PROJETO DE DADOS
USANDO ORIENTAÇÃO A OBJETOS

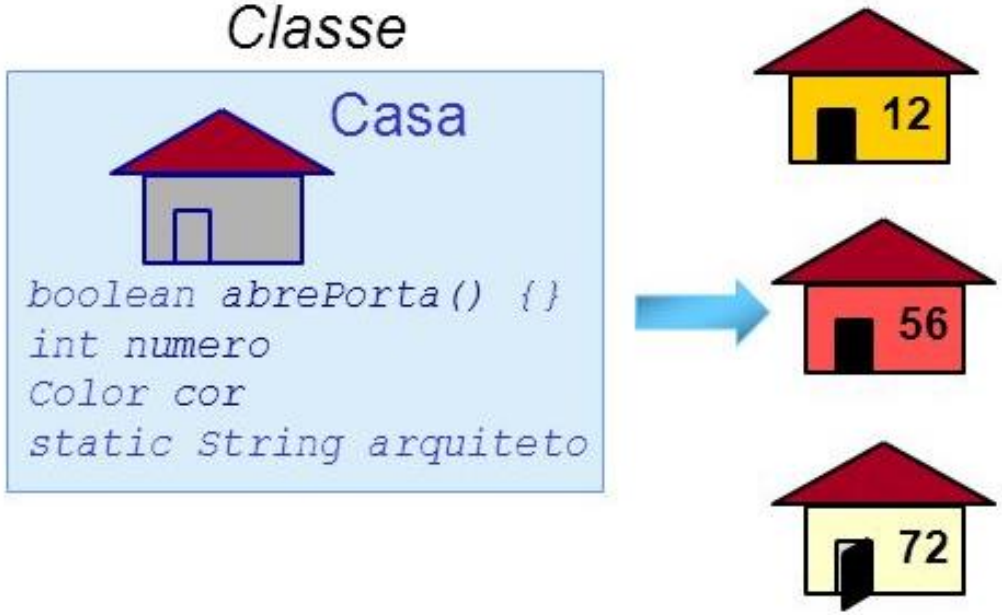
Projeto de Programas – PPR0001

Atividades Envolvidas

- Realizar a abstração de dados (visando o escopo do problema);
- Estudar e escolher as estruturas de dados que permitam a implementação mais adequada;
- Caracterizar a Abrangência dos dados:
  - De componente (local) ou parte do software (global);
  - Persistência de dados: uso de Banco de Dados , arquivos ou memória;

Objetos x Classes

- Qual a diferença entre objetos e classes?



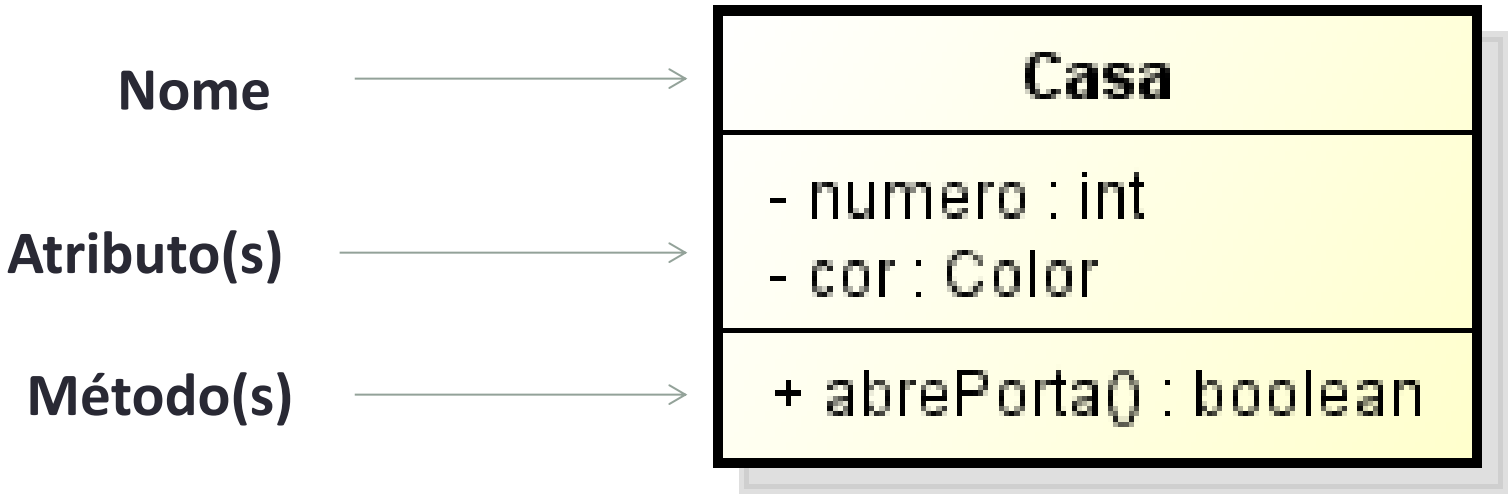
Classes

- Quais são os dados principais de uma classe?

Classes

- Quais são os dados principais de uma classe?
  - **Nome**
  - **Atributos**
  - **Métodos**

Diagrama de Classes



Classes

- Quais são os dados principais de uma classe?

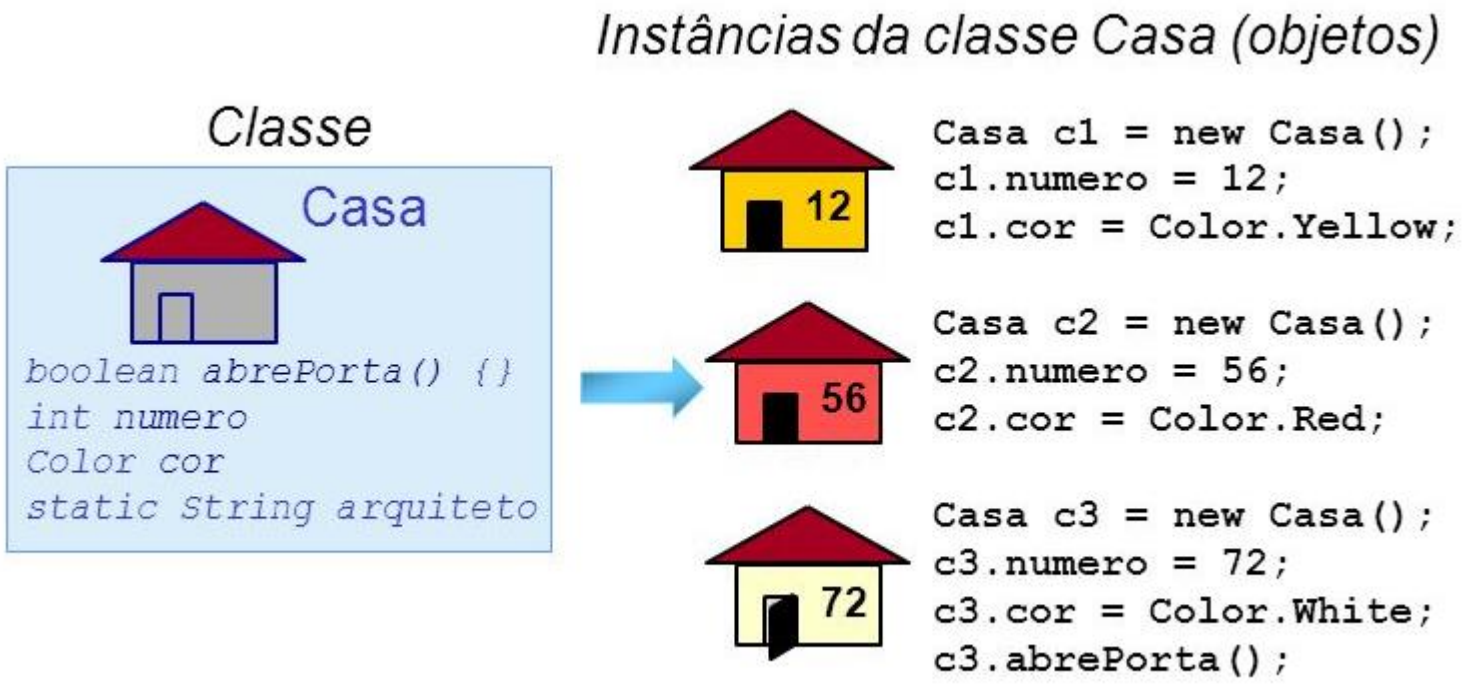
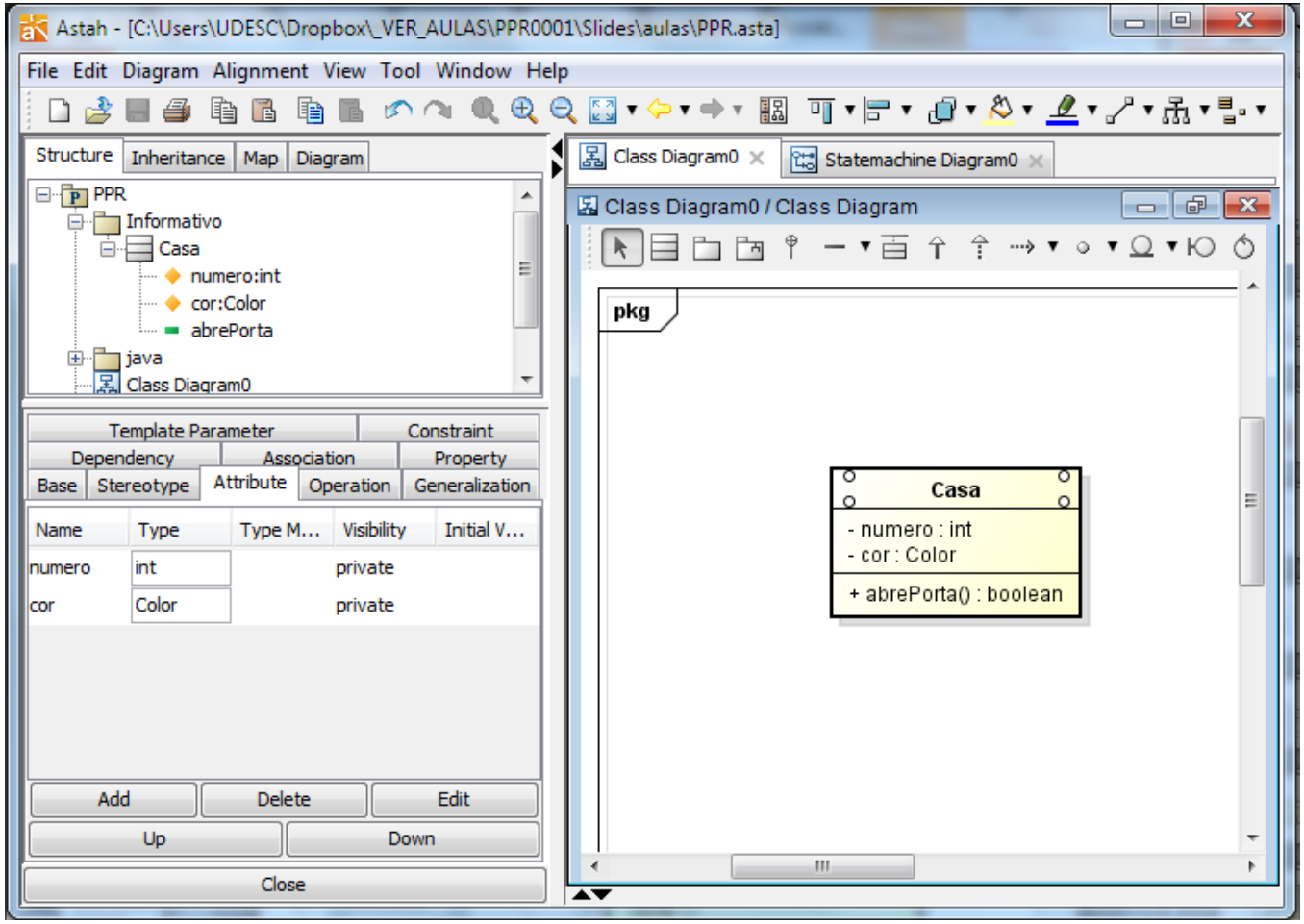


Diagrama de Classes

- Como criar um diagrama de classes no Astah?

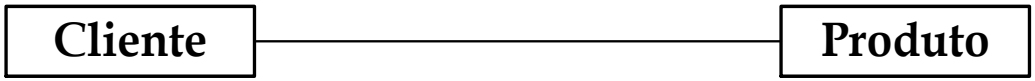


Relacionamentos entre Classes

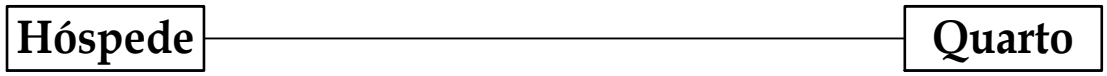
Associação

- Representa ligações possíveis entre objetos, destacando que objetos dessas classes poderão colaborar (troca de mensagem) para realizar alguma tarefa do sistema

(ex1: no domínio de vendas, um cliente compra produtos)



(ex2: em um hotel há vários hóspedes, assim como há vários quartos. Os hóspedes ocupam os quartos)

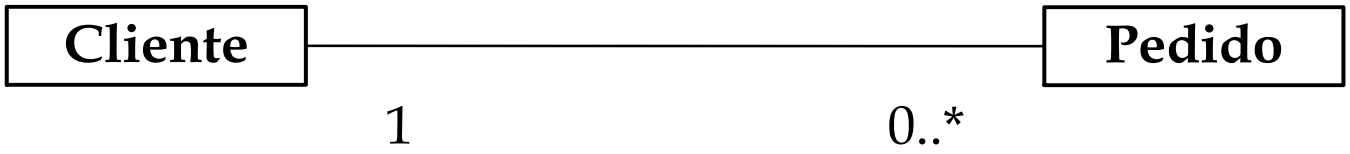


Relacionamentos entre Classes

Associação - Multiplicidade e Conectividade

- Podemos indicar limites inferiores e superiores

(ex1: cada cliente pode estar associado a diversos pedidos)



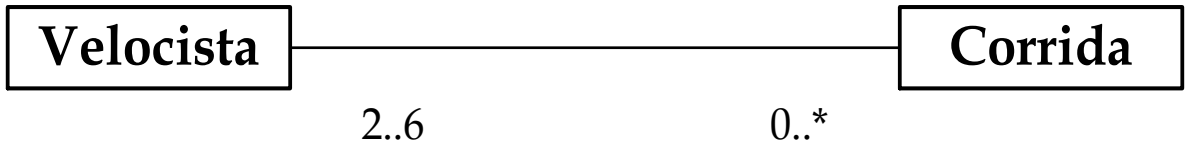
Nome	Simbologia
Apenas Um	1..1 (ou 1)
Zero ou Muitos	0..* (ou *)
Um ou Muitos	1..*
Zero ou Um	0..1
Intervalo Específico	$l_i..l_s$

Relacionamentos entre Classes

Associação - Multiplicidade e Conectividade

- Podemos indicar limites específicos para o problema

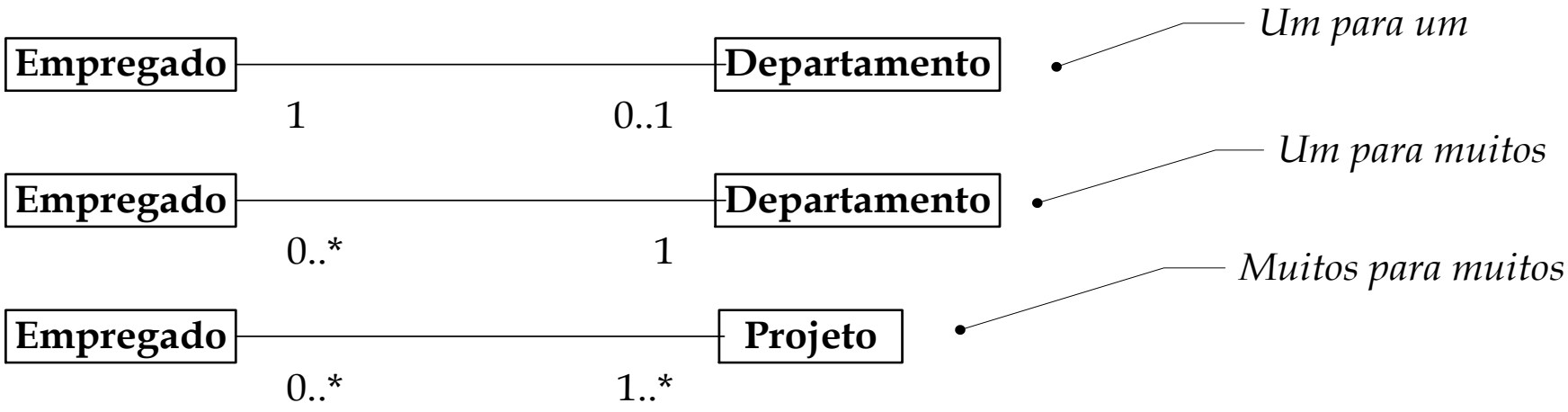
(ex2: toda corrida é composta por no mínimo 2 e no máximo 6 velocistas. Cada velocista pode participar de 0 a n corridas)



Relacionamentos entre Classes

Associação - Multiplicidade e Conectividade

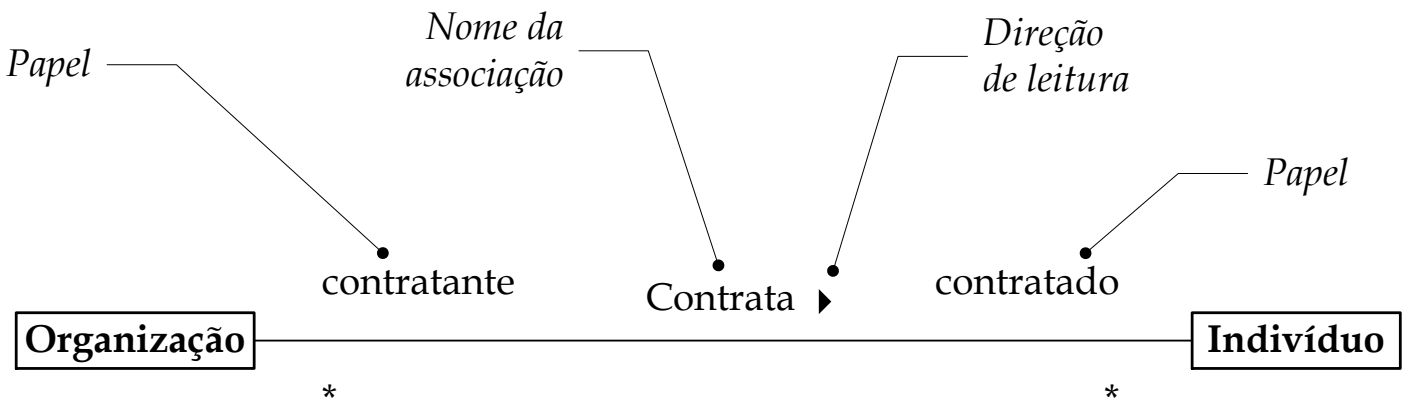
- Em geral existem três possibilidades distintas de conectividade:
  - Um para um
  - Um para muitos
  - Muitos para muitos



Relacionamentos entre Classes

Associação - Nome, direção e papéis

- Toda associação pode ser:
  - Obrigatória: multiplicidade dos dois lados sempre será >= 1
  - Opcional: multiplicidade de um dos lados pode ser 0.
- Para esclarecer melhor o significado de uma associação podemos indicar os seguintes elementos no diagrama:
  - Nome da associação
  - Direção da leitura
  - Papéis



Relacionamentos entre Classes

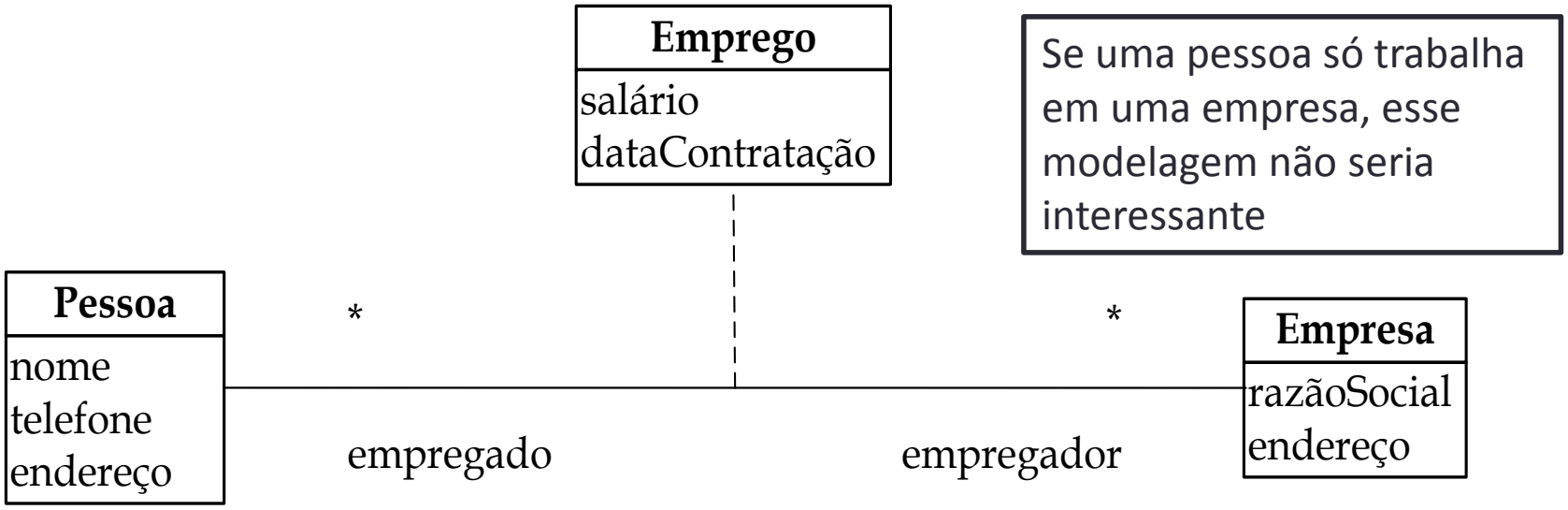
Associação - Nome, direção e papéis

- Atenção: usar em situações em que a associação não é óbvia.
- Pode haver mais de uma associação entre mesmas classes
  - Empregado **trabalha** em um departamento
  - Um departamento é **gerenciado** por um empregado.

Relacionamentos entre Classes

Associação - Classe associativa

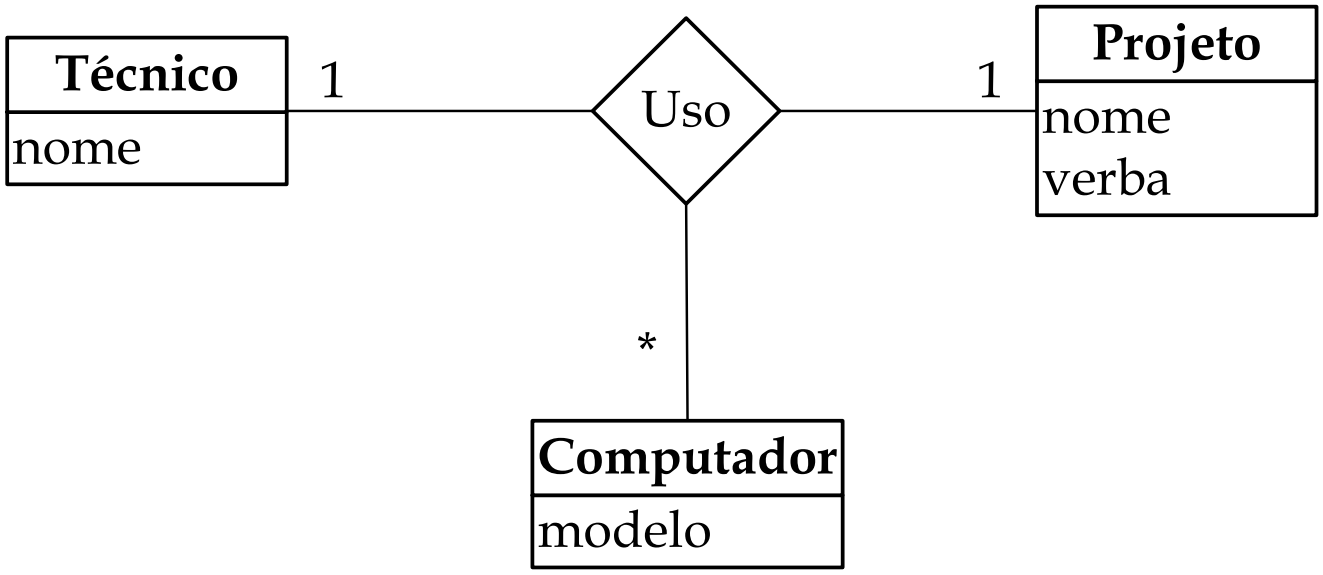
- É uma classe que está ligada a uma associação
- Normalmente necessário quando dados de uma associação entre duas classes precisam ser usados.



Relacionamentos entre Classes

Associação - Associações ternárias

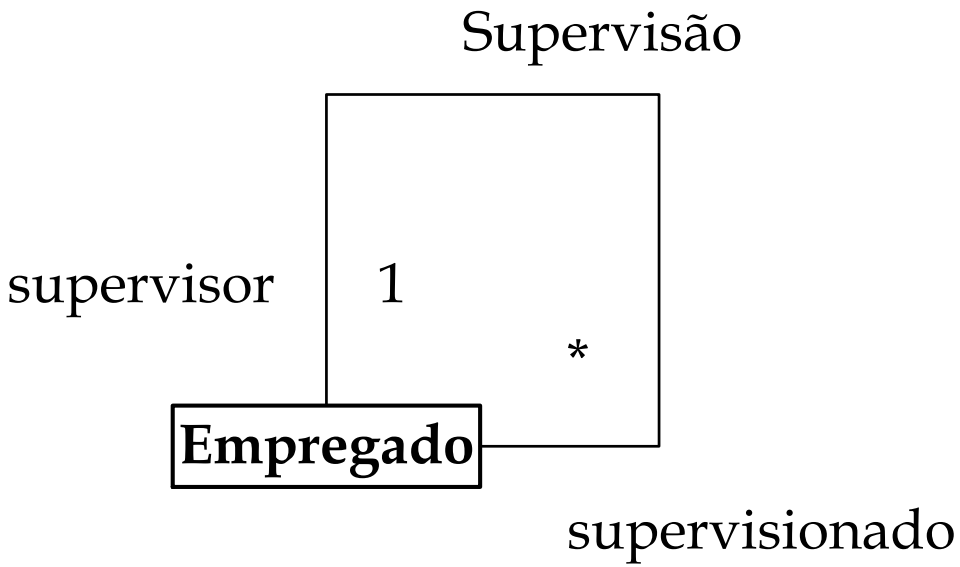
- Quando três classes estão presentes em uma mesma associação
- Pouco comum – podem ser representadas de maneiras diferentes



Relacionamentos entre Classes

Associação – Associações Reflexivas ou auto associações

- Quando uma classe se relaciona consigo mesma.
- Indica que um objeto da classe pode/deve se relacionar com outro objeto da mesma classe com papéis distintos.



Relacionamentos entre Classes

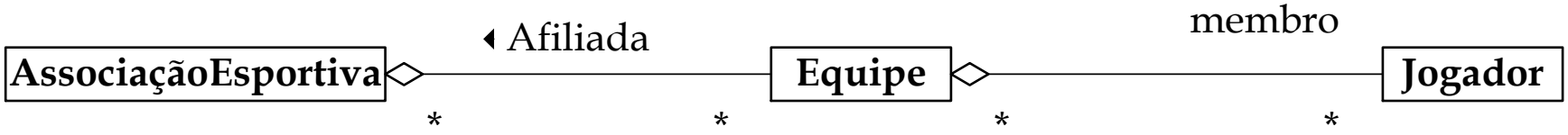
Associação – Agregação e Composição

- Um caso especial de associação – indicam uma relação *todo-parte* ou seja, um objeto está contido no outro, ou então um objeto contém o outro
- Podem ter as características de associações comuns
- São assimétricas: se A é parte de B, B não pode ser parte de A

- Perguntas úteis a se fazer
  - X tem um ou mais Y?
  - Y é parte de X?

Relacionamentos entre Classes

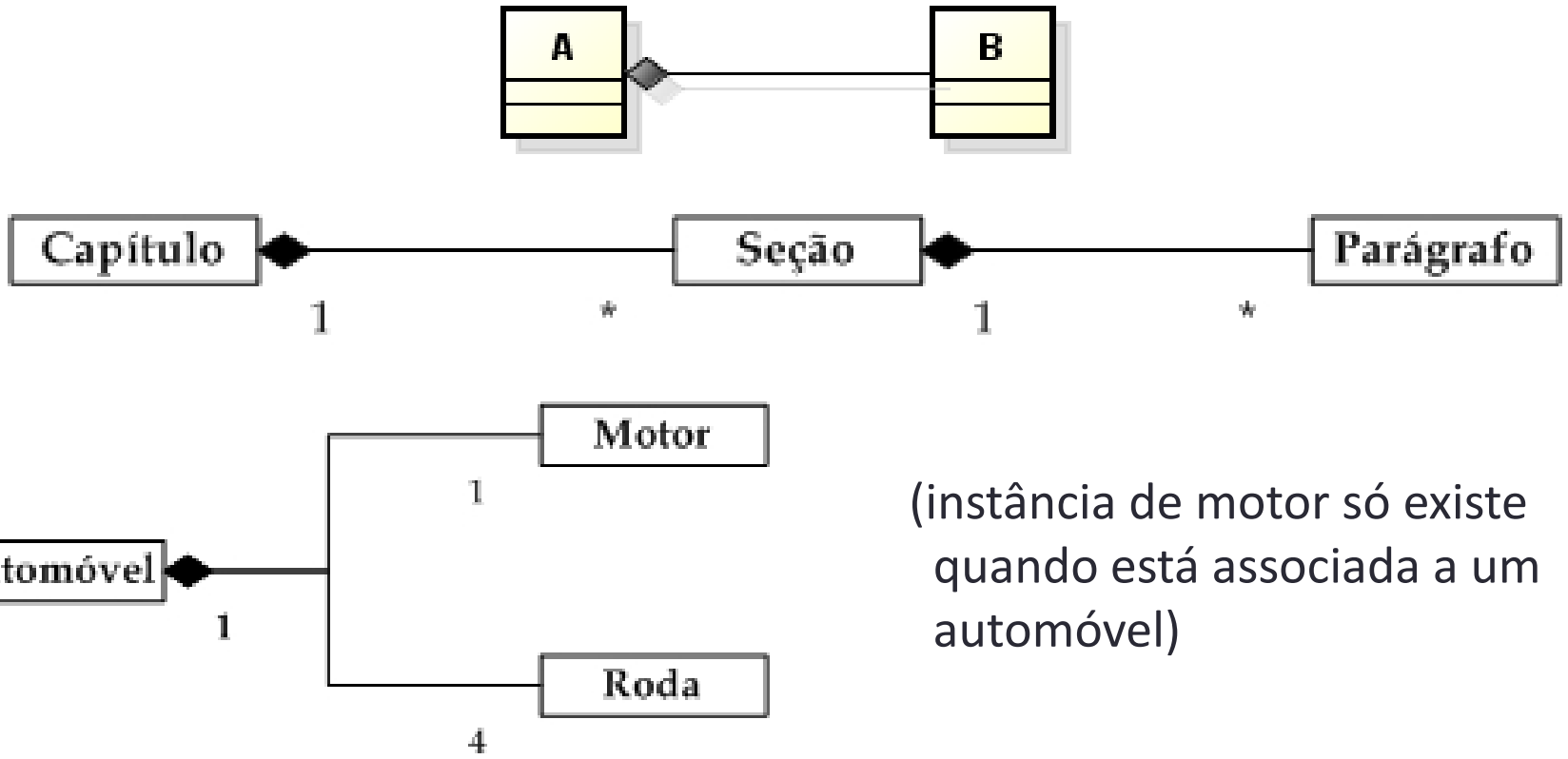
- **Agregação:** relacionamento optativo entre duas classes



(instância de jogador pode existir mesmo sem uma equipe)

Relacionamentos entre Classes

- **Composição:** relacionamento obrigatório entre duas classes



(instância de motor só existe quando está associada a um automóvel)

Relacionamentos entre Classes

Agregação

```
final class Car {
    private Engine motor;

    void setEngine(Engine motor) {
        this.motor = motor;
    }

    void move() {
        if (motor != null){
            motor.work();
        }
    }
}
```

Relacionamentos entre Classes

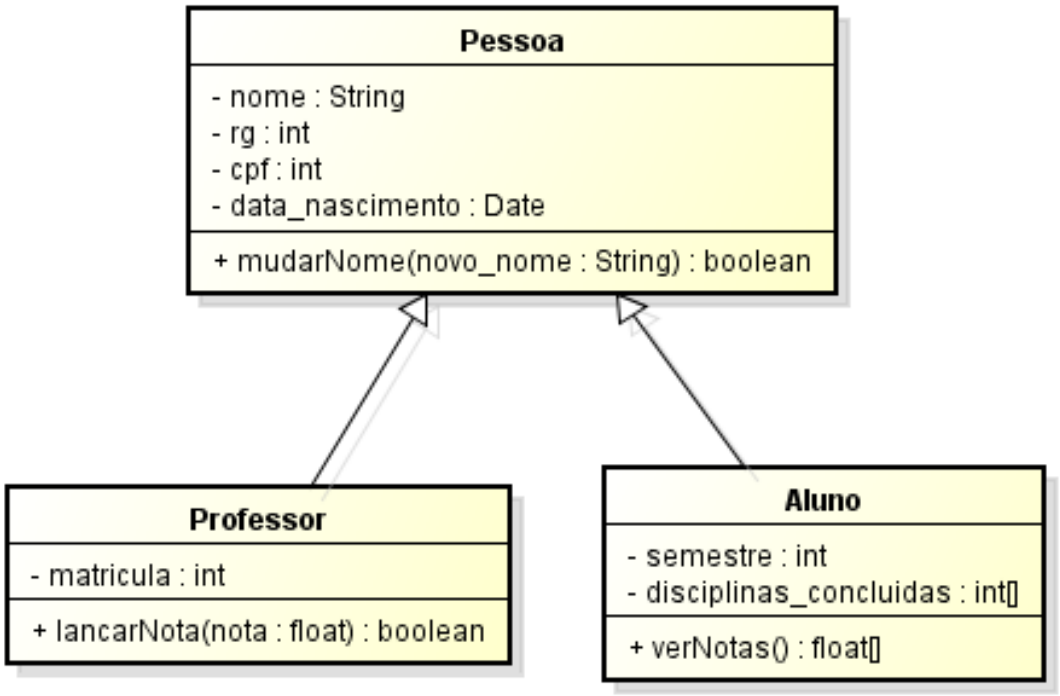
Composição

```
final class Car {
    private final Engine motor;
    Car(EngineSpecs specs) {
        motor = new Engine(specs);
    }

    void move() {
        motor.work();
    }
}
```

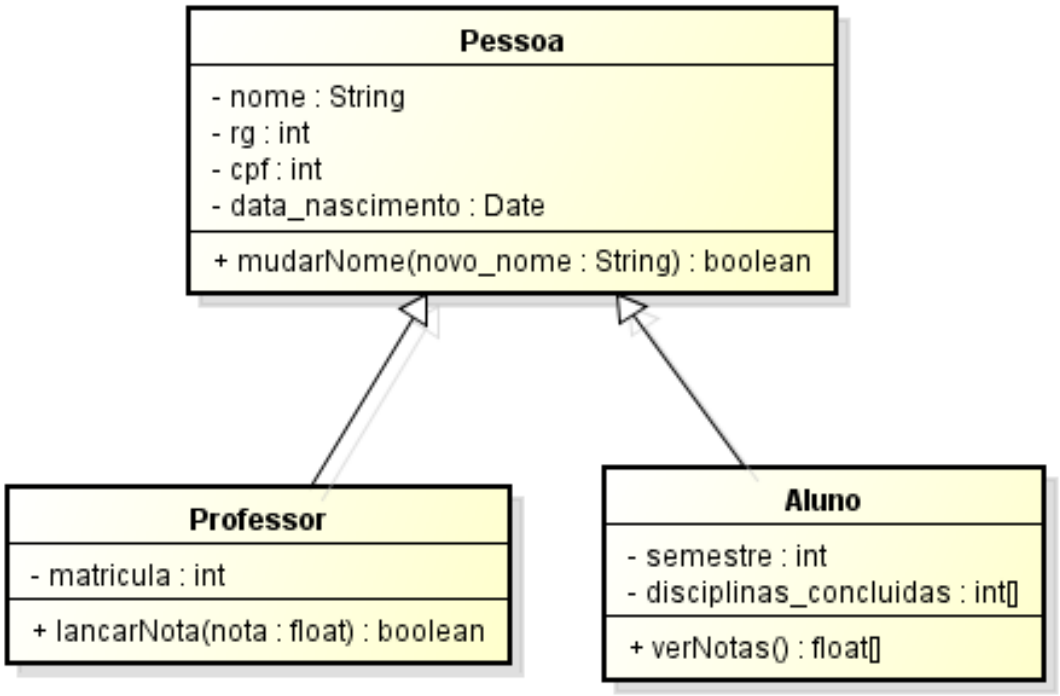
Relacionamentos entre Classes

- **Generalização** ou **Herança** também é representada nos diagramas de classe:
- **OBS:** uma generalização de ator ou caso de uso não implica necessariamente numa generalização de classes



Relacionamentos entre Classes

- **Generalização** ou **Herança** também é representada nos diagramas de classe:
- **OBS:** uma generalização de ator ou caso de uso não implica necessariamente numa generalização de classes



Atividade

- Leia o material de “Técnicas de identificação” de classes disponibilizada na página.

- Realize a identificação das classes e das suas associações para o seu problema. Cada aluno pega um caso de uso e faz a identificação das classes envolvidas. Depois o grupo faz a junção dos resultados.

Bibliografia

- **Básica:**

  BEZERRA, E. Princípios de Análise e Projetos de Sistemas com UML. Rio de Janeiro: Campus, 2003.

  PRESSMAN, R.S. Engenharia de Software. São Paulo: Makron Books, 2002.

  SOMMERVILLE, I. Engenharia de Software. São Paulo: Addison Wesley, 2003.

- **Complementar:**

  WARNIER, J. Lógica de Construção de Programas. Rio de Janeiro: Campus, 1984.

  JACKSON, M. Princípios de Projeto de Programas. Rio de Janeiro: Campus, 1988.

  PAGE-JONES, M. Projeto Estruturado de Sistemas. São Paulo: McGraw-Hill, 1988.