**TECNOLOGIAS DE INFORMAÇÃO:**

## AGENDA 12

**MODELOS ORIENTADOS A OBJETOS (MOO) –** UML e Diagramas de classes. Diagramas de classe podem ser usados para **modelar objetos** que compõem o sistema, exibir relacionamentos e descrever o que esses objetos fazem. Geralmente são criados durante os estágios iniciais do projeto.

O Diagrama de classe não deve faltar em projetos orientados a objetos.

**UML (LINGUAGEM UNIFICADA DE MODELAGEM) –** Permite visualização, especificação, construção e documentação de artefatos de um sistema.

**IDENTIFICAÇÃO DE CLASSE –** Para identificar uma classe deve-se procurar itens com as mesmas informações (atributos) e comportamentos (métodos).

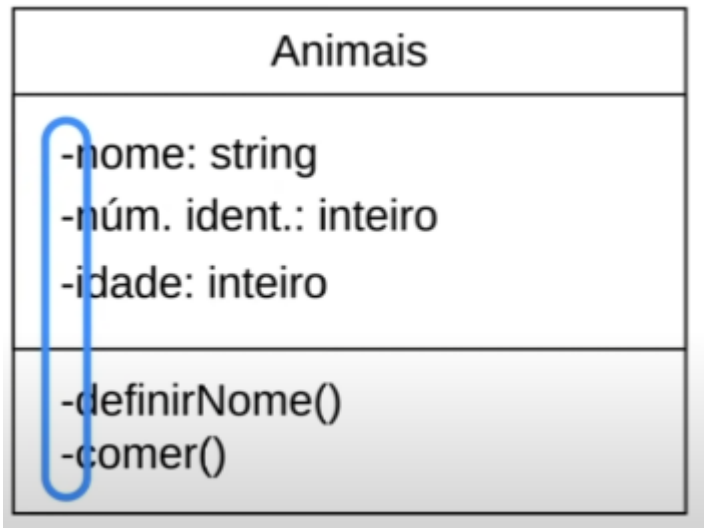
- **Associação:** é um relacionamento estrutural que especifica objetos de uma classe conectados a objetos de outra classe (representação: linha);
- **Agregação:** tipo especial de associação cujas informações de um objeto-todo precisar ser complementada pelas informações de um objeto-parte (representação: linha + losango);
- **Composição:** é uma variação da agregação e também representa uma relação de todo-parte. No entanto na composição o objeto-pai (todo) é responsável por criar e destruir suas partes. Não pode haver mais de uma associação de um objeto-parte a um objeto-pai na composição. (representação: linha + losango pintado). Quando um objeto secundário **não consegue existir** sem um objeto primário.
- **Herança (Especialização e Generalização):** identifica classes-mãe, denominadas gerais e classes filhas (especializadas). Relacionamento "é um tipo de". Exemplo: superclasse **pessoa** (atributos: nome, cpf) e (métodos: andar, falar) e classe **funcionário** (atributos: nome, cpf herdados / salário e ncarteira) e (métodos: trabalhar). (representação: seta)
- **Dependência:** Indica um grau de dependência entre uma classe e outra. (representação: seta tracejada).


**MULTIPLICIDADE –** É um número que vai na linha da relação para quantificar a relação. São restrições numéricas nos relacionamentos.

**DIAGRAMA DE CLASSE:** Representa um contexto abrangente do sistema, demonstrando as classes principais, abstratas, derivadas e comunicação entre as classes.

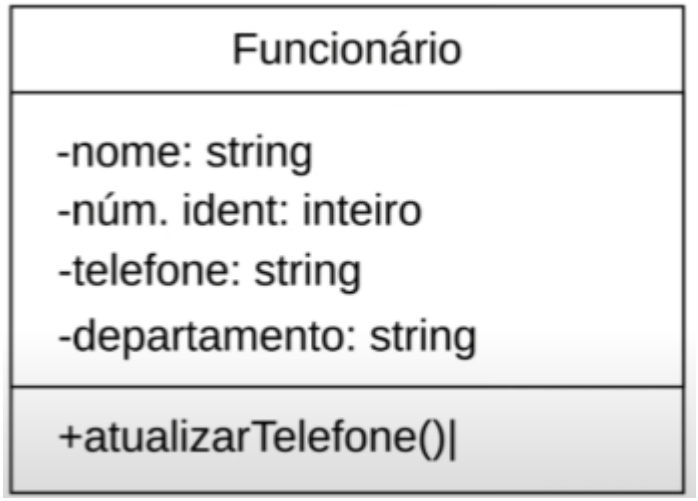
**ATRIBUTOS DE CLASSE –** É um pedaço significativo de dados que contém valores que descrevem cada instância de uma classe.

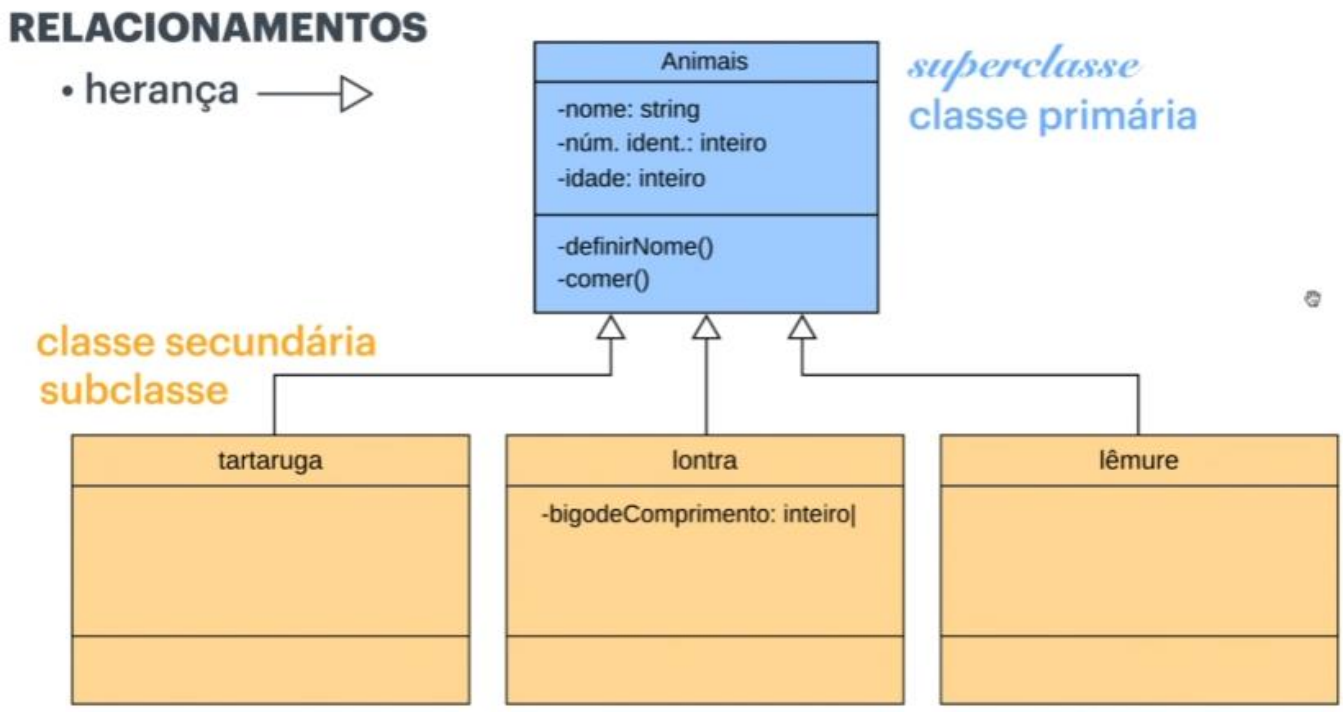
**MÉTODOS –** São operações ou funções. Permitem especificar as características comportamentais de uma classe.




**VISIBILIDADE:** Define a acessibilidade para um atributo ou método.

**1. PRIVADO (-):** Não podem ser acessados por qualquer outra classe.
**2. PÚBLIC (+):** Pode ser acessado por qualquer outra classe.
**3. PROTEGIDO (#):** Pode, ser acessado pela mesma classe ou uma subclasse.
**4. PACOTE/PADRÃO (~):** Pode ser usado por qualquer outra classe que esteja no mesmo pacote (pasta)



**CLASSE ABSTRATA (<<*nome da classe*>> *nome da classe)* –** Animal, no caso acima, é uma classe abstrata, pois ela foi criada apenas para simplificar as coisas e manter o código e manutenção do código mais simples. Abstração é uma maneira de concentrar apenas nos aspectos essenciais do nosso cenário.

**https://www.drawio.com/ (Draw.io)**

**https://www.lucidchart.com/pages (Lucidchart)**

**https://www.youtube.com/watch?v=rDidOn6KN9k&ab_channel=LucidSoftwarePortugu%C3%AAs**
**(Diagrama de classe Lucid Software Português – Youtube)**

**https://www.devmedia.com.br/orientacoes-basicas-na-elaboracao-de-um-diagrama-de-classes/37224**
**(devmedia blog)**