FUNDAMENTOS DA ORIENTAÇÃO A OBJETOS- REVISÃO

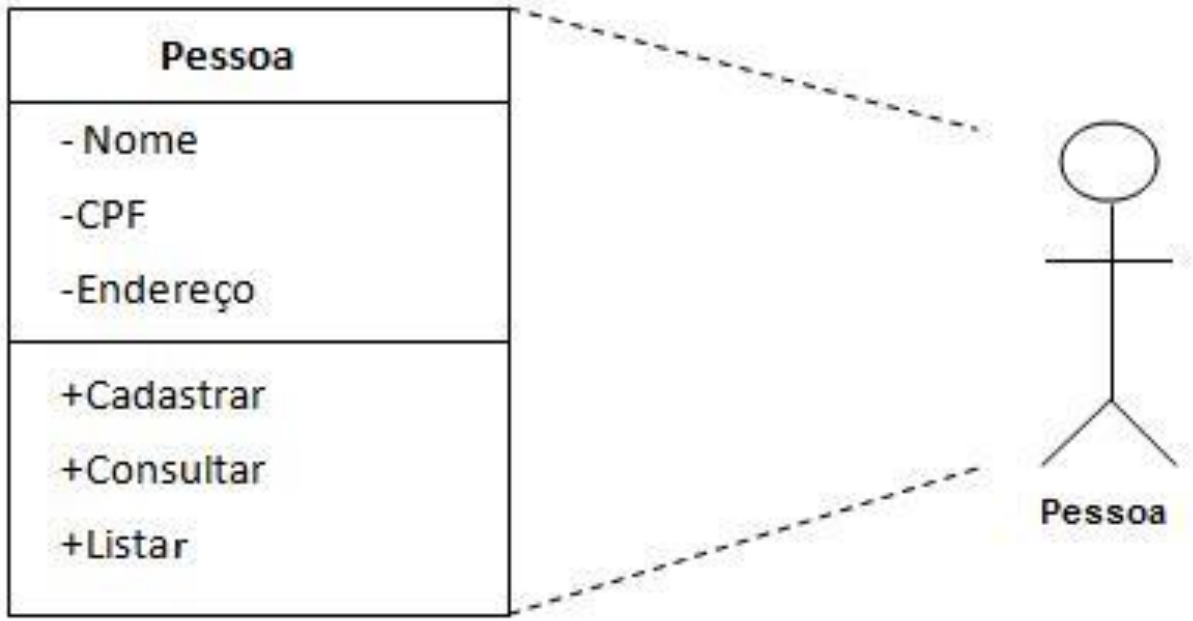
- Dado que a UML é uma ferramenta inserida no paradigma da orientação a objetos, vamos rever alguns conceitos fundamentais, dentre os quais, destacamos:
  - ✓ ***Classificação, abstração e instanciação***
  - ✓ ***Classes de objetos***
  - ✓ ***Atributos ou propriedades***
  - ✓ ***Métodos, operações ou comportamentos***
  - ✓ ***Visibilidade***
  - ✓ ***Herança***
  - ✓ ***Herança múltipla***
  - ✓ ***Polimorfismo***

ABSTRAÇÃO, CLASSIFICAÇÃO, INSTANCIAÇÃO E OBJETO

- **Abstração** – capacidade humana de se concentrar nos aspectos principais de um contexto qualquer, ignorando fatores menos importantes ou acidentais. A abstração permite representarmos o mundo real por meio de objetos;
- **Classificação** – é o ato de agrupar, em um mesmo conjunto, elementos que apresentam as mesmas características e comportamentos;
- **Objeto** – do ponto de vista da OO, objetos são estruturas utilizadas para representar elementos do mundo real, quer sejam concretos ou não;
- **Instanciação** – ato de criar um objeto, por meio da alocação de memória (concreto), a partir de uma classe (modelo).

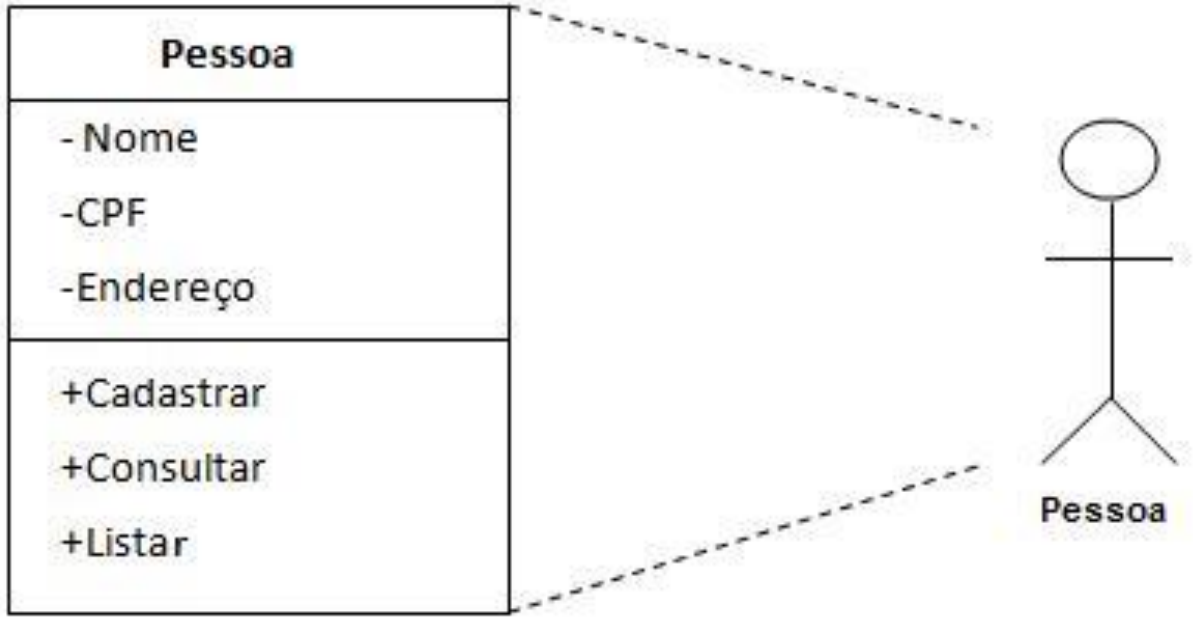
CLASSE

- **Classe** – representação simbólica ou modelo de um grupo de elementos que apresenta características em comum. Define uma categoria de elementos;
- Na UML, uma classe é representada por um retângulo, com até três divisões, contendo o nome da classe, seus métodos e atributos



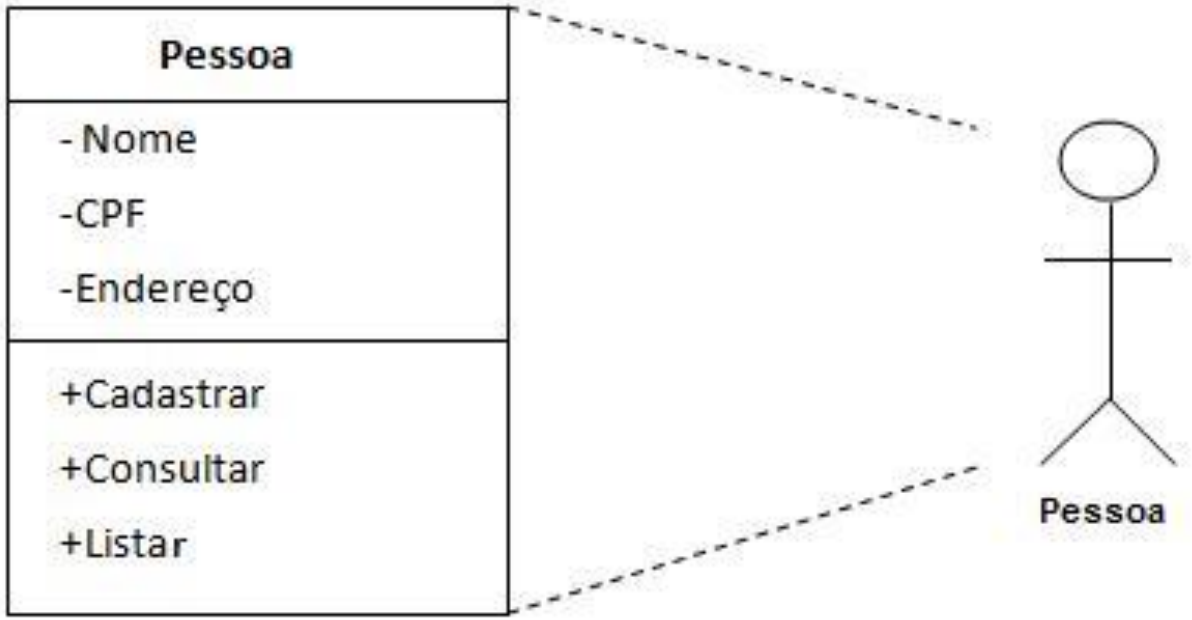
ATRIBUTOS OU PROPRIEDADES

- **Atributos ou propriedades** – representam as características de uma classe, isto é, peculiaridades que costumam variar de um objeto para outro;
- Do ponto de vista da UML, correspondem à segunda divisão do gráfico, contendo duas informações: nome do atributo e tipo de dado que o atributo armazena.



MÉTODOS, OPERAÇÕES OU COMPORTAMENTOS

- **Métodos, operações ou comportamentos** – representam as atividades, ações ou procedimentos que os objetos de uma classe podem executar;
- Do ponto de vista da UML, os métodos são armazenados na terceira divisão de uma classe.

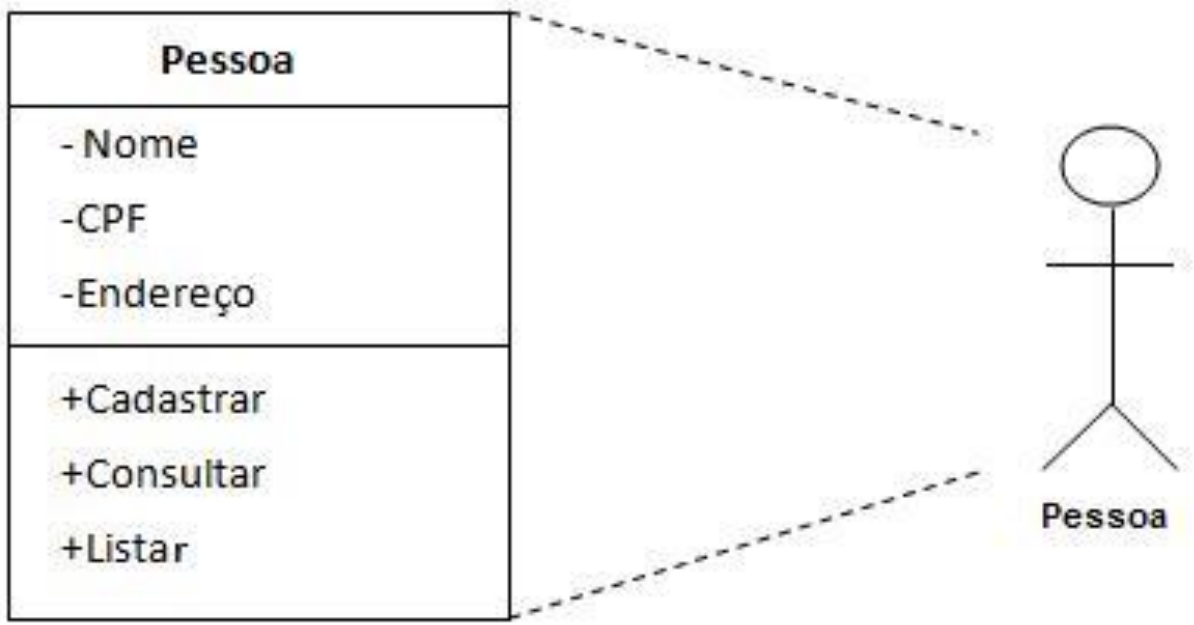


VISIBILIDADE

- **Visibilidade** – define de que maneira os métodos e atributos de um objeto podem ser acessados, ou até mesmo, se podem ser acessados. São elas:
  - ***Privada*** – somente objetos pertencente à classe podem acessá-los. Representada pelo símbolo menos (-);
  - ***Protegida*** – objetos das subclasses também terão acesso aos métodos e atributos da superclasse. Simbolizada por cerquilha (#);
  - ***Pública*** – o método ou atributo deste tipo de classe pode ser acessado por qualquer outro objeto de qualquer classe. Usa o símbolo mais (+);
  - ***Pacote*** – qualquer objeto em qualquer projeto pertencente ao pacote pode acessar métodos e propriedades com este nível de visibilidade. Símbolo é o til (~);

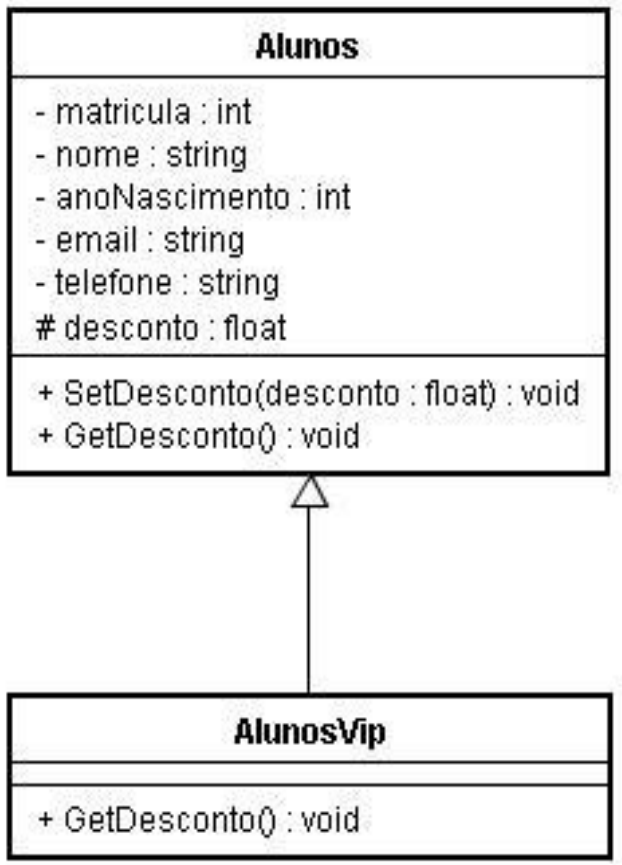
VISIBILIDADE

- Do ponto de vista da UML, a visibilidade é definida colocando-se, no gráfico, o sinal adequado antes do nome do método ou atributo.



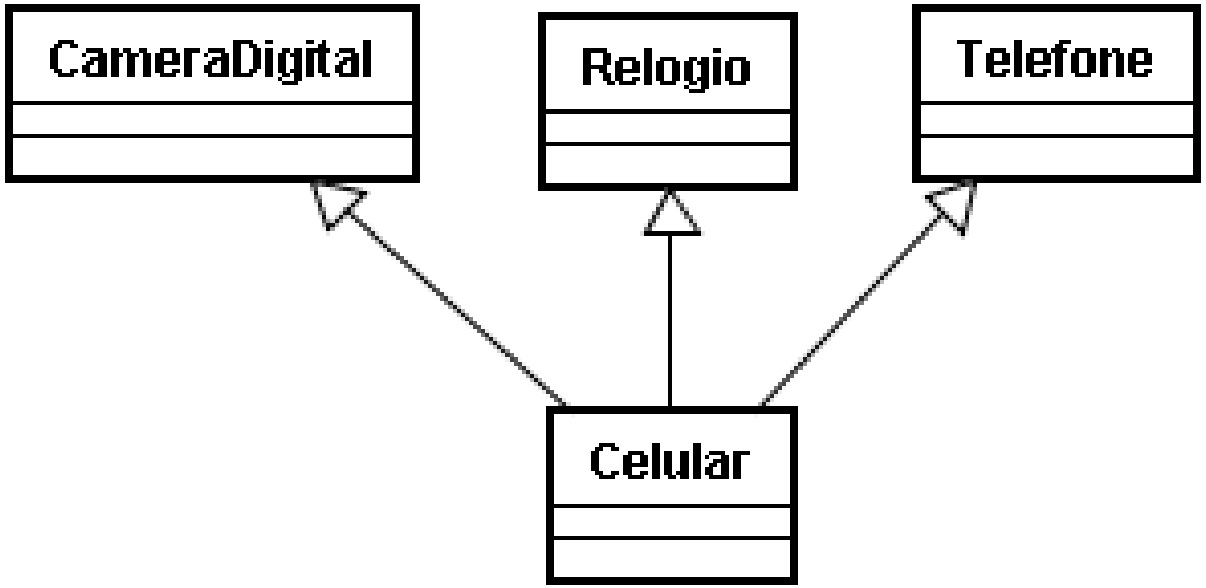
HERANÇA SIMPLES

- **Herança** – mecanismo pelo qual classes diferentes podem compartilhar um mesmo conjunto de características. Desta forma, uma classe pode aproveitar os métodos e atributos já definidos em outra classe;
- A classe que herda é chamada de classe derivada, classe-filha ou subclasse;
- A classe que origina a herança é chamada de classe ancestral, classe-mãe ou superclasse;
- Em OO, a herança impacta diretamente na reutilização de código, agilizando o processo de desenvolvimento e manutenção do sistema;
- Quando a subclasse herda de apenas uma classe, temos a herança simples.



HERANÇA MÚLTIPLA

- **Herança múltipla** – tipo de herança especial onde a classe-filha herda suas características de duas ou mais classes. Nem todas as linguagens de programação baseadas em objetos suportam este mecanismo.



POLIMORFISMO

- **Polimorfismo** – mecanismo que permite que um método herdado de uma superclasse seja redeclarado, na classe-filha. Embora semelhantes, ambos diferem, de alguma forma, em sua implementação;
- O polimorfismo permite a existência de dois ou mais métodos com mesmo nome associados a um objeto. É o sistema que decide qual dos dois será efetivamente invocado e executado.