Faculdades senac

Engenharia de Software 1

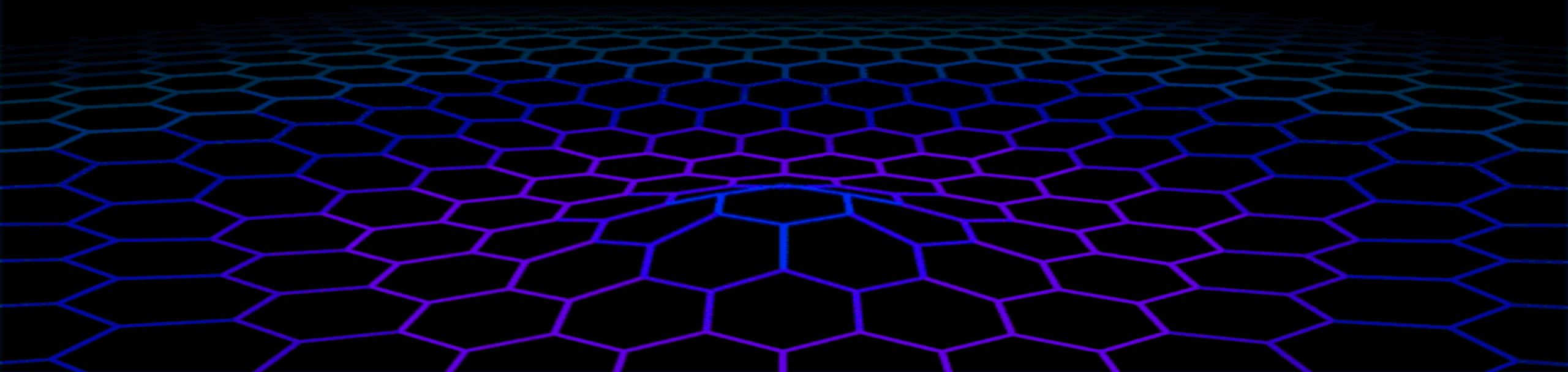


Diagrama de Classe

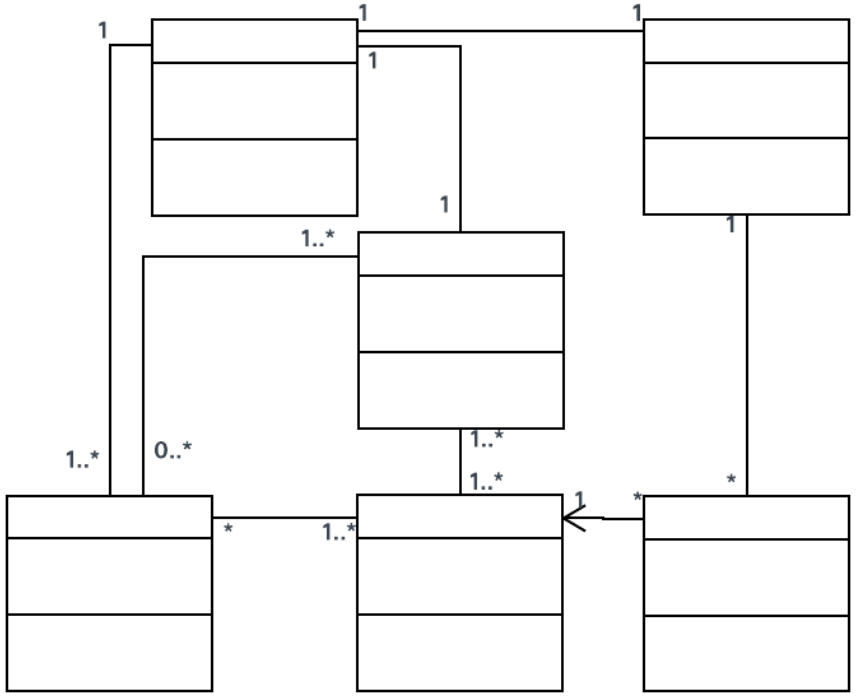
Diagrama de Classe

Diagrama de Classe

Estrutura estática

Exibe Classes, Atributos, Operações e relações entre eles

Descrevem as responsabilidades do sistema



Obs.:

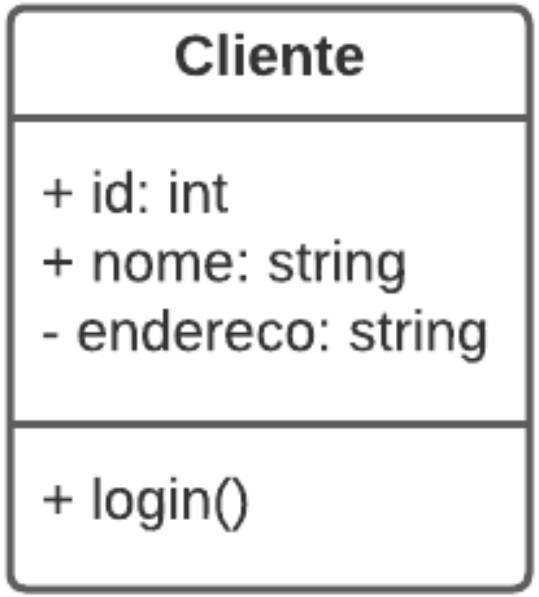
Podem ser mapeados facilmente para linguagens orientadas a objeto.

Podem ser usados para engenharia reversa

Notação de Classe

Uma classe é dividida em 3(três partes):

- Nome
- Atributos
- Operações



Nome

Localizado no compartimento superior

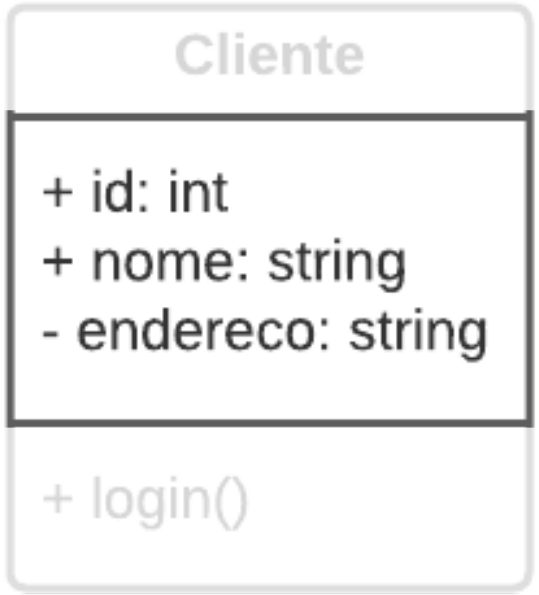
Escrito em **bold**

Deve iniciar com letra maiúscula

Atributos

Devem ser significativos

Ficam no compartimento central



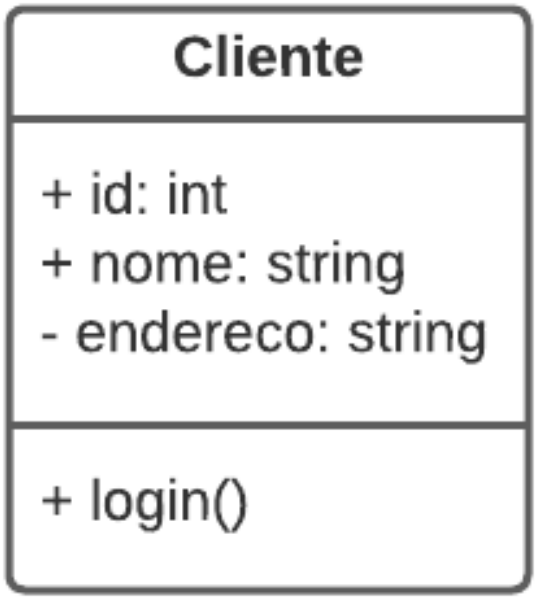
Métodos (Operações)

Processos que uma classe sabe executar

Modificadores de acesso

Níveis de acesso e símbolos correspondentes:

- **+ Public** (Público)
- **# Protected** (Protegido)
- **~ Package** (Pacote)
- **- Private** (Privado)

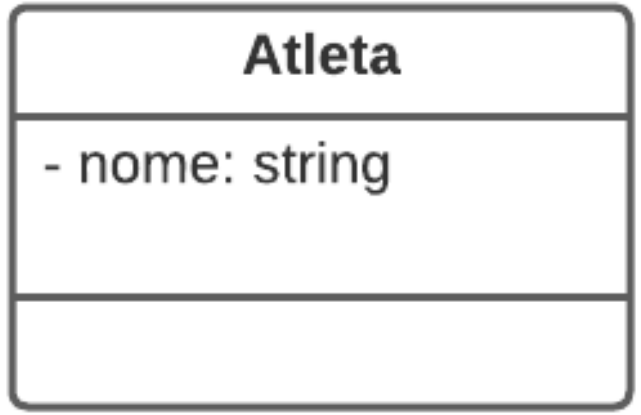


Private (Privado) -

`- Private`

Modificador mais restritivo

Acessível somente pela classe que os declara



Atributo **nome** da classe Atleta com modificador *private*.

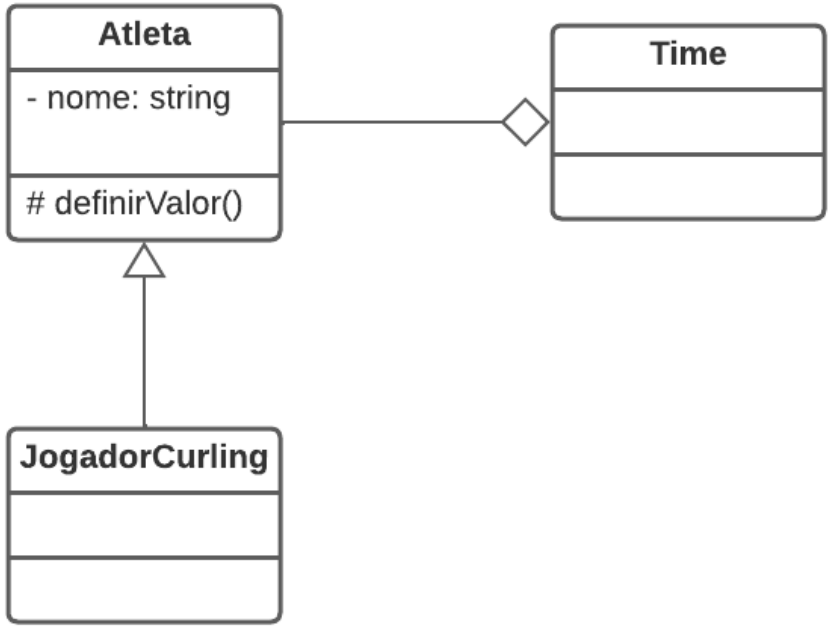
Obs.: Métodos e atributos com o modificador *private* não são herdados.

Protected (Protegido)

`# Protected`

Acessíveis pela classe que os declara e suas subclasses em outros pacotes e outras classes dentro do mesmo pacote.

O método **`definirValor()`** é visível na subclasse "**JogadorCurling**".



Obs.: Métodos e atributos declarados com o modificador *protected* numa *superclasse* devem ser definidos como *protected* ou *public* em suas *subclasses* e nunca *private.*

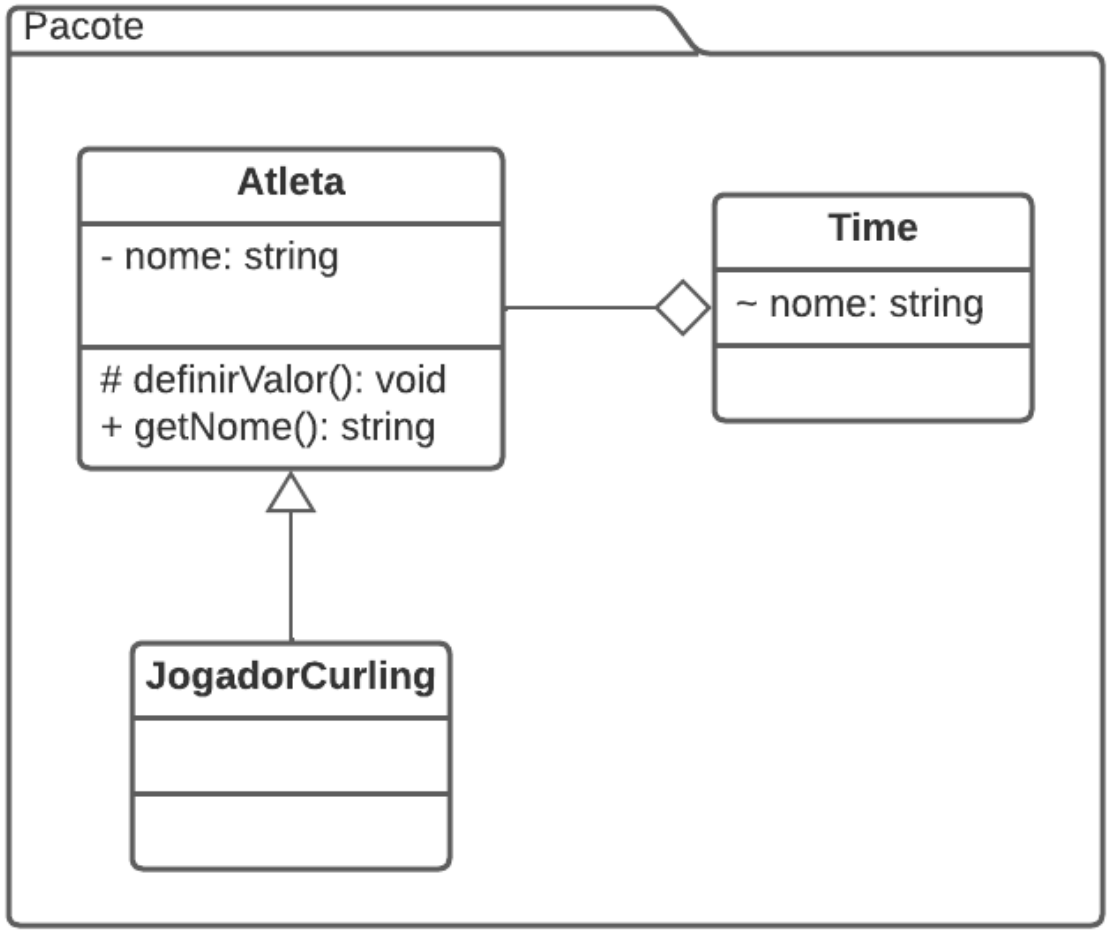
~ Package (Pacote)

~ Package

Modificador padrão quando nenhum for definido

Métodos e classes são visíveis por todas as classes dentro do mesmo pacote

O atributo **nome** da classe "**Time**" é visível por todas as classes dentro do mesmo pacote.



Obs.: Todos os métodos e atributos declarados como *public* são herdados pelas *subclasses*. Métodos e atributos declarados como *public* devem se manter *public* em todas as *subclasses*.

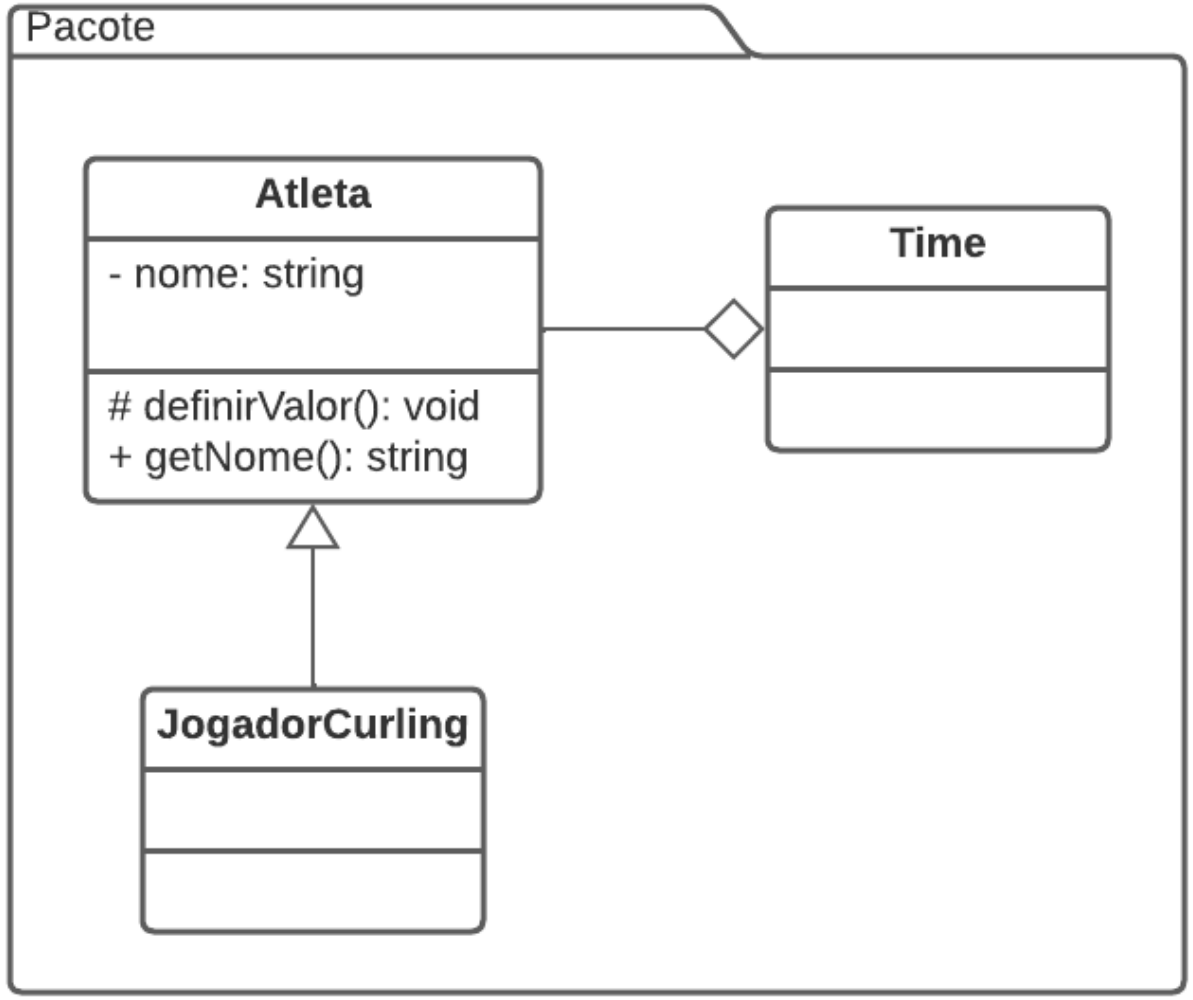
Public

+ Public

Mais permissivo

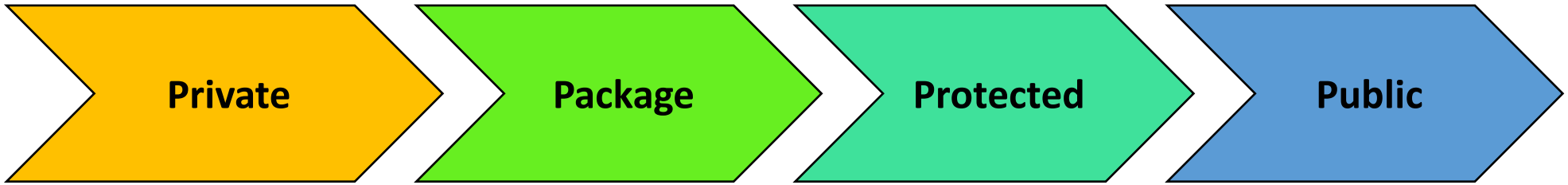
Acessíveis por qualquer classe

O método **getNome()** é visível por qualquer outra classe.



Obs.: Todos os métodos e atributos declarados como *public* são herdados pelas *subclasses*. Métodos e atributos declarados como *public* devem se manter *public* em todas as *subclasses*.

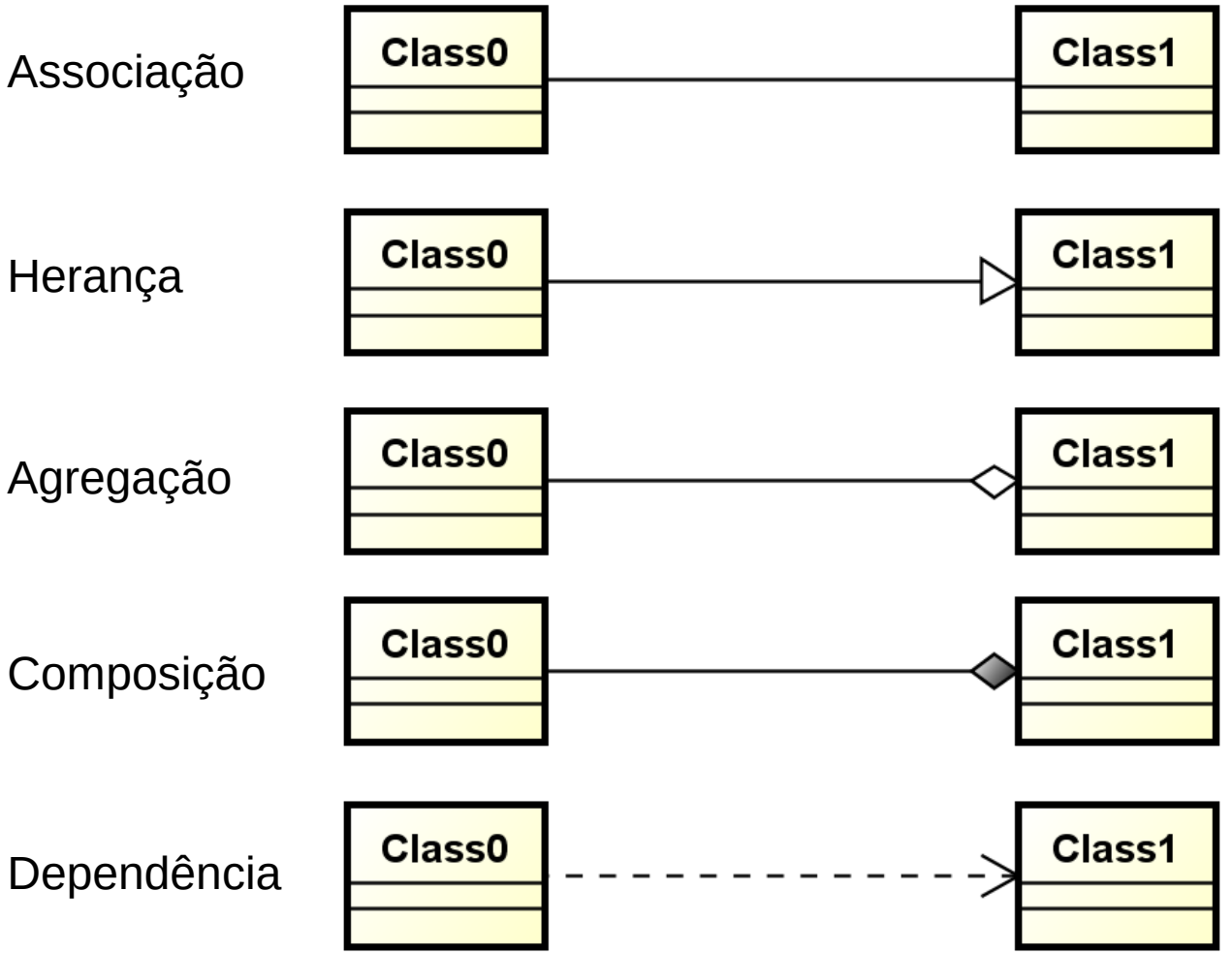
Ordem de visibilidade



do mais restrito ao mais liberal

Relacionamentos de Classe

Principais tipos de relacionamentos:



Desenhando diagrama de Classes

Usamos Diagrama de Classes quando precisamos descrever a visão estática de um sistema ou suas funcionalidades.

1 - Identificar os nomes das classes

O primeiro passo é identificar os objetos primários do sistema.

2 - Distinguir relações

O segundo passo é determinar como cada uma das classes ou objetos estão relacionados entre si.

Criar a estrutura

Adicionar os nomes das classes e ligá-las aos conectores apropriados.

Adicionar atributos e métodos posteriormente

Criar a estrutura

Adicionar os nomes das classes e ligá-las aos conectores apropriados.

Adicionar atributos e métodos posteriormente

Boa práticas

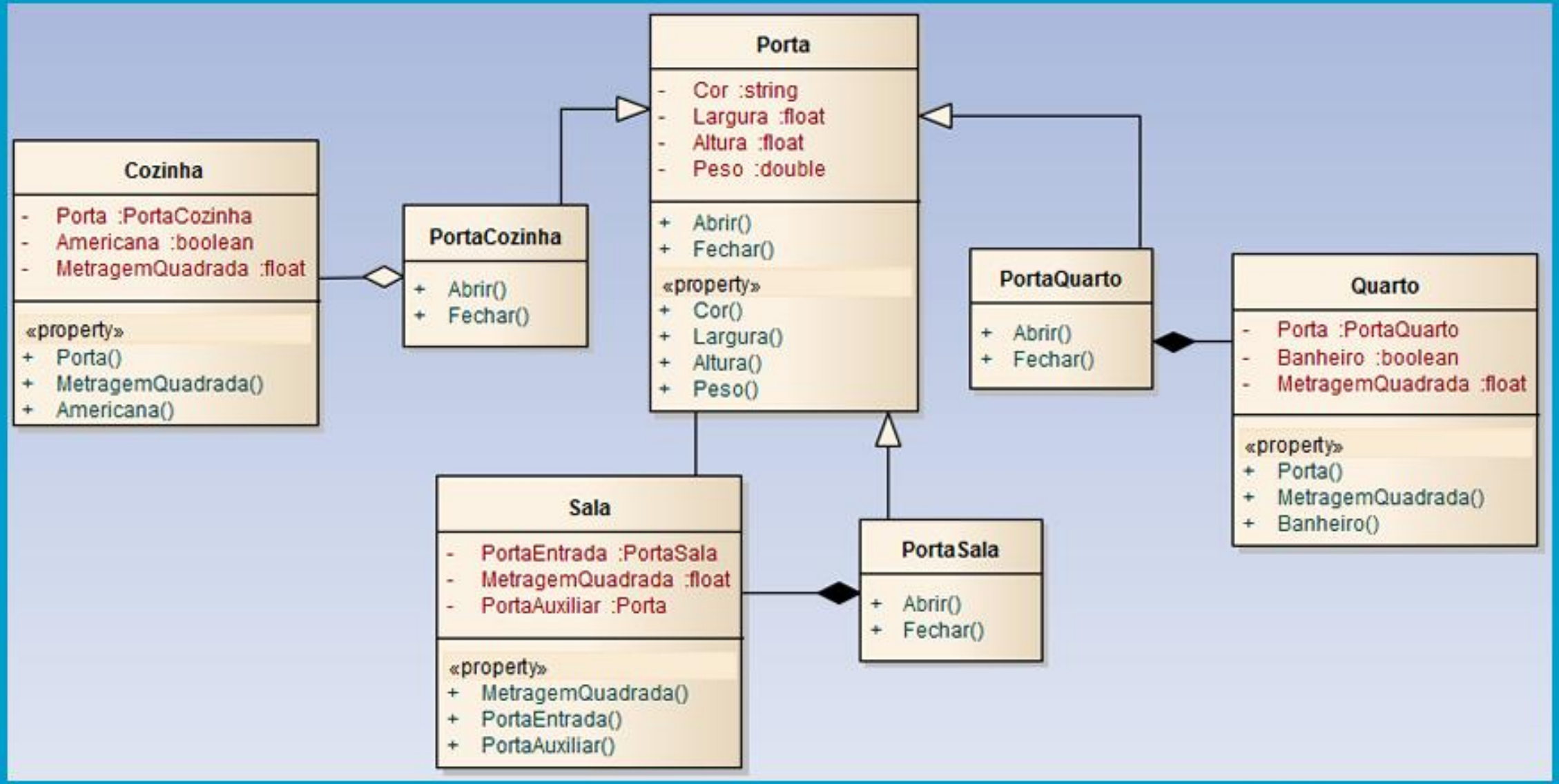
Diagramas tendem a ficar incoerentes à medida que crescem.

Evitar criar diagramas grandes (divida em diagramas menores).

Muitas linhas sobrepostas confundem o leitor.

Cores diferentes em classes diferentes ajudam o leitor a ler melhor.

Exemplo



Fonte: https://www.ateomomento.com.br/uml-diagrama-de-classes/

Glossário

Classes: Representa um objeto ou um conjunto de objetos que compartilham uma estrutura e comportamento comum.

Sinais: Representam comunicações unidirecionais e assíncronas entre objetos.

Tipos de dados: Definem valores de dados.

Pacotes: Retângulos com abas que servem para organizar classificadores em um diagrama.

Objetos: Instâncias de uma classe ou classes.

Artefatos: Representam as entidades concretas em um sistema de software (documentos, bancos de dados, arquivos executáveis, ...)

Interações: Relações e ligações que podem existir em diagramas de classes e objetos.

Generalização (Hereditariedade): Subclasse assumindo a funcionalidade de uma superclasse.

Associação bidirecional: Relação padrão entre duas classes Representada por uma linha reta.


TUTORIAL DE
DIAGRAMAS
DE CLASSES

FIM