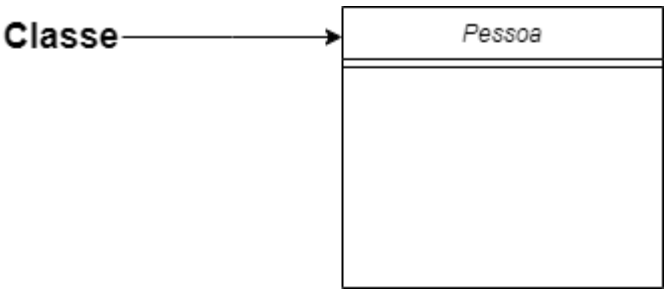
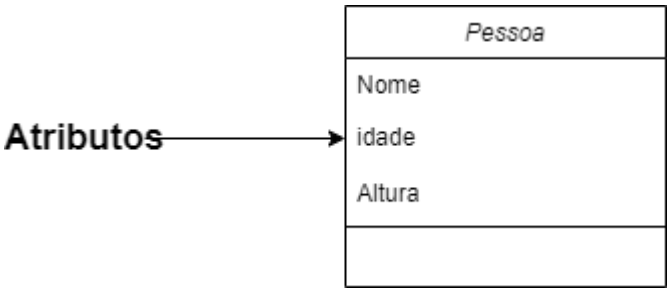
Diagrama de Classe

Os diagramas de classes mostram a estrutura estática do sistema, ou seja, as classes, seus atributos e métodos, e as relações entre elas. Porém, antes disso, vamos compreender o que é cada elemento de um diagrama de classe e o que representa.

- **Classes:** Uma classe representa um conjunto de objetos que compartilham características e comportamentos comuns. É como um molde que define a estrutura e o comportamento de um grupo de objetos. Pense em uma classe como uma descrição abstrata de um conceito do mundo real.

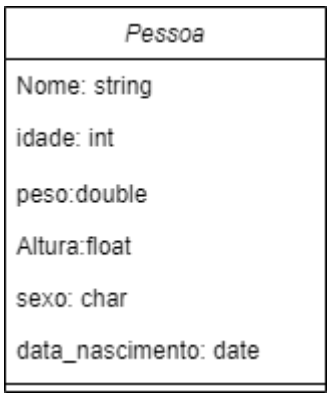


- **Atributos:** Representa uma propriedade ou característica de um objeto que pertence a uma determinada classe. É uma variável que armazena um valor associado a cada instância da classe. Os atributos definem os dados que caracterizam os objetos de uma classe. Os atributos em diagramas de classe UML possuem tipos de dados. O tipo de dado define o conjunto de valores que um atributo pode assumir e as operações que podem ser realizadas sobre ele.



Tipos Primitivos:

- **Inteiros:** Números inteiros (int, long, short).
- **Reais:** Números com casas decimais (float, double).
- **Booleanos:** Valores lógicos (true ou false).
- **Caracteres:** Caracteres individuais (char).
- **Strings:** Sequências de caracteres (string).
- **Datas:** Representações de datas e horários (date, time).



Visibilidade: Refere-se ao nível de acesso que os atributos e métodos têm em uma classe, definindo como e onde podem ser utilizados. Os principais níveis de visibilidade são: público, privado e protegido.

Público (+)

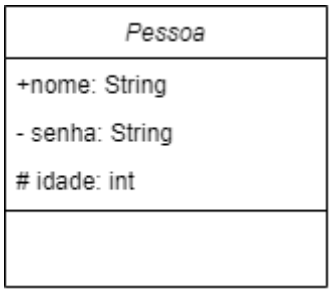
- **Símbolo: +**
- **Descrição:** O atributo é acessível de qualquer outra classe. Não há restrições no acesso.
- **Uso:** Geralmente usados para atributos que devem ser acessíveis amplamente, como em classes de utilidades ou APIs.

Privado (-)

- **Símbolo: -**
- **Descrição**: O atributo é acessível apenas dentro da própria classe. Outras classes não podem acessar diretamente esses atributos.
- **Uso**: Usado para encapsular dados e proteger a integridade do estado interno de um objeto. Isso força a utilização de métodos públicos (getters e setters) para acessar ou modificar os atributos.

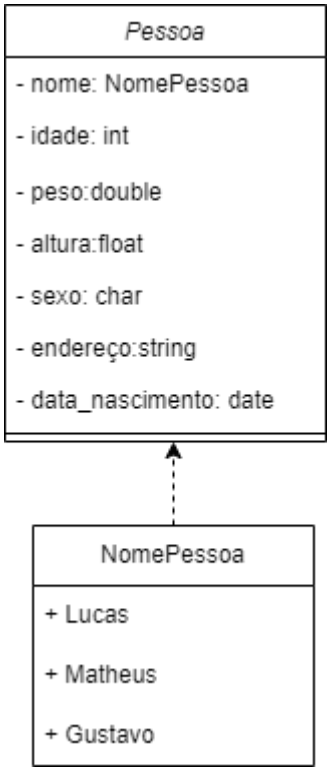
Protegido (#)

- **Símbolo: #**
- **Descrição:** O atributo é acessível dentro da própria classe e por classes que herdam dela (subclasses).
- **Uso:** Utilizado em hierarquias de classes onde os atributos devem ser acessíveis para subclasses, mas não para classes externas.



Tipos Enumerados:

- Definidos pelo usuário para representar um conjunto finito de valores possíveis. Por exemplo, um atributo "Nome" pode ter os valores Gustavo, Lucas e Matheus.



Tipos Objeto:

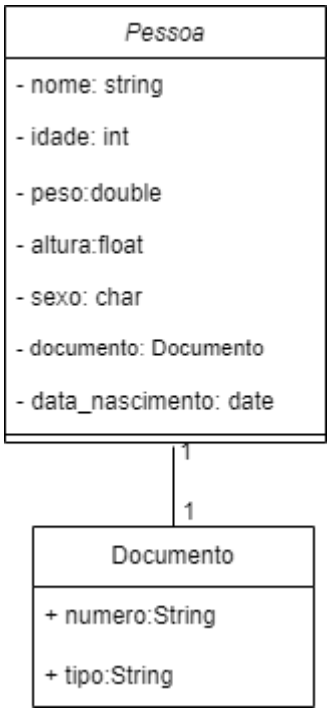
- Referenciam objetos de outras classes, estabelecendo relações entre classes. Mas para explicar tipos objeto não podemos deixar de explicar cardinalidade. A cardinalidade descreve a quantidade de instâncias de uma classe que podem se relacionar com instâncias de outra classe. Ela define a natureza da relação entre as classes em termos de quantidade.

Representação da Cardinalidade em UML

- 1 indica uma relação de um para um.
- 0..1 indica que pode haver zero ou uma instância.
- * (ou 0..*) indica que pode haver zero ou mais instâncias.
- * indica que pode haver muitas instancias.

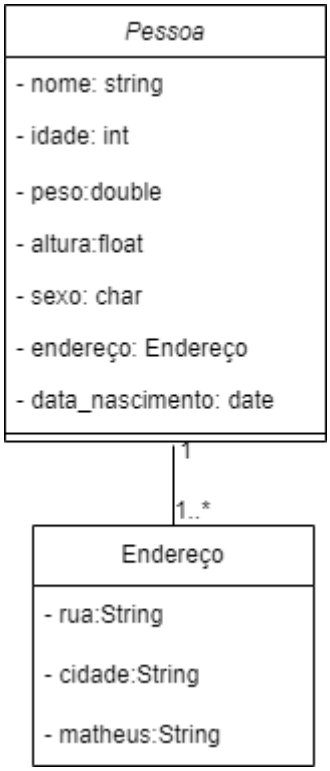
- **Um para um (1:1)**:

  Uma instância de uma classe está relacionada a uma única instância de outra classe. Exemplo: cada Pessoa tem um Documento de identidade único.



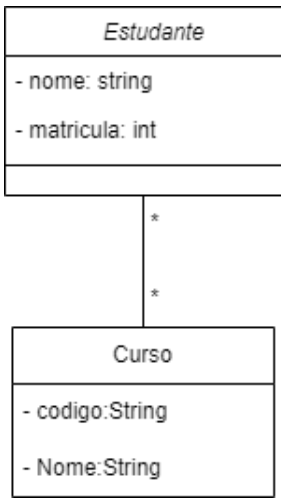
Um para muitos (1:N):

- Uma instância de uma classe está relacionada a várias instâncias de outra classe. Exemplo: um Cliente pode ter vários Pedidos.



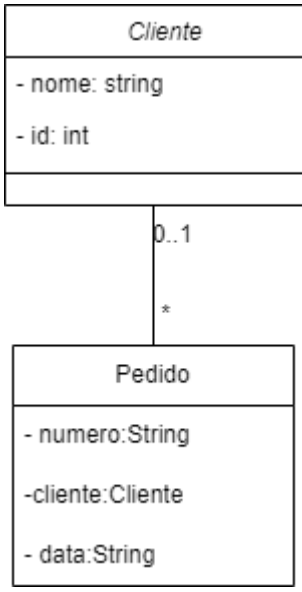
Muitos para muitos (N:M):

- Várias instâncias de uma classe estão relacionadas a várias instâncias de outra classe. Exemplo: Estudantes podem estar matriculados em várias Disciplinas, e cada Disciplina pode ter vários Estudantes.



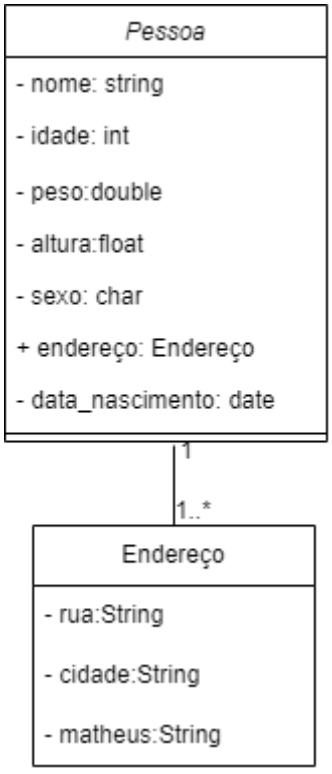
Zero ou Um para Muitos (0..1:N):

- Uma instância de uma classe pode estar relacionada a zero ou muitas instâncias de outra classe.

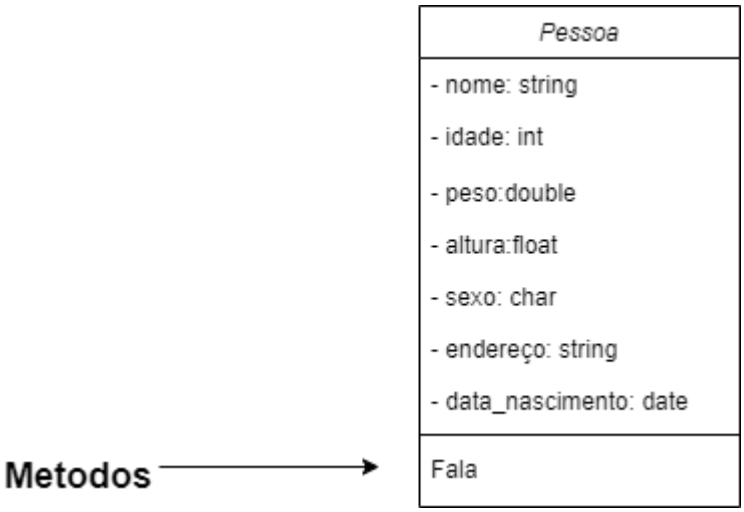


Tipos Coleção:

- Representam conjuntos de elementos, como listas, conjuntos e mapas.



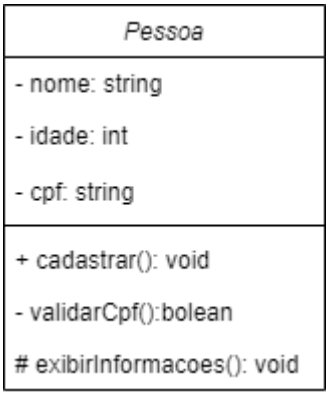
- **Métodos:** os métodos (ou operações) são representados de maneira que descrevem o comportamento da classe. Aqui estão os principais aspectos de como os métodos são representados e interpretados em um diagrama de UML.



Visibilidade: Indica o nível de acesso ao método:

- \+ (público): O método pode ser acessado de fora da classe.
- \- (privado): O método só pode ser acessado dentro da própria classe.
- # (protegido): O método pode ser acessado por classes que herdam da classe original.

  Tipo de Retorno: O tipo de dado que o método retorna. Se não retornar nada, normalmente se usa void.



Métodos:

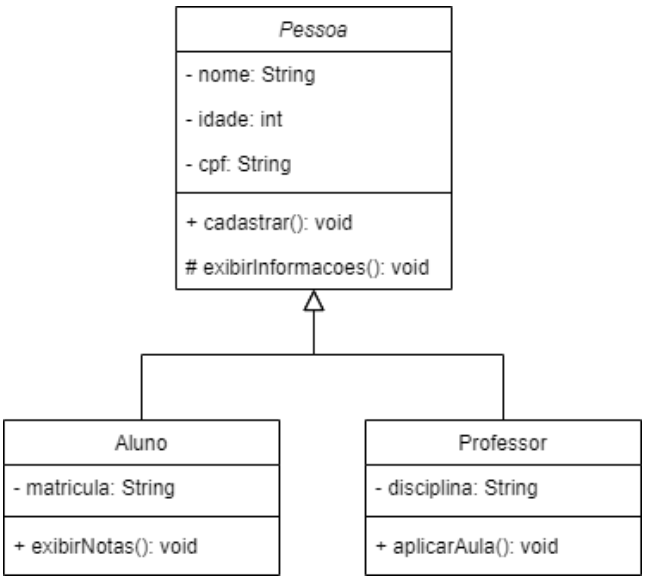
- + cadastrar(): void: Método público que pode ser chamado de fora da classe para cadastrar a pessoa. Ele não retorna nenhum valor.
- - validarCpf(): boolean: Método privado que valida o CPF da pessoa. Este método só pode ser chamado dentro da própria classe Pessoa e retorna um valor booleano indicando se o CPF é válido ou não.
- # exibirInformacoes(): void: Método protegido que pode ser acessado por subclasses (classes que herdam de Pessoa) ou pela própria classe. Ele é usado para exibir as informações da pessoa e não retorna nenhum valor.

Relacionamentos: Os relacionamentos definem como as classes interagem entre si, estabelecendo as conexões e dependências entre elas.

Tipos de Relacionamentos em UML:

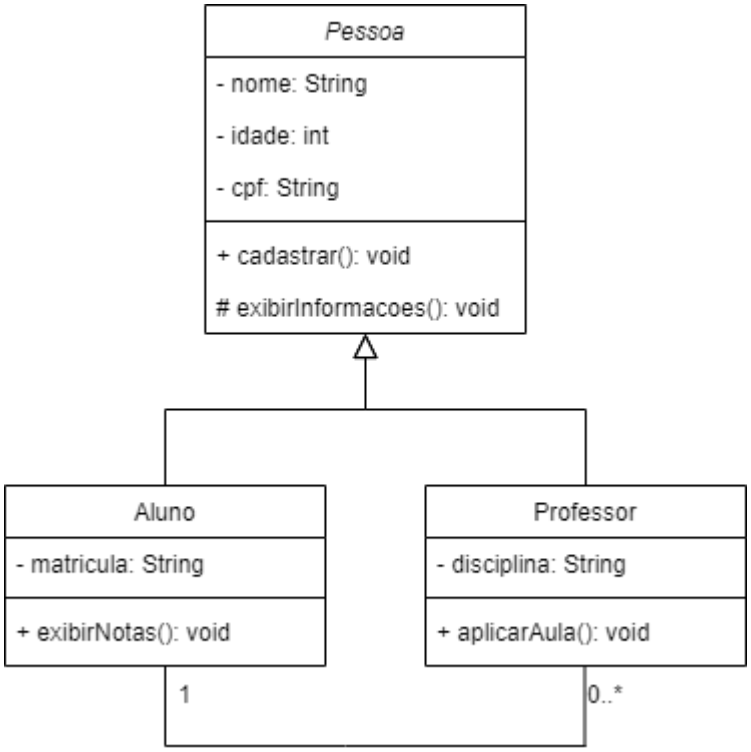
Generalização (Herança):

- Representa um relacionamento "é um".
- Uma classe filha (subclasse) herda todos os atributos e operações de uma classe pai (superclasse).
- É representado por uma linha sólida com uma seta aberta apontando para a superclasse.



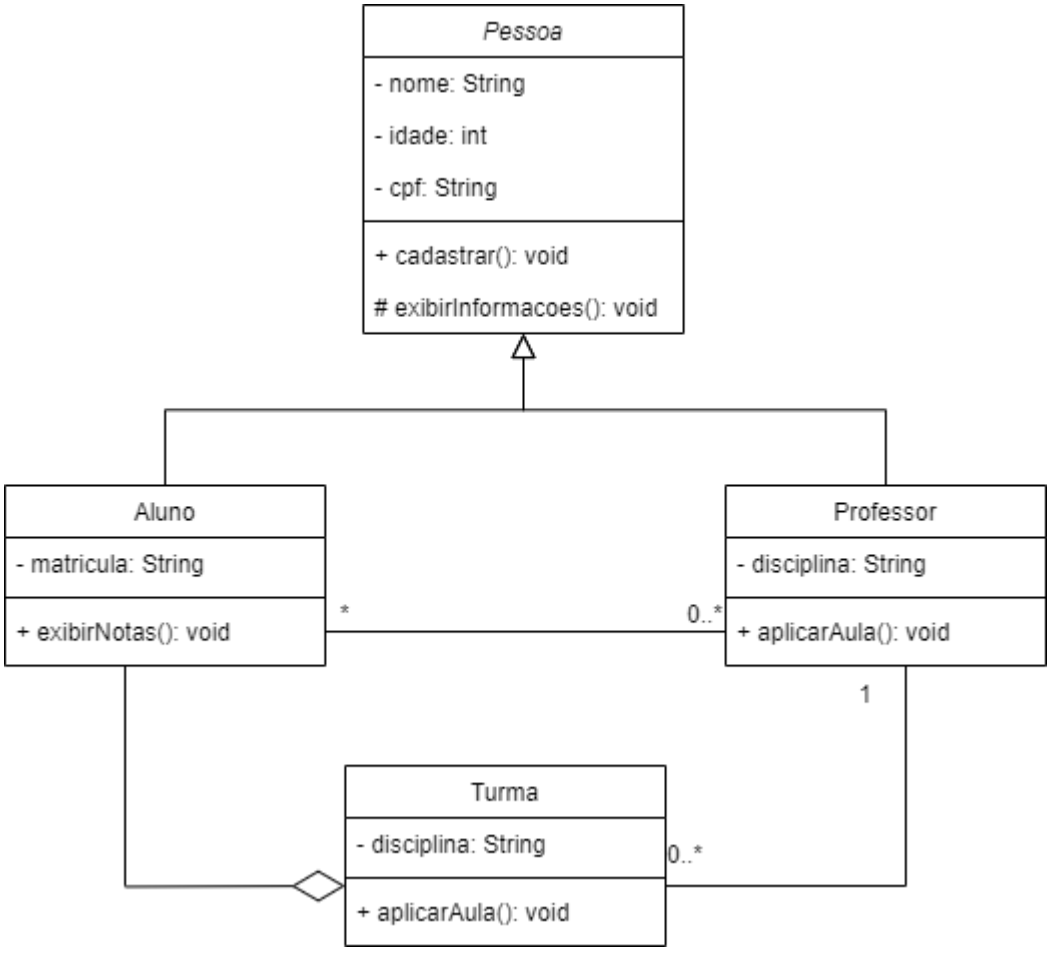
Associação:

- Representa um relacionamento entre duas classes.
- Indica que objetos de uma classe estão associados a objetos de outra classe.
- Pode ser unidirecional ou bidirecional.
- A multiplicidade define a quantidade de objetos que podem participar do relacionamento (um, muitos, zero ou um, etc.).



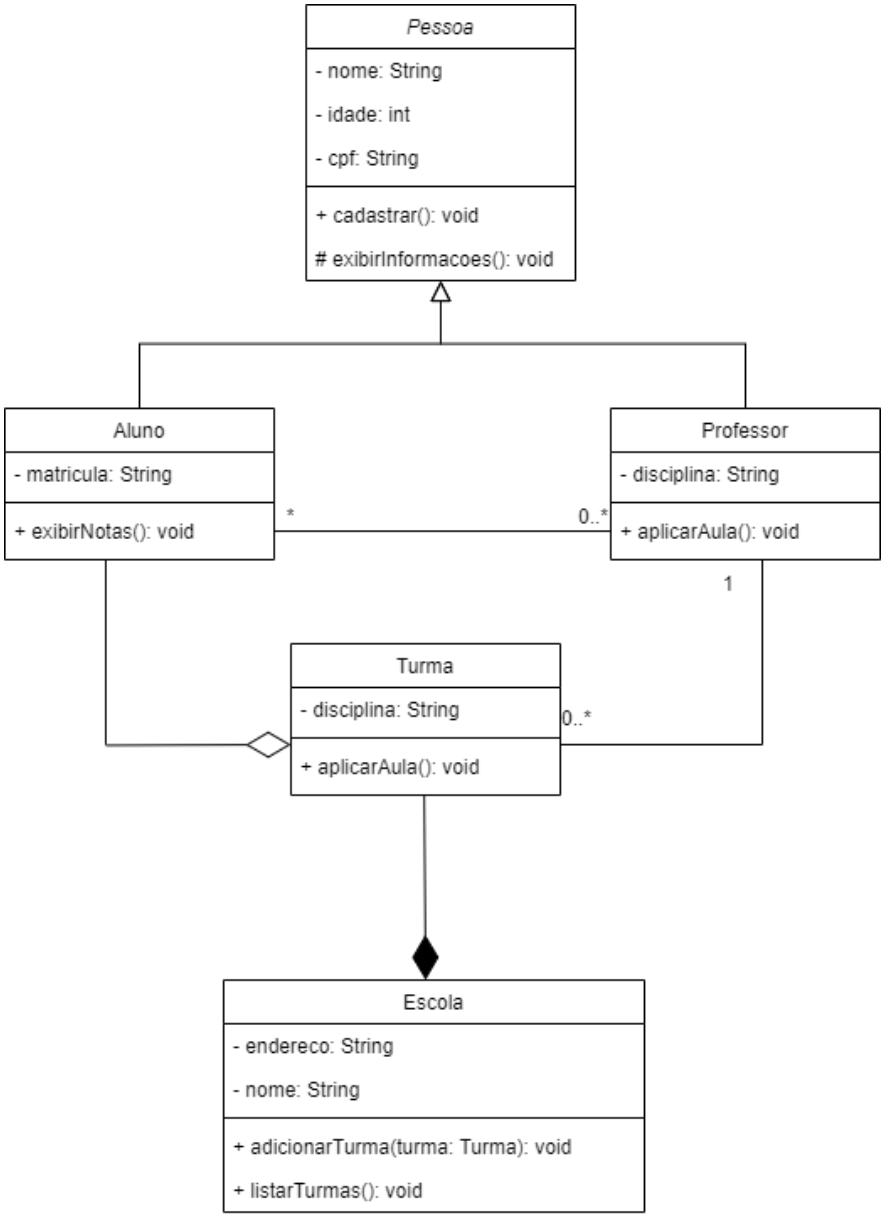
Agregação:

- É um tipo especial de associação que representa um relacionamento "tem um".
- Indica que uma classe é parte de outra classe, mas pode existir independentemente dela.
- É representada por um losango vazio na extremidade da classe que contém o todo.



Composição:

- É um tipo especial de agregação que representa um relacionamento "tem um" mais forte.
- Indica que uma classe é parte essencial de outra classe e não pode existir independentemente dela.
- É representada por um losango cheio na extremidade da classe que contém o todo.



Dependência:

- Representa um relacionamento mais fraco entre classes.
- Indica que uma classe depende de outra, mas não há um relacionamento de associação direto.
- É representada por uma linha tracejada com uma seta.