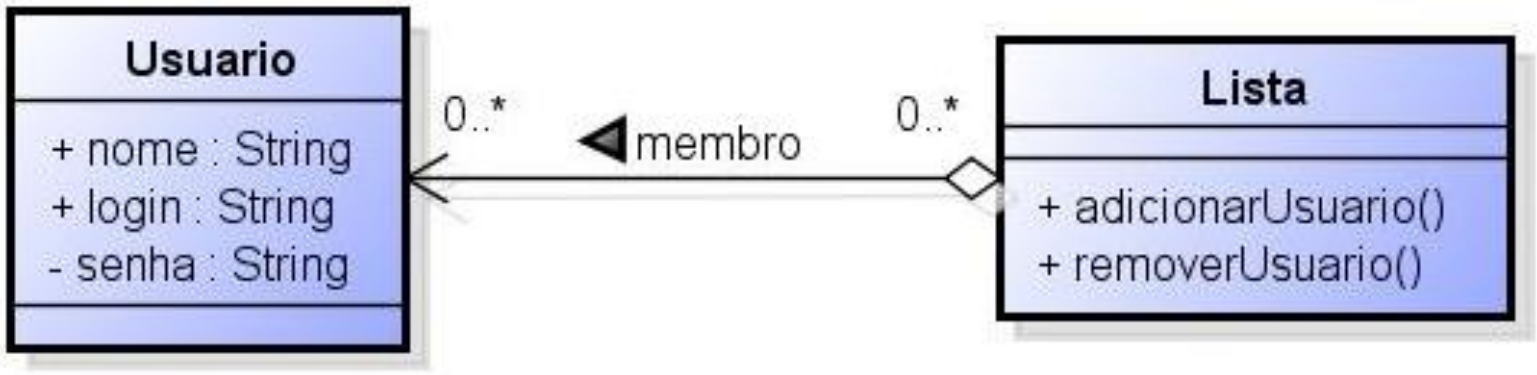
- O relacionamento de agregação é um tipo especial de associação, que é utilizado quando se deseja representar vínculos do tipo “todo/parte” entre objetos.

Agregação

- Na UML usamos o relacionamento de agregação, representado pelo losango, quando pretendemos mostrar que as informações de um objeto, denominado objeto-todo, precisam ser complementadas pelas informações de um ou mais objetos, ditos objetos-parte.



Exemplo

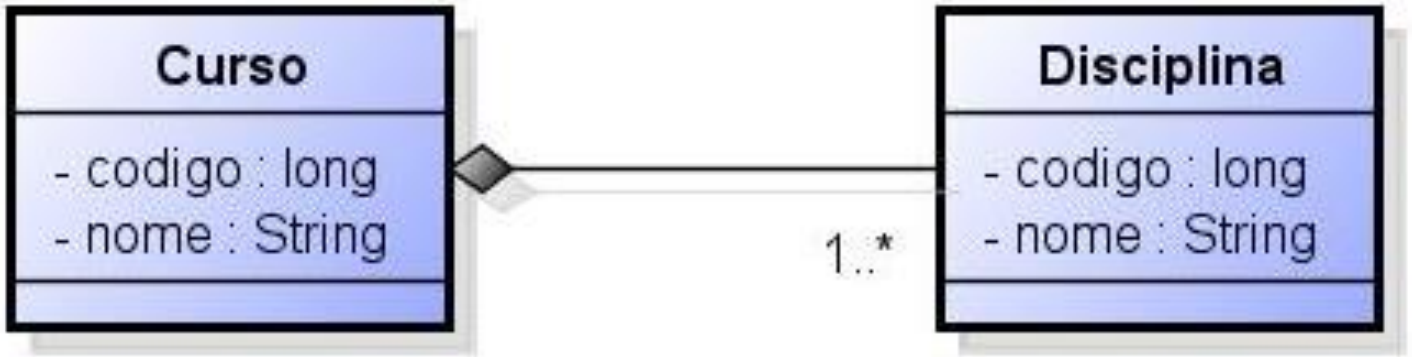


Por exemplo, uma instância da classe Lista poderá conter como membro nenhuma ou muitas instâncias da classe Usuario como suas partes. Uma instância da classe Usuario poderá pertencer a nenhuma ou muitas instâncias da classe Lista.

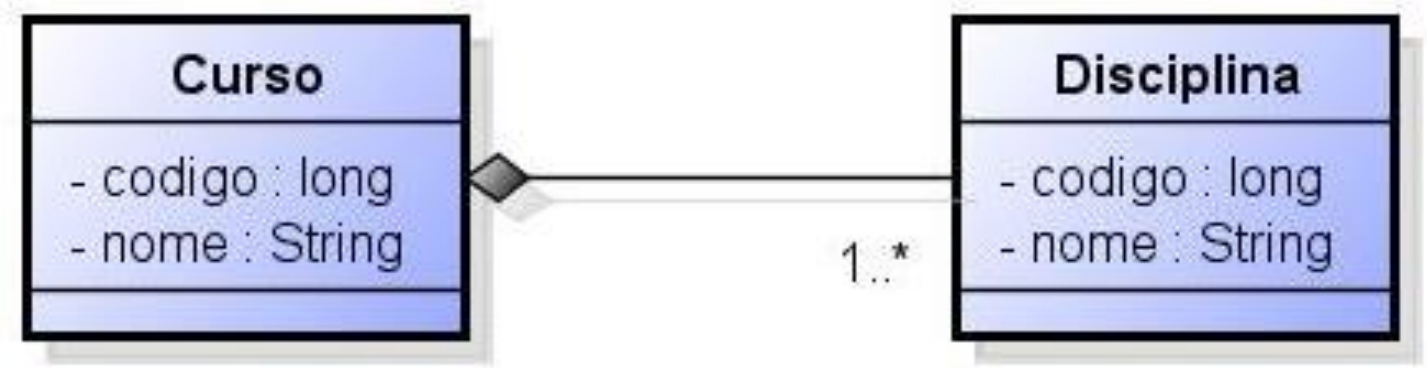
Caso o objeto-todo deixe de existir, suas partes permanecerão existindo. Por exemplo, se um objeto da classe **Lista** for excluído, os objetos da classe **Usuario** continuarão existindo no sistema.

Composição

- A composição é um tipo especial de agregação, que apresenta um vínculo mais forte entre o objeto-todo e os objetos-parte.
- Representado pelo losango preenchido.



Exemplo



Neste exemplo, se um curso (objeto-todo) deixar de existir, suas disciplinas (objetos-parte) também deixarão.

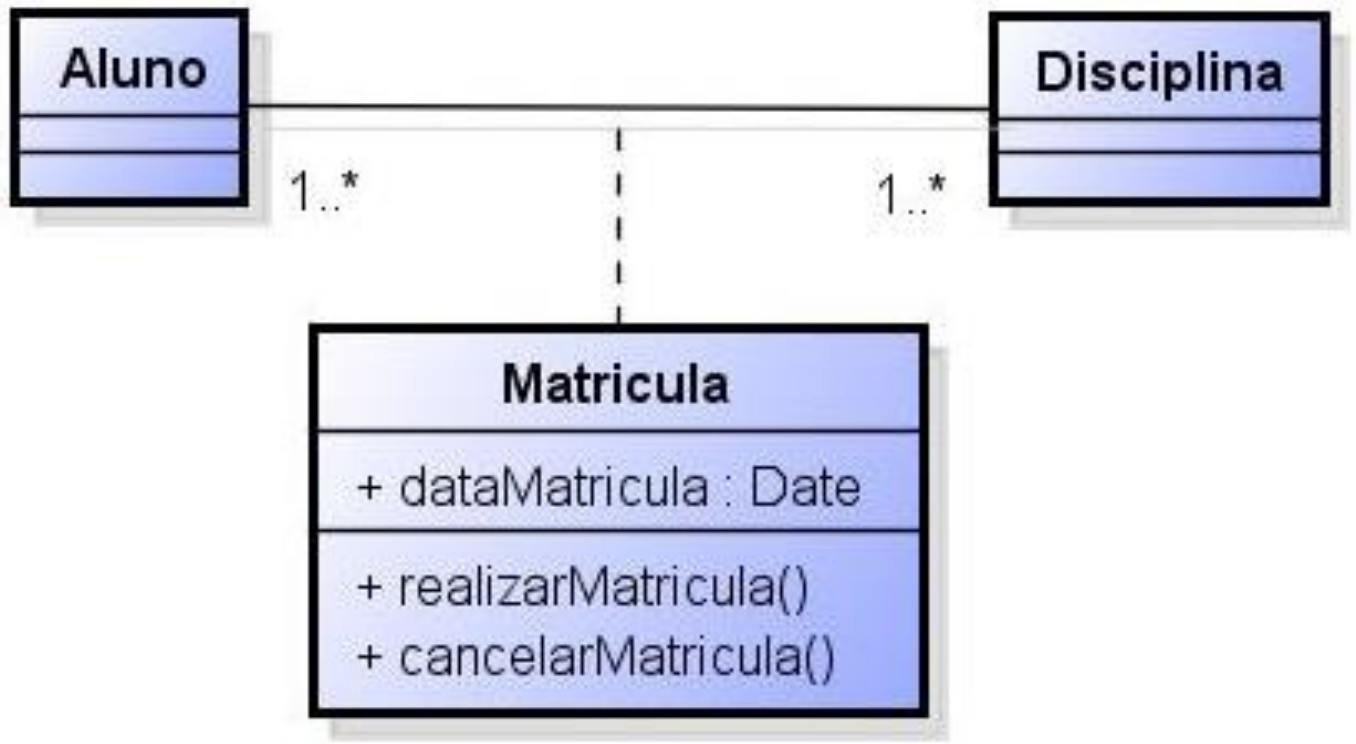
Diferentemente da agregação, na composição os objetos-parte estão associados a um único objeto-todo. Por exemplo, as disciplinas de Engenharia de Software I e Banco de Dados II pertencem ao curso de Sistemas para a Internet. Se o curso deixar de ser ofertado, as disciplinas deixam de existir.

Classe Associativa

- As classes associativas derivam das associações que possuem multiplicidade “muitos” (*) em todas as suas extremidades.

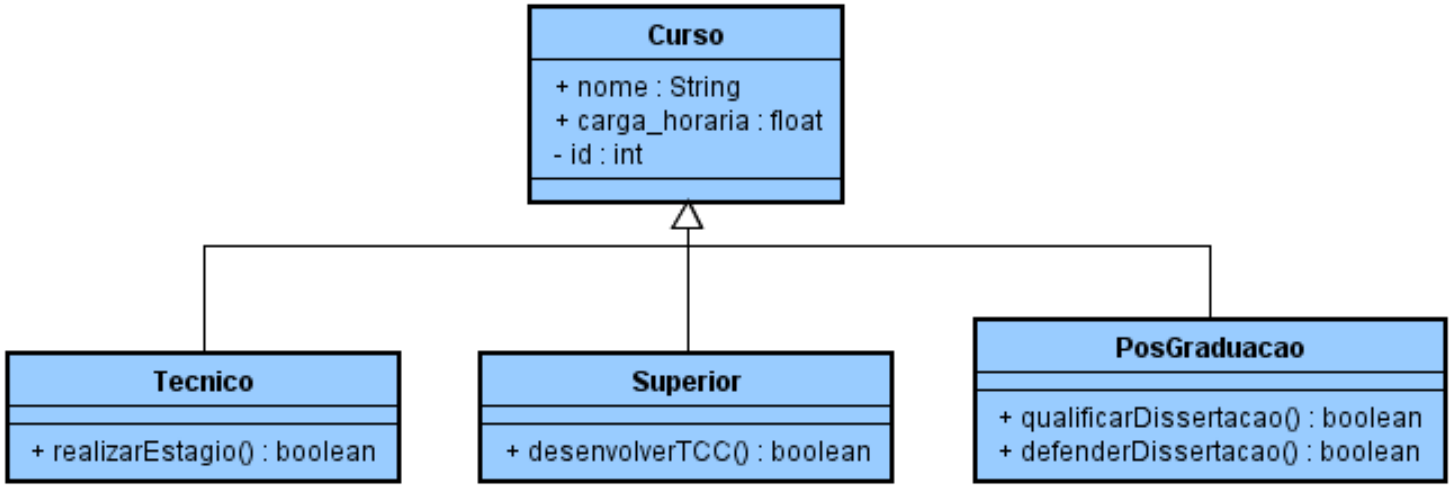
- As classes associativas devem ser utilizadas quando existir atributos relacionados a essa associação e estes não poderem ser armazenados em nenhuma das classes envolvidas.

Classe Associativa

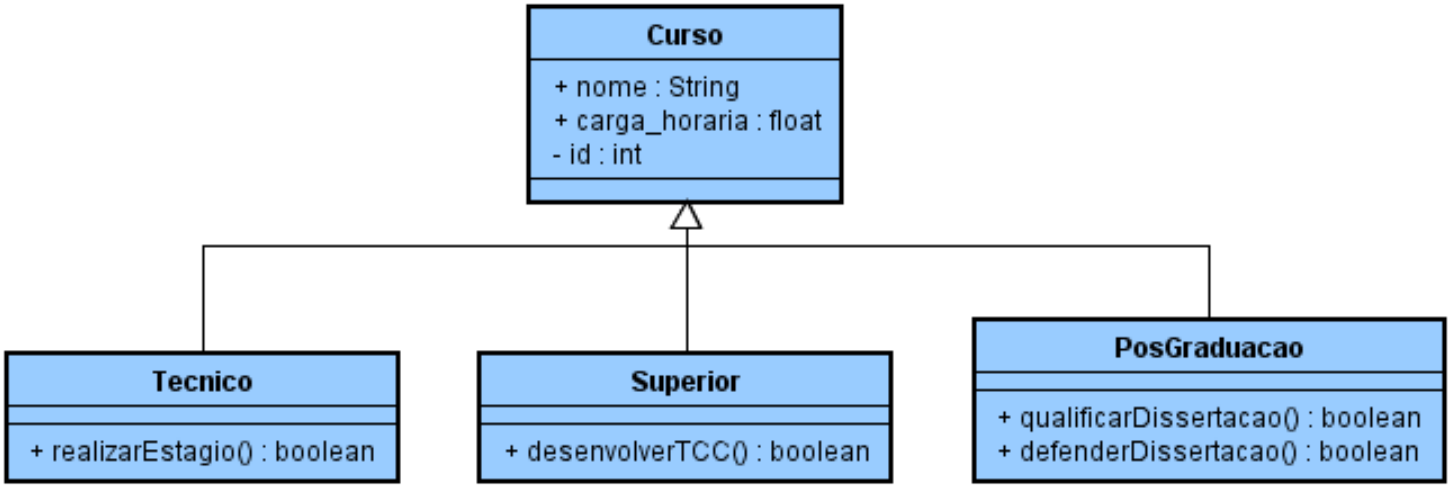


Generalização

- Na UML a generalização (ou especialização) é um tipo de relacionamento utilizado quando classes de um sistema possuem atributos e operações muito semelhantes.



Exemplo



Através da generalização é possível definir uma ou mais classes a partir de uma classe existente, reaproveitando seus atributos e operações. Por exemplo, as classes Tecnico, Superior e PosGraduacao herdam os atributos e operações da classe Curso.

Estereótipos

- Utilizados para indicar que determinados componentes do sistema executam funções diferentes.
- A UML apresenta diversos estereótipos, além de permitir a criação de novos.
- Os estereótipos podem ser utilizados em todos os diagramas, porém é mais utilizado nos diagramas de classes.

Principais Estereótipos

- <<entity>>
  - Utilizado quando se deseja indicar que a classe armazena informações sobre uma entidade.

Principais Estereótipos

- <<boundary>>
  - Utilizado em classes que servem de comunicação entre os atores externos e o sistema. Geralmente é associado à própria interface do sistema.

Principais Estereótipos

- <<control>>
  - Representa classes que servem de intermediárias entre as classes <<boundary>> e os demais componentes do sistema.

Principais Estereótipos

- <<control>>
  - Representa classes que servem de intermediárias entre as classes <<boundary>> e os demais componentes do sistema.

Os objetos desta classe são responsáveis por interpretar os eventos ocorridos na classe <<boundary>> e retransmiti-los aos objetos da classe <<entity>>

Diagrama de Classes: Dependência

- Relacionamento utilizado quando uma classe depende de atributos ou de operações de outra classe para poder executar suas operações.

Diagrama de Classes: Dependência

- Relacionamento utilizado quando uma classe depende de atributos ou de operações de outra classe para poder executar suas operações.

A classe **JanelaUsuario** depende da operação **retornarAmigos(): List<Usuario>,** da classe **ConsultaAmigos,** para executar a operação **visualizarAmigos()** e esta depende da operação **retornarAmigos(): List<Usuario>** da classe **Usuario**.

Diagrama de Classes: Dependência

- Relacionamento utilizado quando uma classe depende de atributos ou de operações de outra classe para poder executar suas operações.

A classe **JanelaUsuario** depende da operação **retornarAmigos(): List<Usuario>,** da classe **ConsultaAmigos,** para executar a operação **visualizarAmigos()** e esta depende da operação **retornarAmigos(): List<Usuario>** da classe **Usuario**.

O estereótipo <<boundary>> indica que a classe **JanelaUsuario** serve de comunicação entre os atores externos do sistema e o sistema.

Diagrama de Classes: Dependência

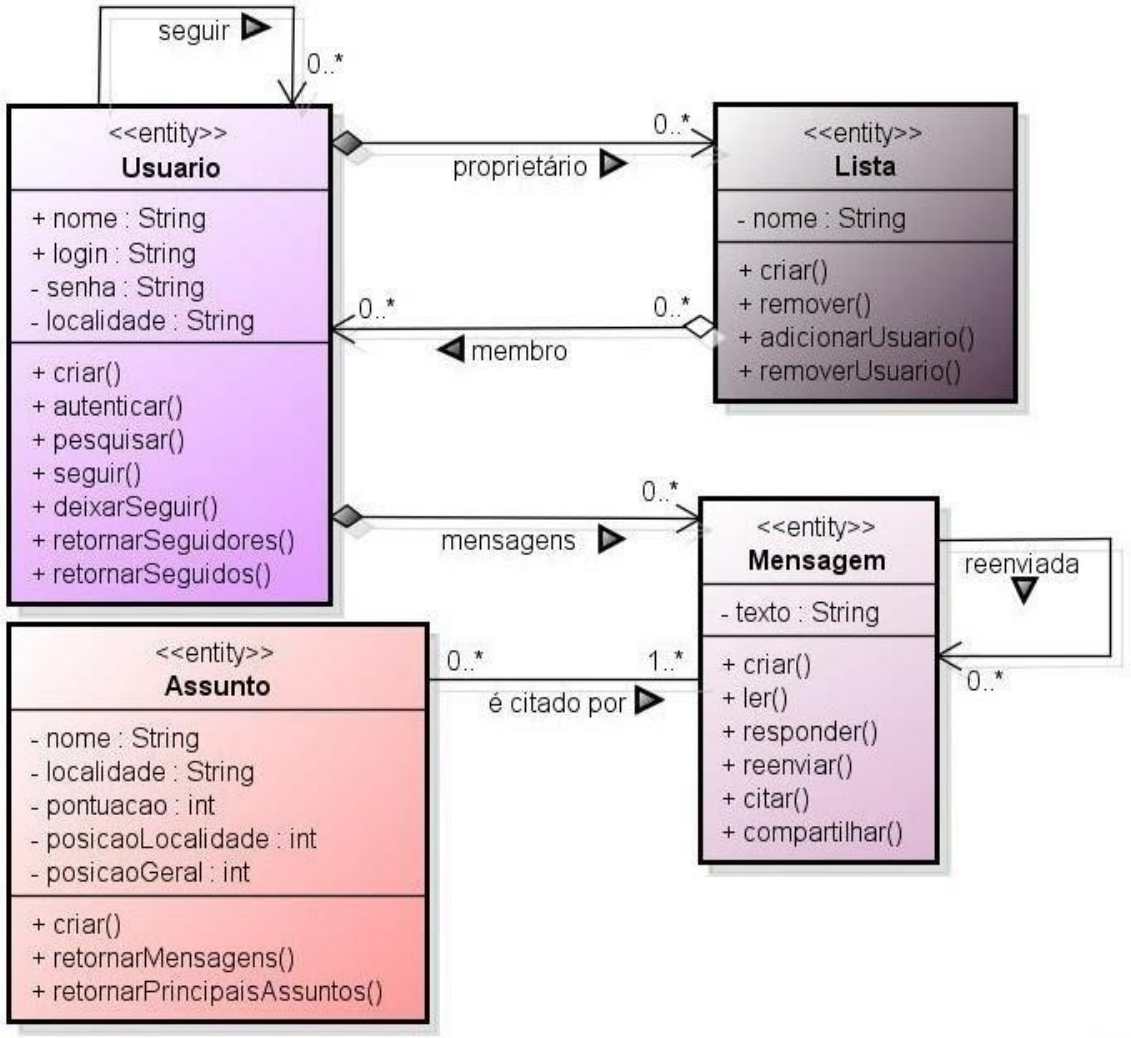
- Relacionamento utilizado quando uma classe depende de atributos ou de operações de outra classe para poder executar suas operações.

A classe **JanelaUsuario** depende da operação **retornarAmigos(): List<Usuario>,** da classe **ConsultaAmigos,** para executar a operação **visualizarAmigos()** e esta depende da operação **retornarAmigos(): List<Usuario>** da classe **Usuario**.

O estereótipo <<boundary>> indica que a classe **JanelaUsuario** serve de comunicação entre os atores externos do sistema e o sistema.

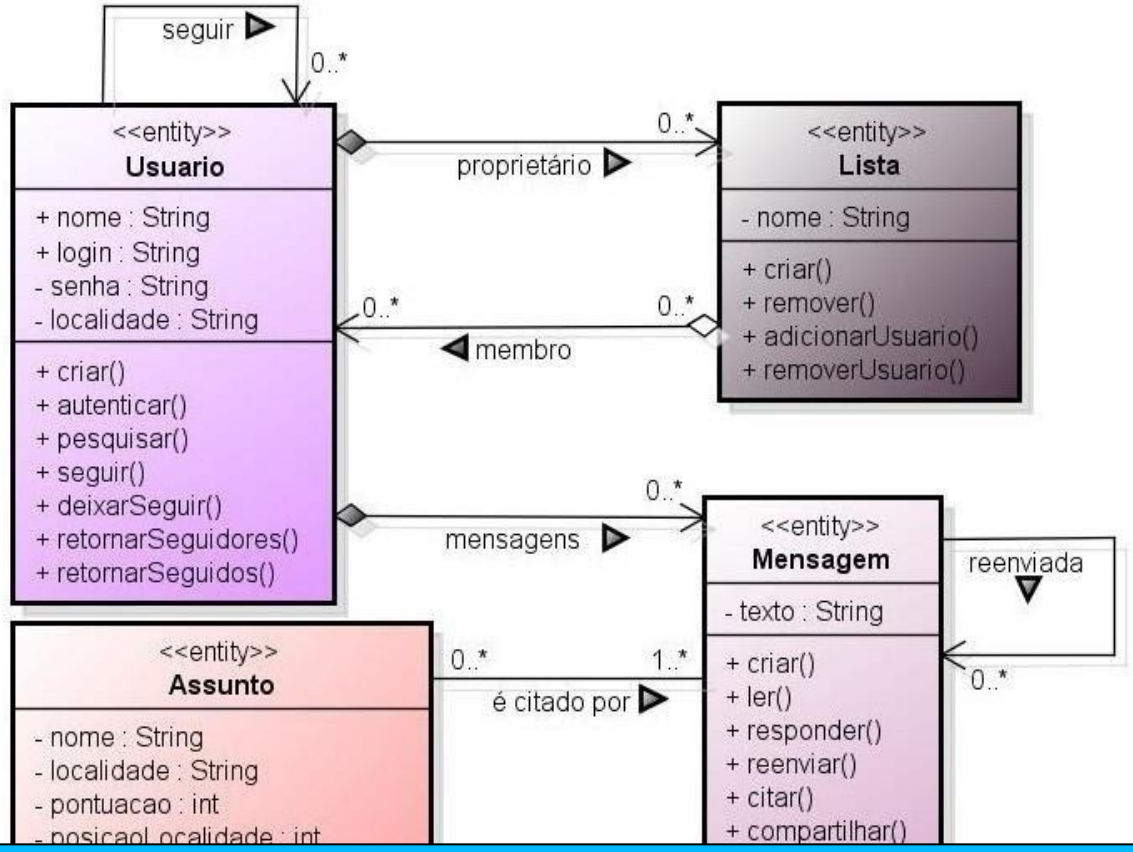
O estereótipo <<control>> indica que a classe **ConsultaAmigos** é responsável por intermediar as classes **JanelaUsuario** e **Usuario.** Os objetos instanciados nesta classe são responsáveis por interpretar os eventos ocorridos sobre os objetos da classe **JanelaUsuario** e retransmiti-los aos objetos da classe **Usuario**.

Exemplo



O relacionamento de composição entre as classes Usuario e Lista indica que um objeto-todo da classe Usuario pode conter ou não muitos objetos-parte da classe Lista. Já um objeto-parte da classe Lista pertence somente a um objeto-todo da classe Usuario. Nesse caso, um usuário é proprietário de nenhuma ou muitas listas, e uma lista pertence somente a um usuário.

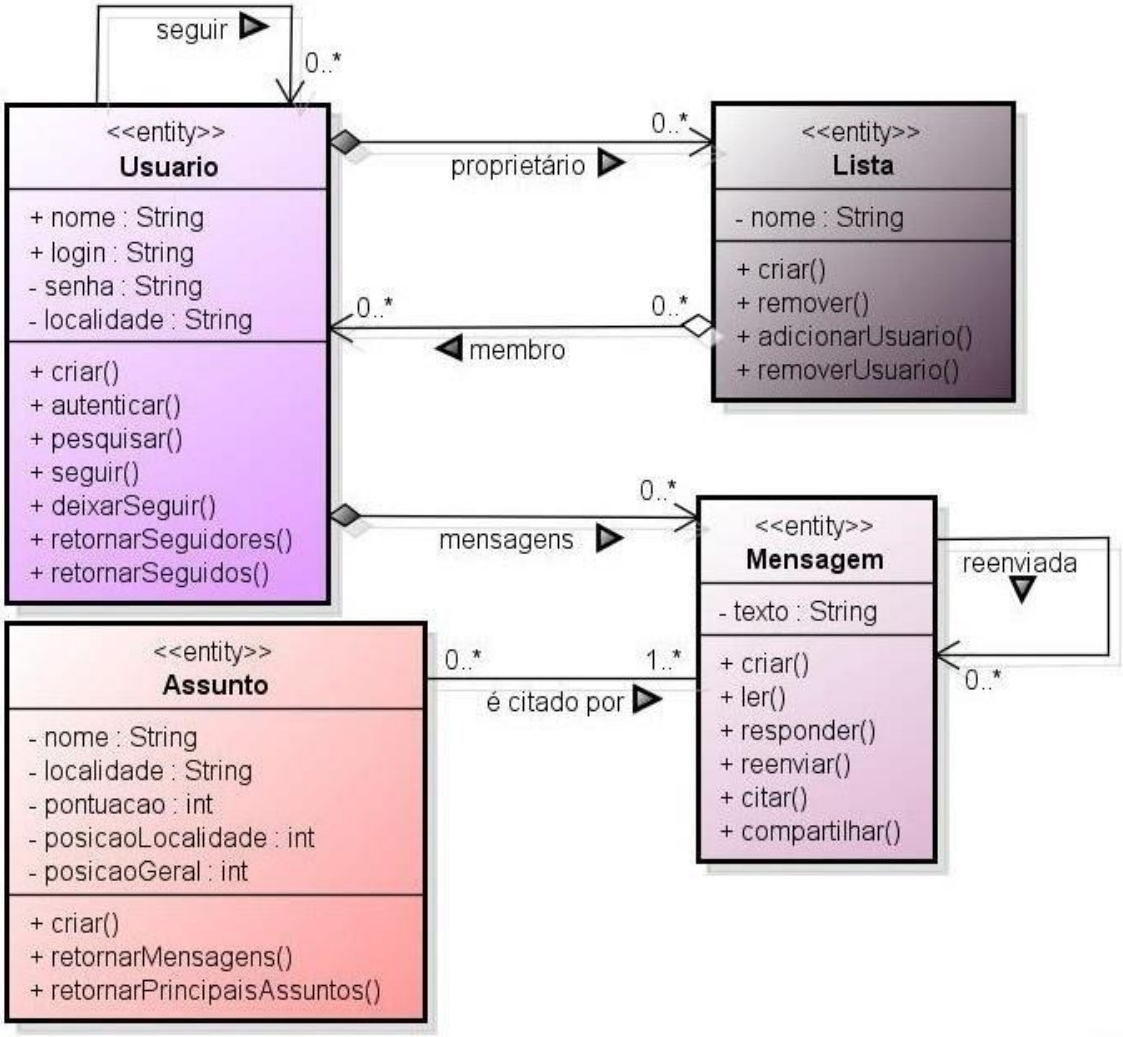
Exemplo



O relacionamento de composição entre as classes Usuario e Lista indica que um objeto-todo da classe Usuario pode conter ou não muitos objetos-parte da classe Lista. Já um objeto-parte da classe Lista pertence somente a um objeto-todo da classe Usuario. Nesse caso, um usuário é proprietário de nenhuma ou muitas listas, e uma lista pertence somente a um usuário.

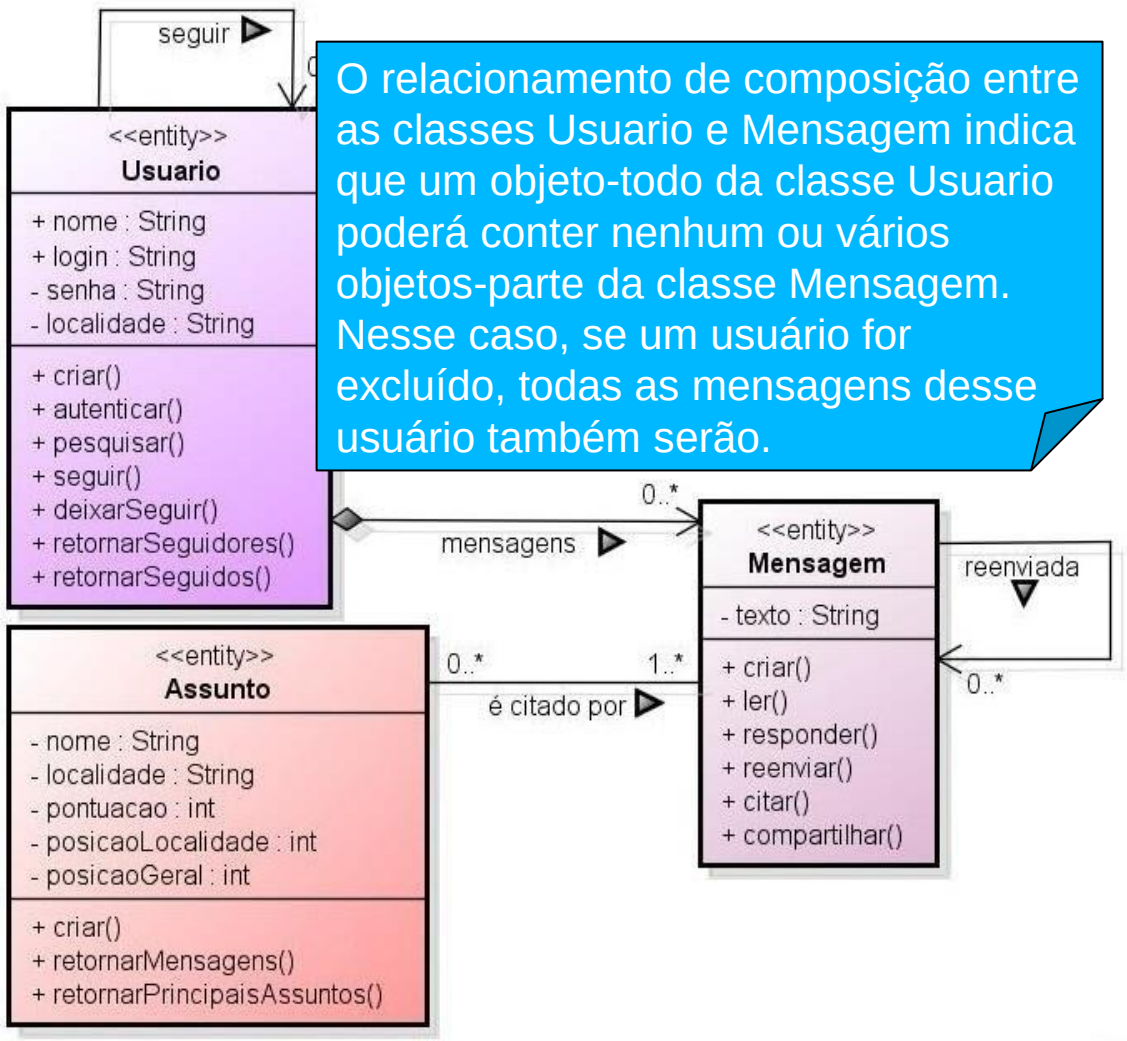
Nota-se ainda um relacionamento de agregação entre as classes Lista e Usuario, em que um objeto-todo da classe Lista poderá conter nenhum ou muitos objetos-parte da classe Usuario. Assim, uma lista poderá ter como membros nenhum ou muitos usuários, assim como um usuário poderá pertencer ou não a muitas listas.

Exemplo



A classe Mensagem representa um texto inserido por um usuário. A classe possui como atributo texto: String, e suas operações são criar(), responsável pela instanciação de um objeto da classe; ler(), possibilita que o usuário leia as mensagens; responder(), cria uma nova mensagem, com texto próprio, relacionada a outra; reenviar(), cria uma nova mensagem relacionada a outra (a nova mensagem não possui texto próprio.); citar(), cria uma nova mensagem, com texto próprio, e inclui um assunto à mensagem, por exemplo, “#case”; compartilhar(), permite enviar a mensagem para outras redes sociais ou por *e-mail*.

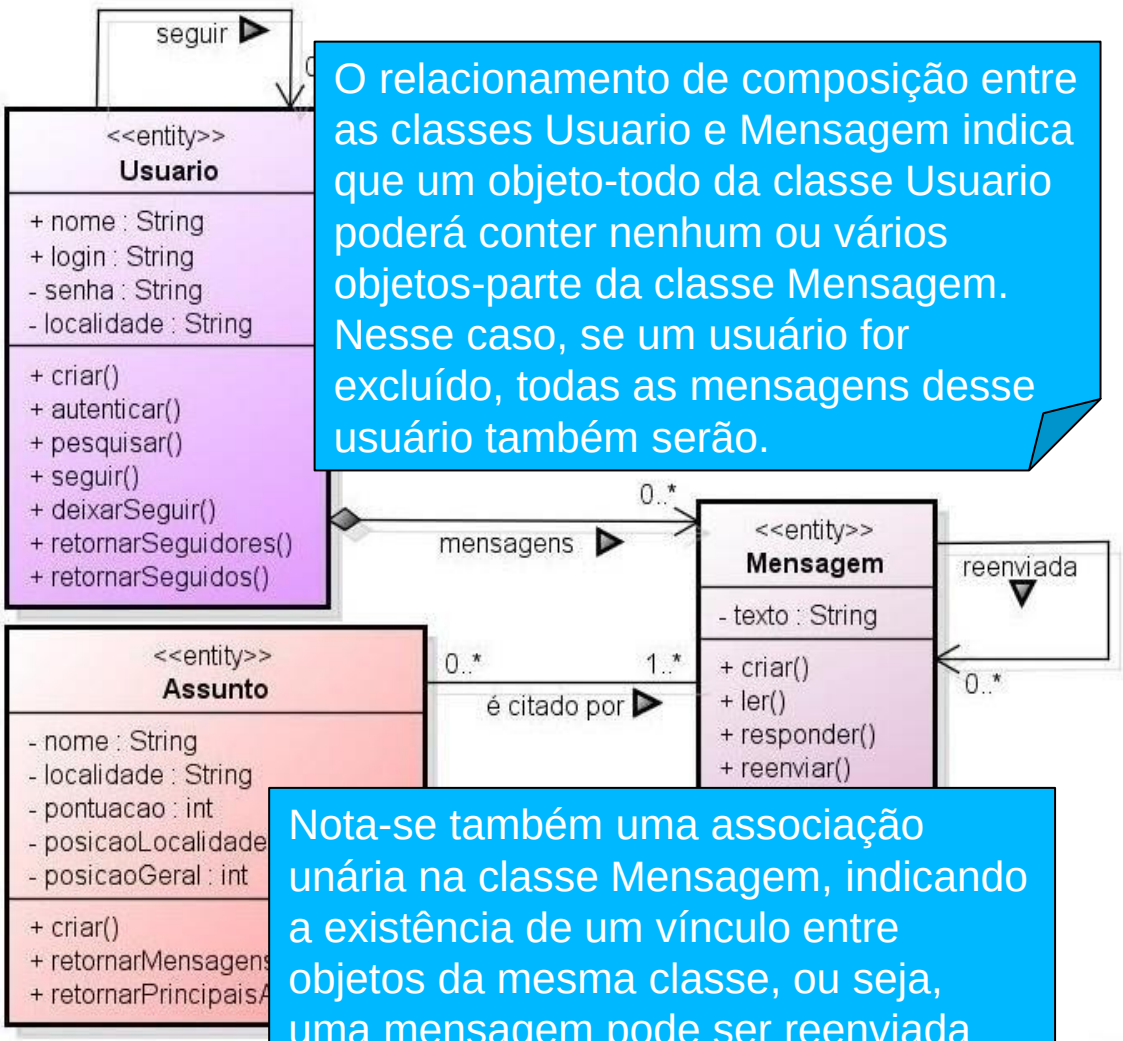
Exemplo



O relacionamento de composição entre as classes Usuario e Mensagem indica que um objeto-todo da classe Usuario poderá conter nenhum ou vários objetos-parte da classe Mensagem. Nesse caso, se um usuário for excluído, todas as mensagens desse usuário também serão.

A classe Mensagem representa um texto inserido por um usuário. A classe possui como atributo texto: String, e suas operações são criar(), responsável pela instanciação de um objeto da classe; ler(), possibilita que o usuário leia as mensagens; responder(), cria uma nova mensagem, com texto próprio, relacionada a outra; reenviar(), cria uma nova mensagem relacionada a outra (a nova mensagem não possui texto próprio.); citar(), cria uma nova mensagem, com texto próprio, e inclui um assunto à mensagem, por exemplo, “#case”; compartilhar(), permite enviar a mensagem para outras redes sociais ou por *e-mail*.

Exemplo



O relacionamento de composição entre as classes Usuario e Mensagem indica que um objeto-todo da classe Usuario poderá conter nenhum ou vários objetos-parte da classe Mensagem. Nesse caso, se um usuário for excluído, todas as mensagens desse usuário também serão.

Nota-se também uma associação unária na classe Mensagem, indicando a existência de um vínculo entre objetos da mesma classe, ou seja, uma mensagem pode ser reenviada por nenhuma ou muitas mensagens.

A classe Mensagem representa um texto inserido por um usuário. A classe possui como atributo texto: String, e suas operações são criar(), responsável pela instanciação de um objeto da classe; ler(), possibilita que o usuário leia as mensagens; responder(), cria uma nova mensagem, com texto próprio, relacionada a outra; reenviar(), cria uma nova mensagem relacionada a outra (a nova mensagem não possui texto próprio.); citar(), cria uma nova mensagem, com texto próprio, e inclui um assunto à mensagem, por exemplo, “#case”; compartilhar(), permite enviar a mensagem para outras redes sociais ou por *e-mail*.

Referências