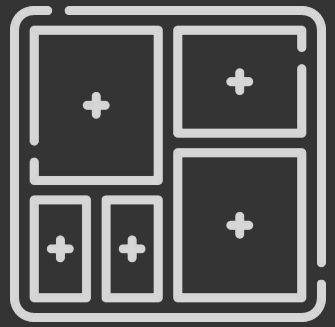


Relacionamentos entre Classes

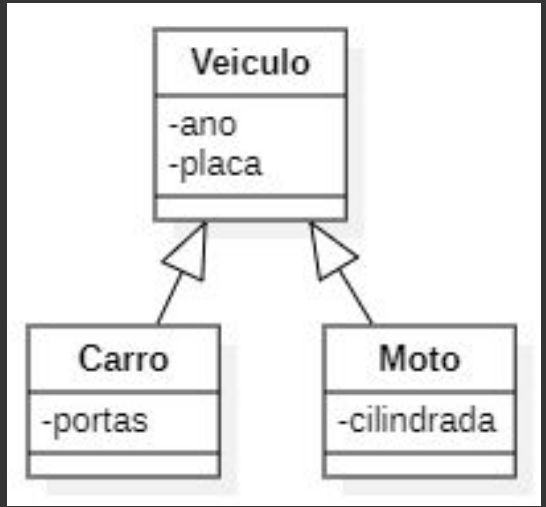
Relacionamentos entre objetos

- Objetos representam entidades do mundo real
- Entidades do mundo real raramente existem de forma isolada
  - Um **carro** tem **rodas**
  - Uma **pessoa** é amiga de outra **pessoa**
  - Um **quadrado** é uma **forma**
  - Uma **empresa** tem **funcionários**

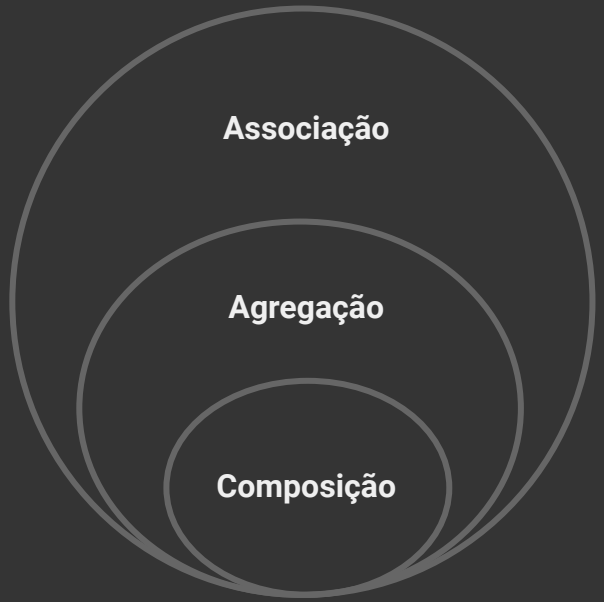
Portanto, é natural representarmos relacionamentos entre os objetos que correspondam a relacionamentos entre as respectivas entidades

Relacionamentos entre objetos

- Já vimos um tipo de relacionamento entre classes
- Herança
  - Relacionamento do tipo “é-um”
  - Hierarquia
  - Generalização / Especialização

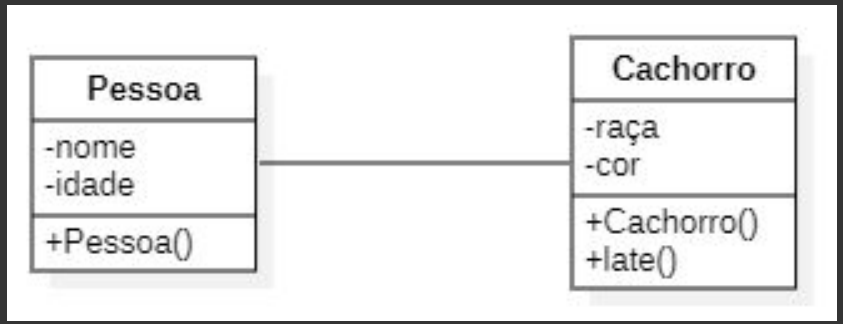


Outros relacionamentos entre objetos



Associação

- É um relacionamento bastante genérico que indica que existe uma conexão entre os objetos em questão
- Exemplo:
  - Um **pessoa** é dona (tutora) de **cachorros**
  - Um **usuário** tem uma **reserva**
  - Um **livro** possui uma **editora**



Associação

- Em geral, a associação é implementada de forma que uma classe torna-se atributo de outra classe
  - A classe **Pessoa** pode ter um **Cachorro** (ou uma lista de Cachorros) como atributo.
  - A classe **Cachorro** pode ter um tutor **Pessoa** (ou uma lista de Pessoas) como atributo.

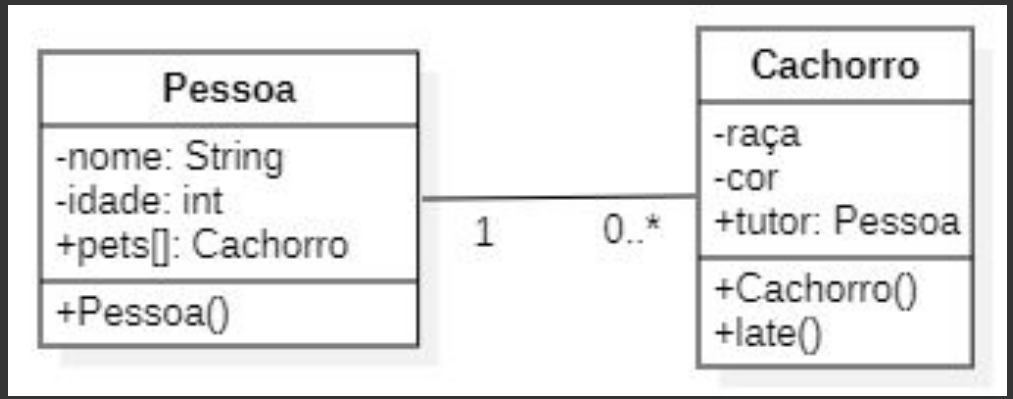
A escolha depende da **navegabilidade** e da **multiplicidade** do relacionamento

Quem enxerga quem

Número de ocorrências do relacionamento

Associação

- Um **pessoa** é dona (tutora) de **cachorros**
  - Uma possível modelagem:



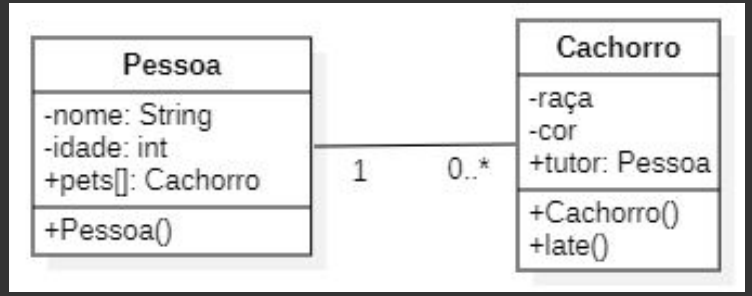
Associação

- Um **pessoa** é dona (tutora) de **cachor**
  - Uma possível modelagem:

Multiplicidade (ou Cardinalidade)
- Cada Pessoa tem 0 ou mais objetos cachorro
- Cada Cachorro é vinculado a uma única Pessoa

Associação sem setas: navegabilidade bidirecional
- Pessoa acessa seus objetos cachorro
- Cachorro acessa seu objeto Pessoa

Associação



- Um **pessoa** é dona (tutora) de **cachorros**
  - Implementação

Pessoa.java

```
class Pessoa{
    String nome;
    int idade;
    Cachorro pets[] = new Cachorro[10];
}
```

Cachorro.java

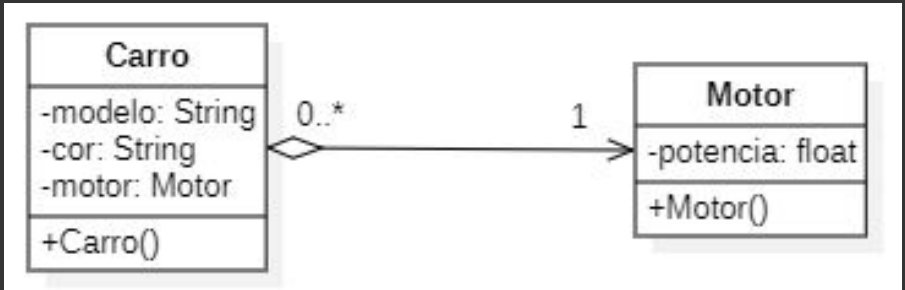
```
class Cachorro{
    String raca;
    String cor;
    Pessoa tutor;
}
```

Agregação

- É um caso especial de associação onde o elemento associado corresponde a uma parte do elemento principal.
- Relacionamento do tipo “tem um” ou “todo-parte” entre objetos
- As partes têm existência independente do todo
- Exemplo:
  - Um **carro** é constituído por (dentre outras partes), um **motor**
  - Uma **biblioteca** é um agregado de **livros**
  - Uma **playlist** é um conjunto de **músicas**

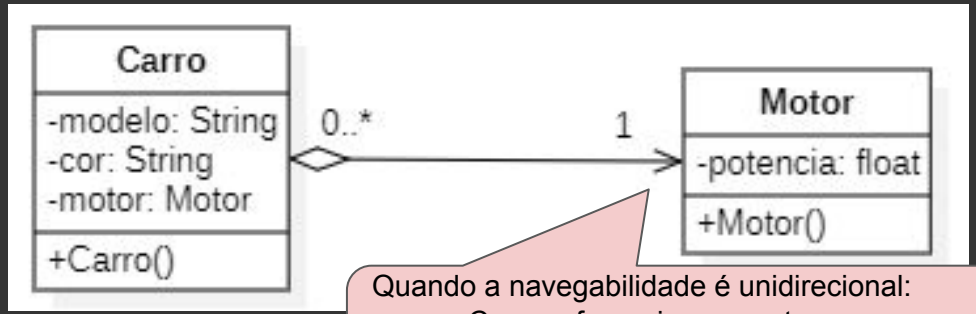
Agregação

- Um **carro** tem um **motor**
  - Motor é uma parte do carro
  - O conceito de motor pode existir sem o conceito de carro, ou ainda o mesmo tipo de motor pode estar em carros diferentes

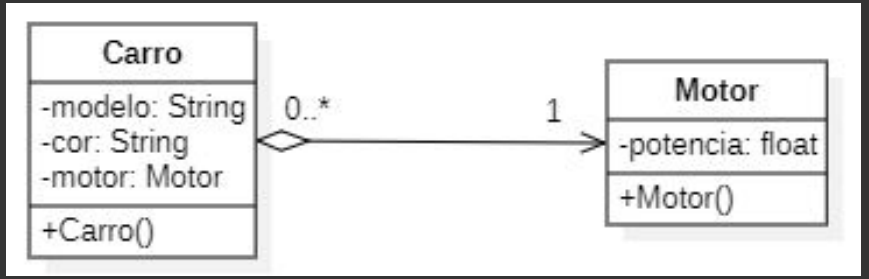


Agregação

- Um **carro** tem um **motor**
  - Motor é uma parte do carro
  - O conceito de motor pode existir sem o conceito de carro, ou ainda o mesmo tipo de motor pode estar em carros diferentes



Agregação



- Um **carro** tem um **motor**
  - Implementação

Carro.java

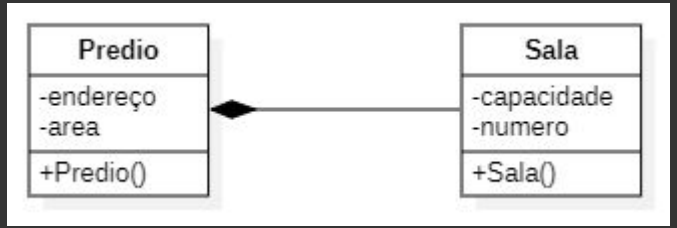
```
class Carro{
    String modelo;
    String cor;
    Motor motor;
}
```

Motor.java

```
class Motor{
    float potencia;
}
```

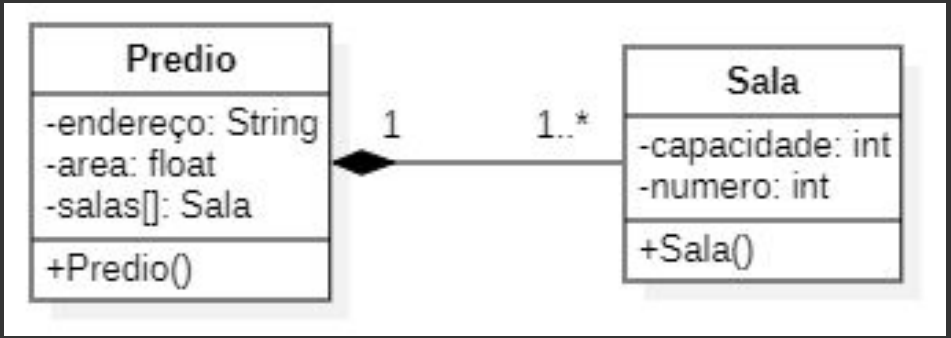
Composição

- É um caso especial de agregação onde as partes pertencem somente a um único todo
- Relacionamento do tipo “tem um” ou “todo-parte” entre objetos tal que, quando o objeto que é o todo deixa de existir, o objeto que é a parte também deixa de existir
- Exemplo:
  - Um **prédio** tem **salas**
  - Um **livro** tem **capítulos**
  - Uma **compra** a prazo possui **parcelas**

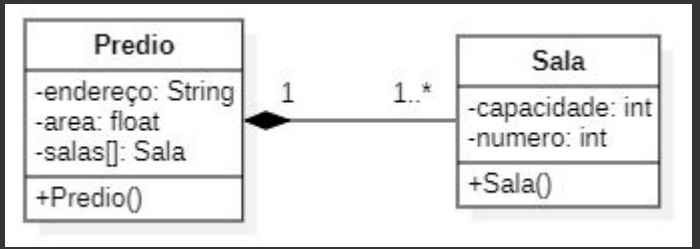


Composição

- Um **prédio** tem **salas**
  - Cada sala existe somente associada ao prédio ao qual pertence



Composição



- Um **prédio** tem **salas**
  - Implementação

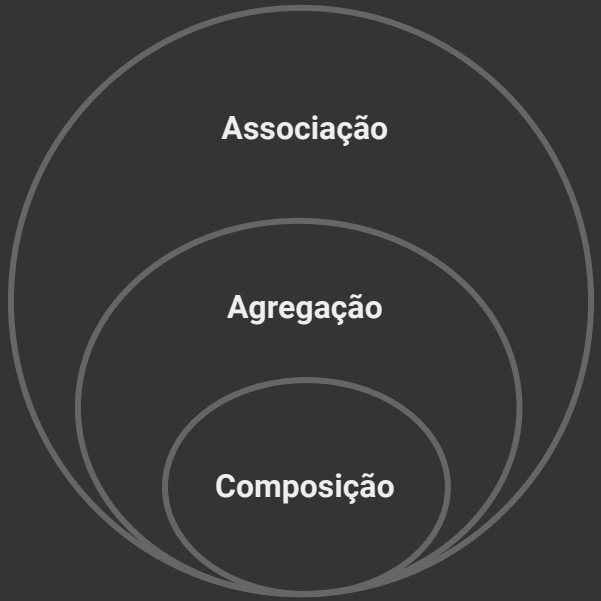
Predio.java

```
class Predio {
     String endereco;
     float area;
     ArrayList<Sala> salas = new ArrayList<Sala>();

     public void addSala(int capacidade, int numero){
       this.salas.add(new Sala(capacidade, numero));
     }
}
```

Sala.java

```
class Sala {
     int capacidade;
     int numero;

     public Sala(int c, int n){
          this.capacidade = c;
          this.numero = n;
     }
}
```

Resumo



Existe uma relação entre as classes

Existe uma relação do tipo *todo-parte* entre as classes

Existe uma relação do tipo *todo-parte* entre as classes e um objeto da classe *parte* não existe sem o objeto da classe *todo*

Resumo

- A implementação é semelhante (não existem instruções/comandos específicos para cada tipo de associação)
- As diferenças são semânticas e determinam determinadas restrições sobre como o sistema deve funcionar