Diagrama de Classes – UML, Encapsulamento e Relacionamentos entre Classes

Prof. Valdir Pedrinho de Tomin Junior, Eng.

Universidade Federal de Santa Catarina
Centro Tecnológico de Joinville
EMB5631 – Programação III

e-mail: valdir.pedrinho@ufsc.br

25 de agosto de 2018

A UML

O que é?

Linguagem de Modelagem Unificada (do inglês, UML - Unified Modeling Language) é uma linguagem-padrão para a elaboração da estrutura de projetos de software.

A UML como um padrão de diagramação

Em 2000, a UML foi aprovada como padrão pelo OMG (Object Management Group), um consórcio internacional de empresas que define e ratifica padrões na área de Orientação a Objetos.

A UML

Onde pode ser utilizada?

A UML se destina principalmente a sistemas complexos de softwares. Tem sido empregada de maneira efetiva em domínios como os seguintes[1]:

- Sistemas de informações corporativos;
- Serviços bancários e financeiros;
- Telecomunicações;
- Transportes;
- Defesa/Espaço Aéreo;
- Vendas de Varejo;
- Eletrônica médica;
- Serviços distribuídos.

[1] BOOCH, G; RUMBAUGH, J e JACOBSON, I: UML, Guia do Usuário: tradução; Fábio Freitas da Silva, Rio de Janeiro, Campus, 2012.

Diagramas da UML

Tipos de Diagrama

A UML, conforme o OMG, possui 14 tipos de diagramas, divididos em duas grandes categorias: Estruturais e Comportamentais:

Estruturais:

- Classes;
- Objetos;
- Componentes;
- Perfil;
- Implementação;
- Pacotes;
- Estrutura Composta;

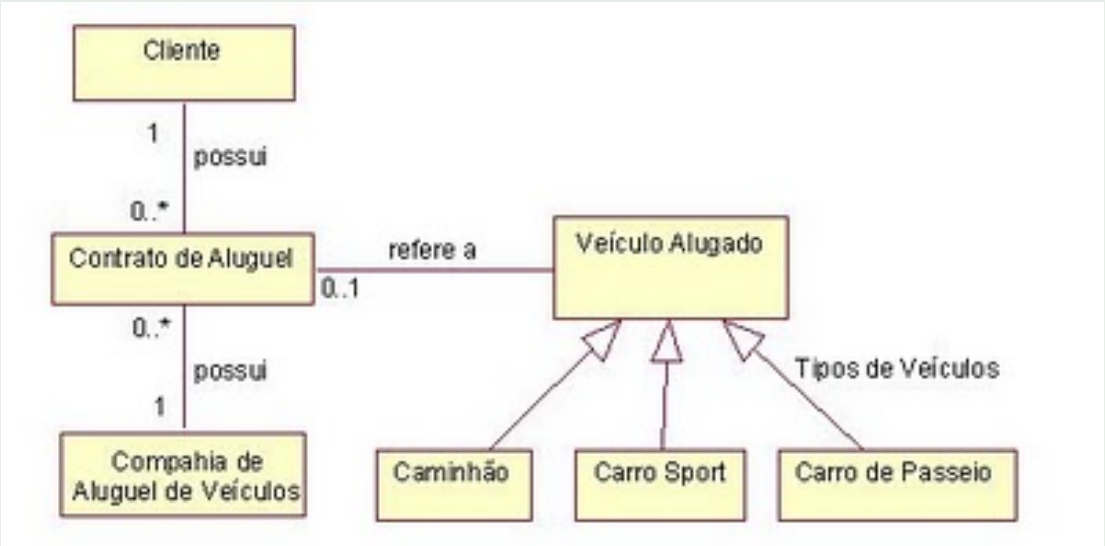
Comportamentais:

- Atividades;
- Caso de Uso;
- Máquina de Estados;
- Sequência;
- Comunicação;
- Visão Geral de Interação;
- Tempo.

Diagramas da UML

Qual o nosso interesse?

Vamos trabalhar com o Diagrama de Classes!



E os outros diagramas?

Este os demais diagramas serão abordados na disciplina de Modelagem de Sistemas

Classes

Nome

Identificador para a classe. Normalmente primeira letra maiúscula e demais minúsculas.

Atributos

Propriedades da classe. Para cada propriedade especifica-se:

- **Nome:** identificador para o atributo;
- **Tipo:** o tipo do atributo (inteiro, real, outra classe etc.);
- **Valor padrão:** valor inicial para o atributo, este é facultativo.

Métodos

Funcionalidades da classe (funções que a classe possui / pode realizar).
Para cada método especifica-se um *assinatura*, composta por:

- **Nome:** identificador para o método;
- **Tipo:** referente ao valor de retorno, isto é, qual o tipo de valor que este método retorna;
- **Lista de argumentos:** parâmetros necessários para execução da função (método), sendo necessários o tipo e um identificador para o parâmetro;

Acessibilidade / Visibilidade

Especifica quão acessível o método ou atributo é.

+ (Público)

Palavra-chave **public** em C++. Especifica que estes membros (métodos e atributos) são acessíveis a partir de qualquer função. Isso se aplica a todos os membros declarados até o próximo especificador de acesso ou o fim da classe. Ou seja, os membros são visíveis a todos.

Acessibilidade / Visibilidade

– (Privado)

Palavra-chave **private** em C++. Especifica que esses membros são acessíveis somente dentro de funções membros e amigos da classe (veremos essa relação mais adiante). Isso se aplica a todos os membros declarados até o próximo especificador de acesso ou o fim da classe. Ou seja, membros visíveis somente para membros da classe.

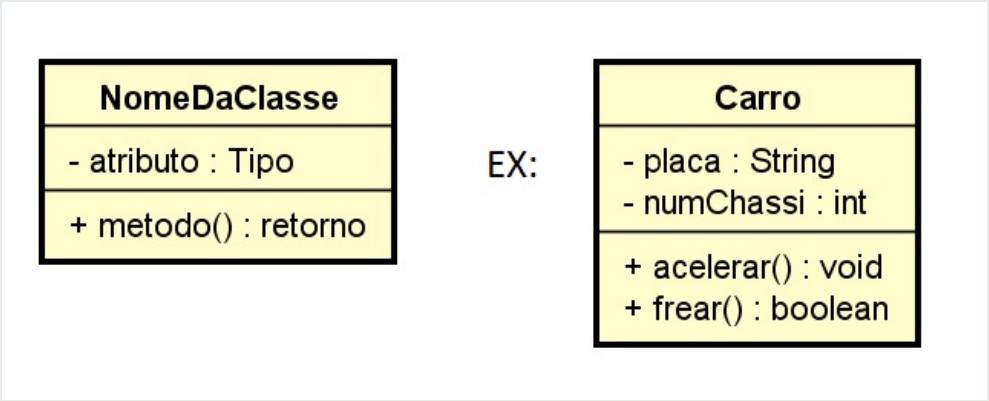
Acessibilidade / Visibilidade

(Protegido)

Palavra-chave **protected** em C++. Especifica o acesso a membros da classe na lista de membros até o próximo especificador de acesso ou o final da definição de classe. O *protected* é mistura entre *public* e *private*, ou seja, os membros protegidos são visíveis somente para membros da classe ou subclasses (herdeiras).

Classes

Diagramando...



Interpretando...

Temos dois atributos:

- placa, do tipo texto (**string**), privado;
- numChassi, do tipo inteiro (**int**), privado.

Temos dois métodos:

- acelerar, não recebe parâmetro e não retorna nada, público;
- frear, não recebe parâmetro e retorna um booleano (**bool**), público.

Classes vs TDA

Tipo de Dado Abstrato (TDA)

- Acesso direto aos atributos (variáveis contidas em um **struct**);
- As operações de um TDA são implementadas fora do escopo do TDA.

Classes

- Encapsulamento: pode haver restrição ao acesso de atributos ou métodos (funções-membros);
- Generalização de um TDA, para além de dados, uma classe pode também conter funções de manipulação desses dados (métodos).

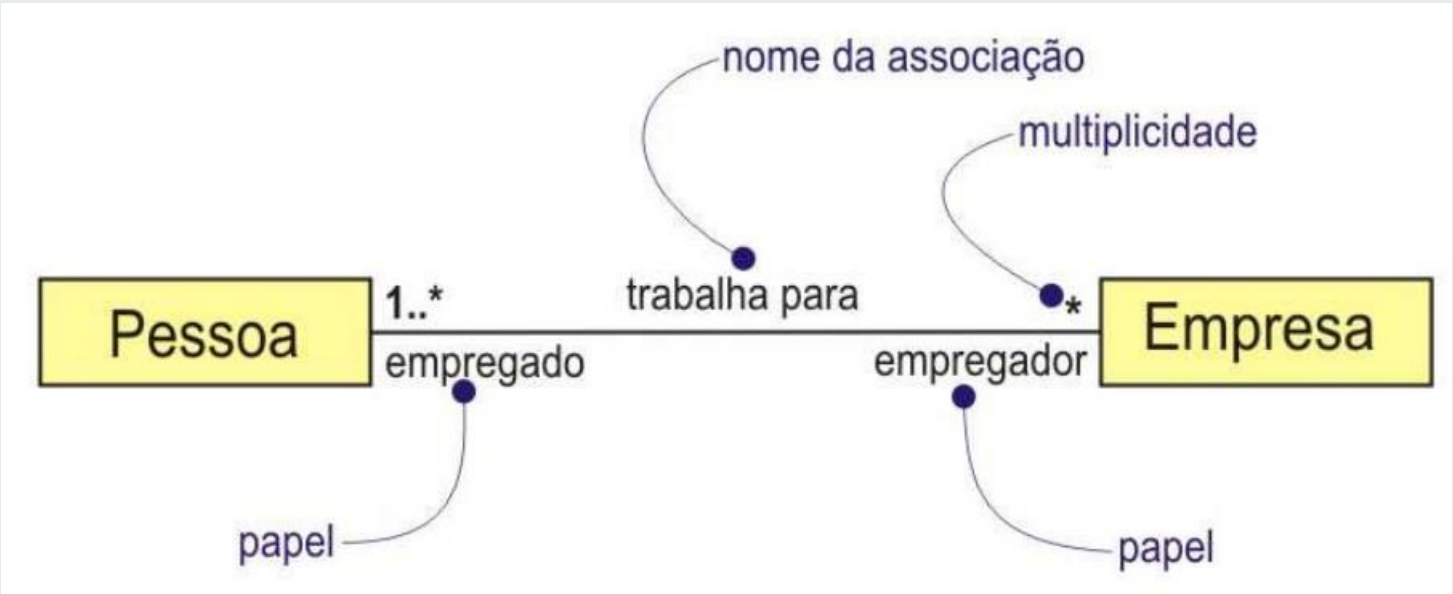
Relacionamentos

Relacionamentos podem possuir:

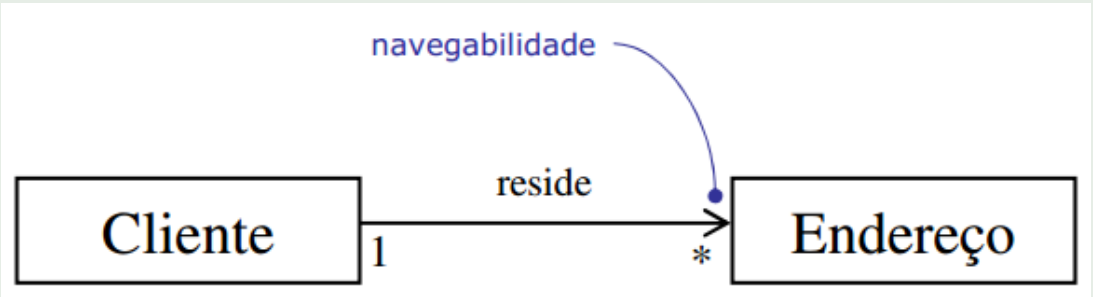
- Nome: descrição dada ao relacionamento;
- Sentido de leitura;
- Navegabilidade: indicada por uma seta no fim do relacionamento;
- Multiplicidade: 0..1, 1..*, 2, 3..7
- Tipo: associação (agregação, composição), generalização e dependência;
- Papéis: refere-se ao emprego de cada classe no relacionamento.

Relacionamentos

Exemplo 1



Exemplo 2



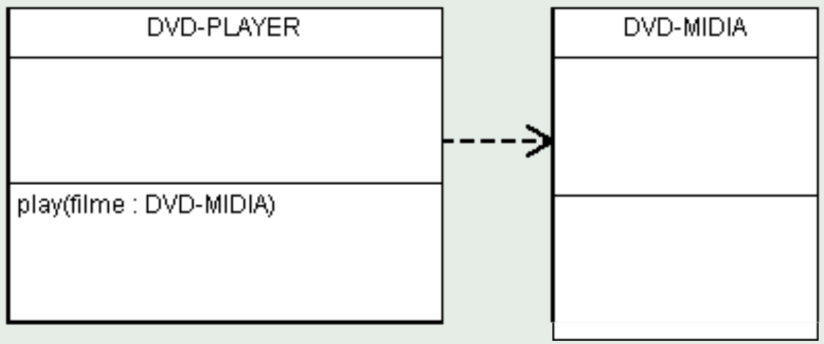
Tipos de Relacionamentos

Dependência (“precisa de”)

Duas classes possuem um relacionamento de dependência quando uma alteração em uma dessas classes pode afetar a outra classe. O inverso não é verdade. Nesse contexto, diz-se que uma classe utiliza a outra como argumento em sua assinatura.

Exemplo de Dependência

As classes DVD-PLAYER e DVD-MIDIA apresentam um relacionamento de dependência. A assinatura do método play da classe DVD-PLAYER recebe como parâmetro um objeto ou instância da classe DVD-MIDIA.



Obs.: poderia ser linha contínua

Tipos de Relacionamentos

Generalização ("é um tipo de")

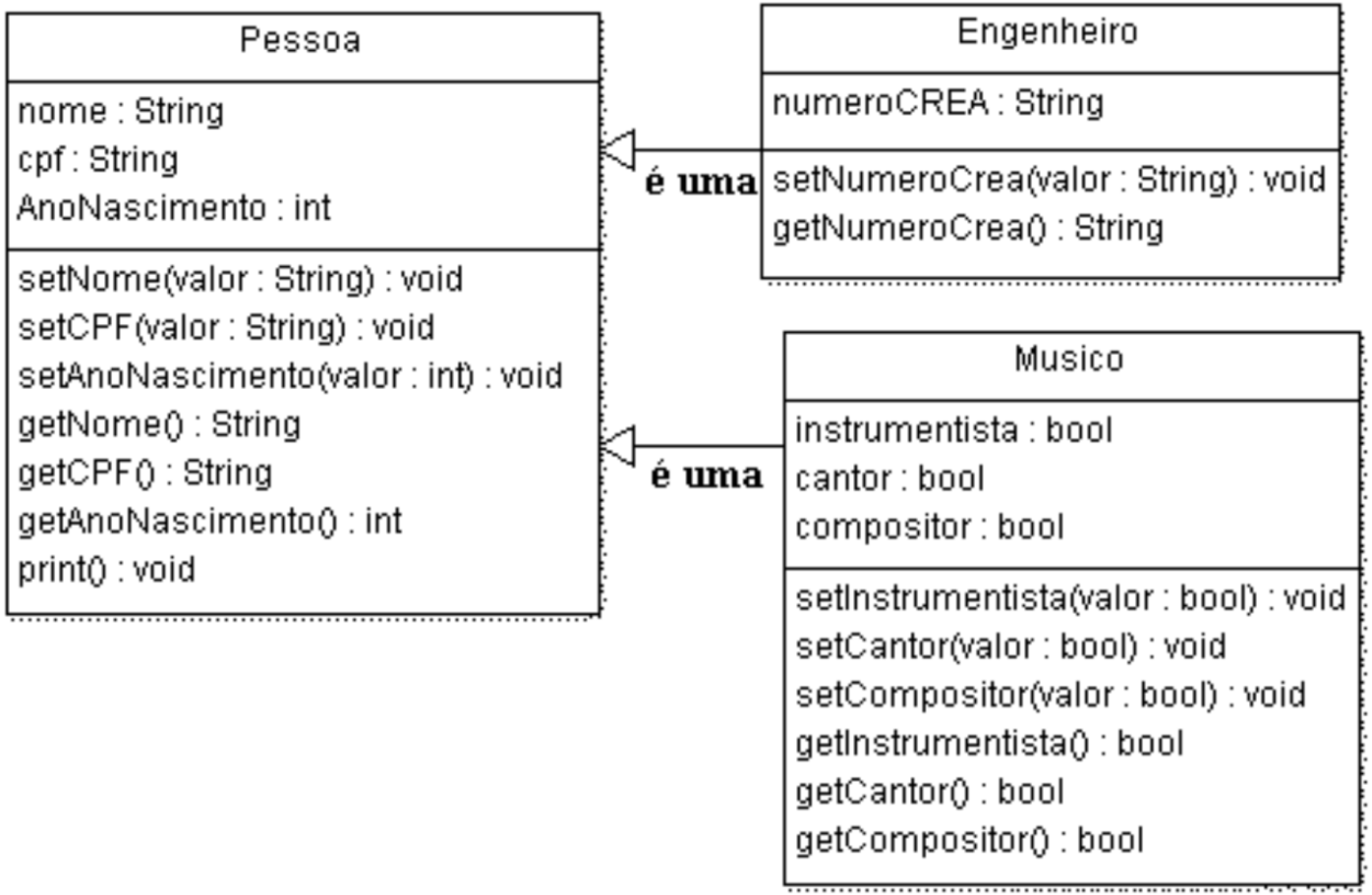
Duas classes possuem um relacionamento de generalização quando uma das classes especializa ou detalha a outra. A classe genérica é denominada de superclasse ou classe pai e a outra classe de subclasse ou classe filha.

Exemplo de Generalização

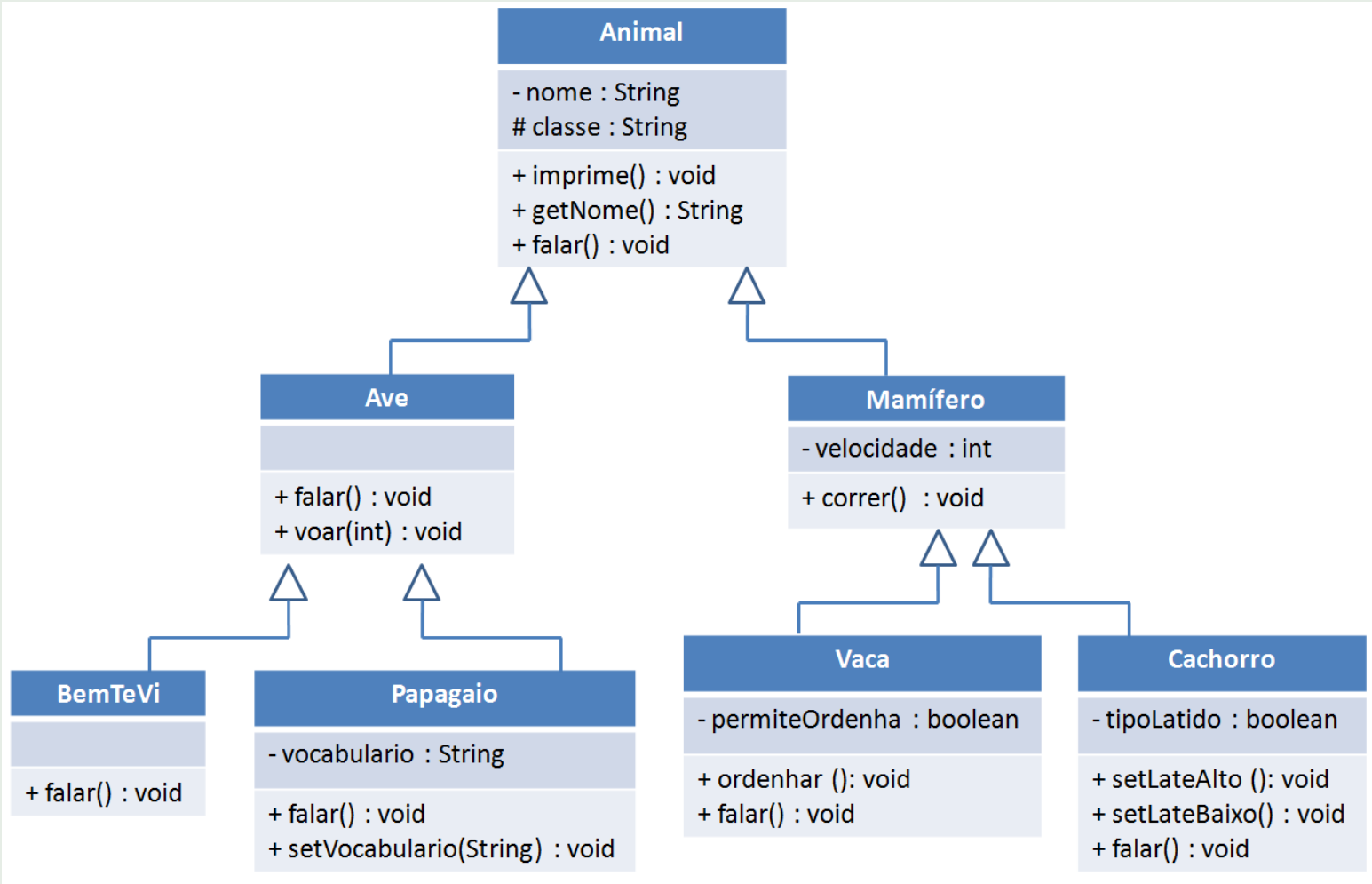
Três classes são representadas: Pessoa, Engenheiro e Musico. Nessa representação, a generalização permite dizer "Engenheiro é uma Pessoa" e "Musico é uma Pessoa". Ou seja, os objetos do mundo real representados pela classe Engenheiro e pela classe Pessoa, possuem algumas características comuns (atributos e métodos). Essas características comuns são os membros da classe Pessoa.

Tipos de Relacionamentos

Exemplo de Generalização



Outro Exemplo de Generalização



Tipos de Relacionamentos

Associação

Uma associação é uma relação estrutural. Ou seja, ela informa que uma classe faz parte da estrutura de outra. Por exemplo: o motor faz parte da classe automóvel, um empregado está associado a uma empresa ou um músico está associado a uma banda.

Subdivisões da Associação

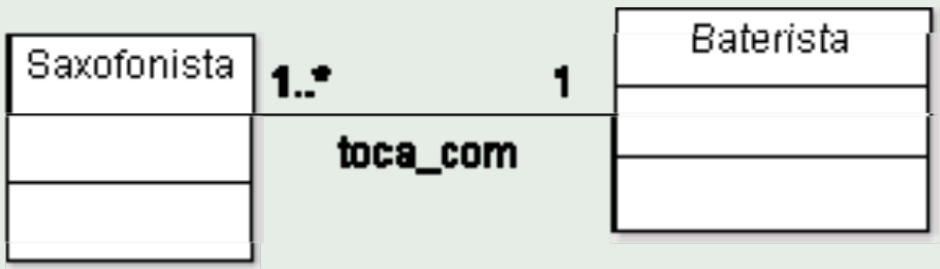
- Plana;
- Agregação:
  - Agregação Simples;
  - Composição.

Tipos de Relacionamentos

Associação Plana

Representa uma relação estrutural onde as classes possuem a mesma importância. Uma linha ligando duas classes representa graficamente essa relação.

Exemplo de Associação Plana



Tipos de Relacionamentos

Exemplo de Associação Plana

```
#ifndef Saxofonista_h
#define Saxofonista_h

class Baterista;

class Saxofonista {
public:
  Baterista *toca_com;
};

#endif // Saxofonista_h
```

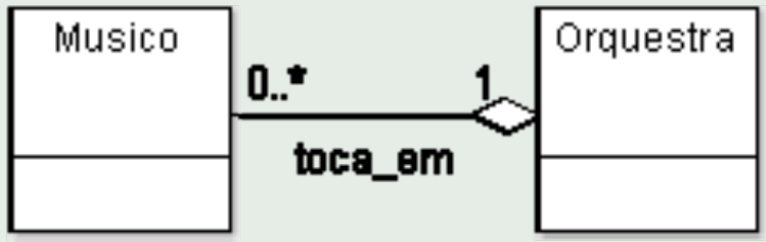
```
#ifndef Baterista_h
#define Baterista_h

#include <vector>

class Saxofonista;

class Baterista {
public:
std::vector< Saxofonista* >
    toca_com;
};

#endif // Baterista_h
```

Tipos de Relacionamentos

Agregação Simples

Representa a estrutura todo-parte. Ela é representada por uma linha ligando as duas classes da relação e a presença de um losango. O losango é colocado na conexão entre a linha e a classe que é considerada a mais importante da relação.

Exemplo de Agregação Simples



Tipos de Relacionamentos

Exemplo de Agregação Simples

```
#ifndef Musico_h
#define Musico_h

class Orquestra;

class Musico {
public:
  Orquestra *toca_em;
};

#endif // Musico_h
```

```
#ifndef Orquestra_h
#define Orquestra_h
#include <vector>

class Musico;

class Orquestra {
public:
  std::vector< Musico* >
      toca_em;
};

#endif // Orquestra_h
```

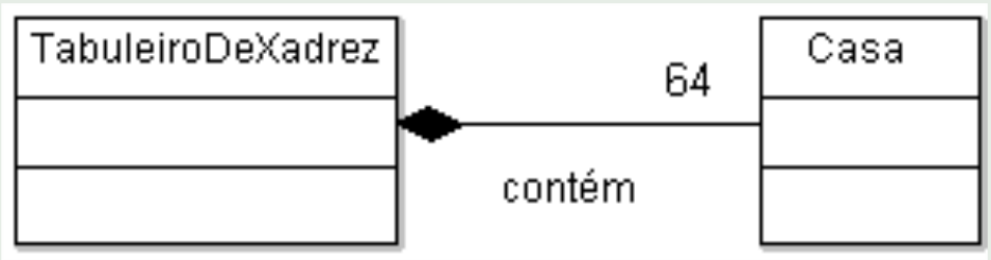
Tipos de Relacionamentos

Composição

A agregação simples possui uma variante, a composição. Essa variação adiciona um grau de importância semântica à relação. Ela define a relação de posse ou possessão. Isso significa que um objeto da classe pertence apenas e exclusivamente ao objeto da outra classe.

Exemplo de Composição

A classe TabuleiroDeXadrez que deve conter instâncias da classe Casa. As classes Casa, que podem ter o atributo cor como sendo "preta" ou "branca", compõe um Tabuleiro de Xadrez. Mais especificamente, um tabuleiro de Xadrez é composto por 32 casas pretas e 32 casas brancas.



Tipos de Relacionamentos

Exemplo de Composição

```
#ifndef TabuleiroDeXadrez_h
#define TabuleiroDeXadrez_h

#include "Casa.h"

class TabuleiroDeXadrez {
public:
  Casa contem[64];
};

#endif //
   TabuleiroDeXadrez_h
```

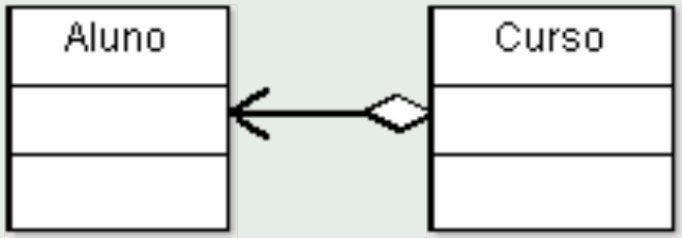
```
#ifndef Casas_h
#define Casas_h

class TabuleiroDeXadrez;

class Casa {
public:
  TabuleiroDeXadrez *
      contem;
};

#endif // Casa_h
```

Tipos de Relacionamentos

Associação Unidirecional

Em algumas situações, pode-se desejar que uma das classes que participa da relação não tenha consciência disso. Nesses casos, têm-se uma relação unidirecional.

Exemplo de Associação Unidirecional



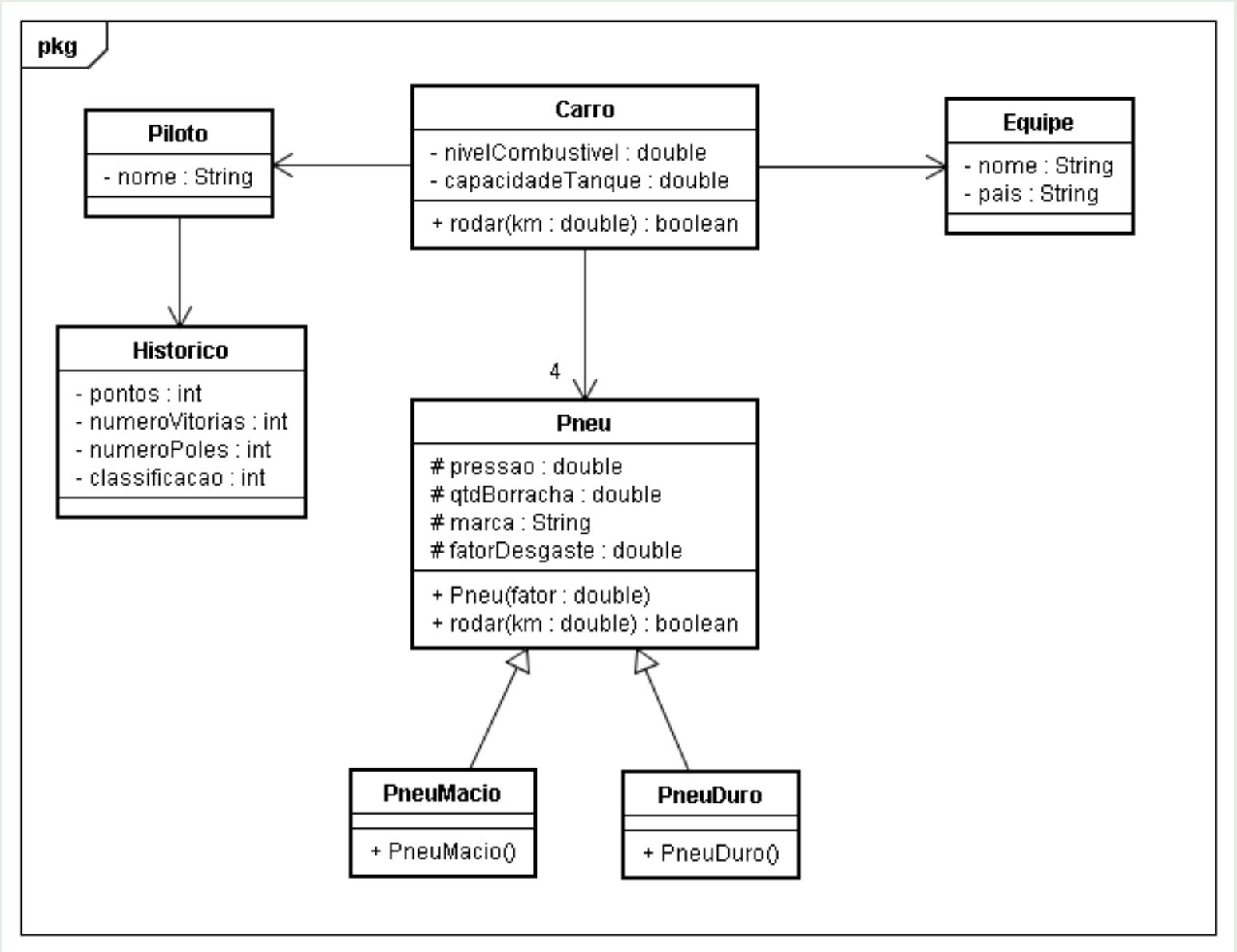
Tipos de Relacionamentos

Exemplo de Associação Unidirecional

```
#ifndef Aluno_h
#define Aluno_h

class Aluno {

};

#endif // Aluno_h
```

```
#ifndef Curso_h
#define Curso_h

class Aluno;

class Curso {
public:
  Aluno *myAluno;
};

#endif // Curso_h
```

Um último exemplo...



Softwares

Sugestões de softwares para documentação

- Modelio;
- Umbrello UML Modeller;
- Draw.io (online).

Referências

- http://www.dca.fee.unicamp.br/cursos/PooJava/desenvolvimento/umlclass.html
- http://www.univasf.edu.br/~ricardo.aramos/disciplinas/ES_II_2011_1/aulas/DiagrClasses.pdf
- http://wiki.les.inf.puc-rio.br/wiki/images/7/7f/Aula1-diagrama_classes.pdf
- http://www.professorvida.com.br/if62c/material/relacionamentos.pdf