Programação I

Associações e Referências

Samuel da Silva Feitosa

Aula 8

Introdução

- **Associação**, **agregação** e **composição** são conceitos de extrema importância dentro da **Programação** e **Modelagem** Orientada a Objetos.
- Na POO, um software é construído com inspiração em conceitos do mundo real, os quais podem ser relacionados.
  - Exemplo: Empresa e Funcionários.
- Classes e objetos são relacionados através de associações, agregações e composições.

Associação

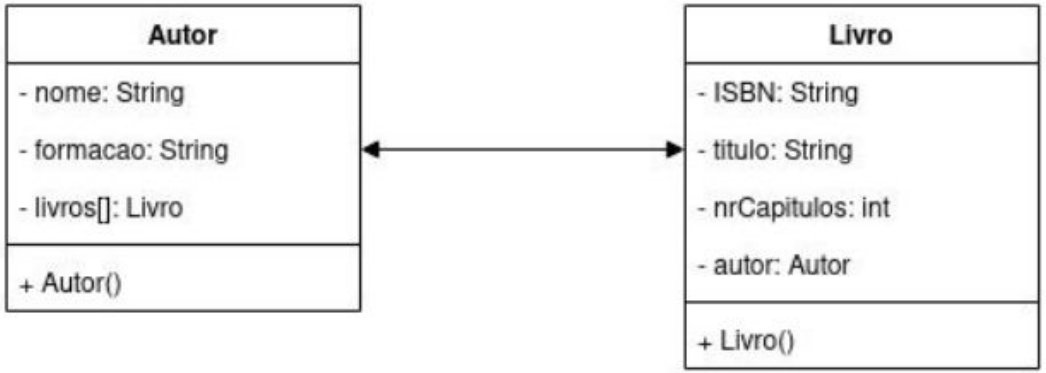
- É um **relacionamento** simples entre classes.
  - Por exemplo: **Autor** e **Livro**.
- Muitas das vezes é implementado de forma que uma classe torna-se **atributo** de outra classe.
  - A classe Autor pode ter um Livro (ou uma lista de livros) como atributo.
  - A classe Livro pode ter um Autor (ou uma lista de autores) como atributo.
- Outros exemplos:
  - Animal de estimação e seu dono, Empresa e seu endereço, etc.

Associação - Exemplo

- Classes Autor e Livro e seu diagrama UML.

```java
public class Autor {
    String nome;
    String formacao;
    Livro livros[];
}
```

```java
public class Livro {
    String ISBN;
    String titulo;
    int nrCapitulos;
    Autor autor;
}
```



Agregação

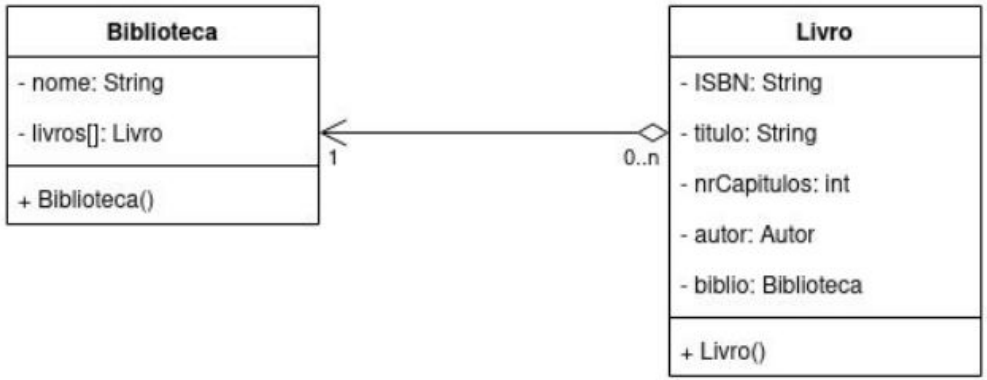
- Agregação é um tipo especial de associação onde o **elemento associado corresponde a uma parte** do elemento principal.
- As classes devem ter a característica de “**todo-parte**”, e a “parte” **pode estar contida em outros** elementos do “todo”.
  - Exemplo: Biblioteca e Livro.
- **Importante**: o conceito de Livro **pode existir sem** o conceito de Biblioteca, ou o mesmo Livro **pode estar** em diferentes bibliotecas.

Agregação - Exemplo

- Classes Biblioteca e Livro e seu diagrama UML.

```java
public class Biblioteca {
    String nome;
    Livro livros[];
}
```

```java
public class Livro {
    String ISBN;
    String titulo;
    int nrCapitulos;
    Autor autor;
    Biblioteca biblio;
}
```



Composição

- Composição é um tipo especial de associação, onde as **“partes” pertencem somente a um único “todo”**.
- **Não faz sentido** pensar em um objeto da classe principal **sem os objetos que o compõem**.
- O **“todo” só existe** enquanto as **“partes” também existem**, e vice-versa.
  - Exemplo: Livro e Capítulo.
- Em geral, um capítulo deve estar em um livro.

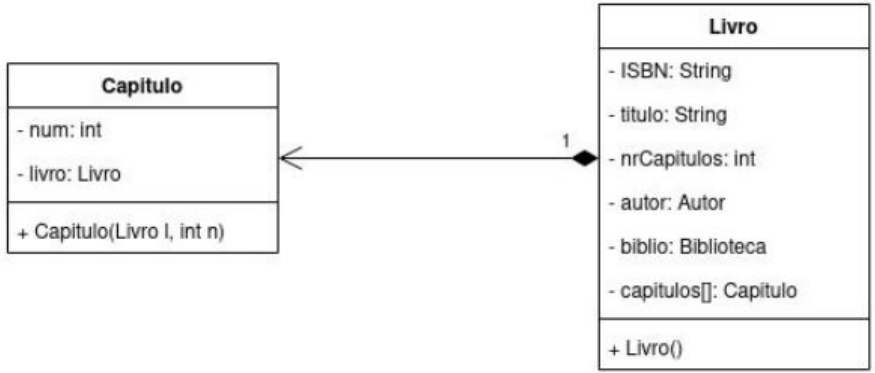
Composição - Exemplo

- Classes Capítulo e Livro e seu diagrama UML.

```java
public class Capitulo {
    int num;
    Livro livro;

    Capitulo(Livro livro, int num) {
        this.num = num;
        this.livro = livro;
    }
}
```

```java
public class Livro {
    String ISBN;
    String titulo;
    int nrCapitulos;
    Autor autor;
    Biblioteca biblio;
    Capitulo capitulos[];
}
```

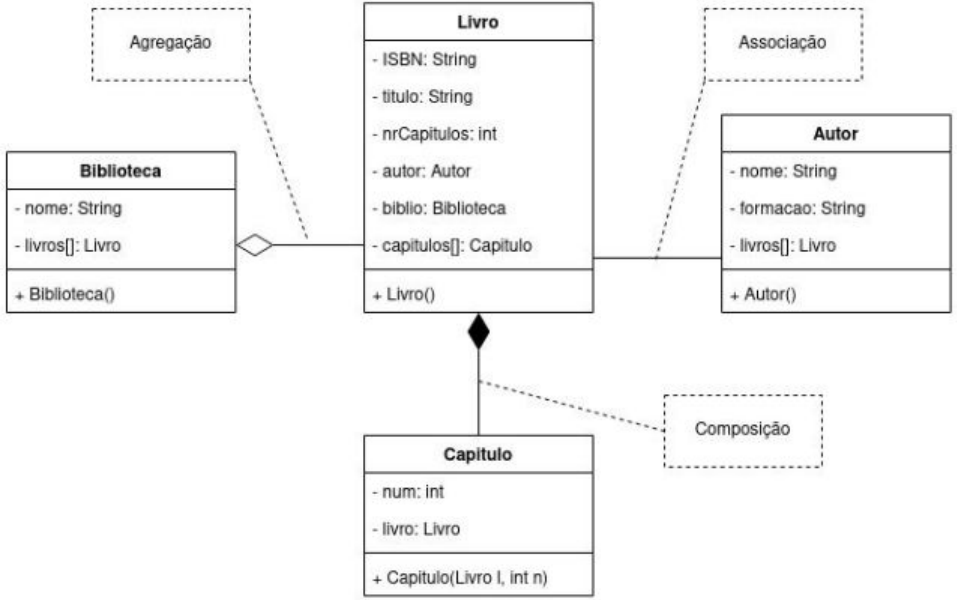


Na prática

- Percebam que o código usado para representar as associações, agregações e composições **são basicamente os mesmos**.
  - O objeto relacionado vira um atributo na classe.
- Em geral, estes relacionamentos são representados da **mesma forma** em linguagens de programação orientada a objetos.

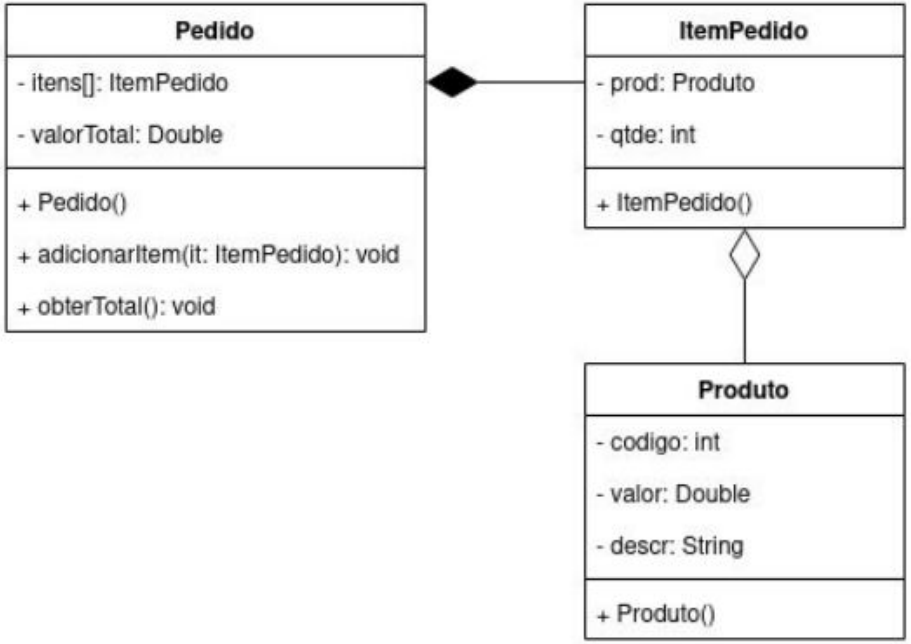
Representação UML

- Como vimos, em UML é possível representar de forma mais clara os relacionamentos.



Outro Exemplo - Pedido

- Diagrama UML para representar um sistema com Produtos e Pedidos.



Associações no contexto de um Banco (1)

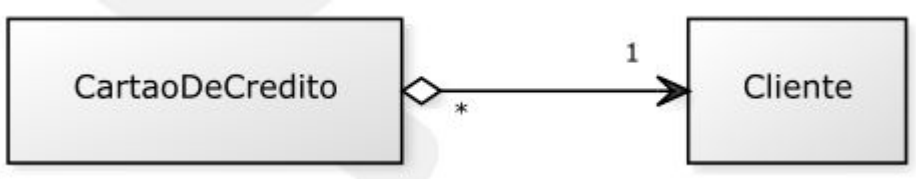
- Todo cliente do banco pode adquirir um cartão de crédito.
  - Dentro do sistema do banco, deve existir um objeto que represente o cliente e outro que represente o cartão de crédito.
  - Para expressar a relação entre o cliente e o cartão de crédito, algum vínculo entre esses dois objetos deve ser estabelecido.

Associações no contexto de um Banco (2)

- Duas classes deveriam ser criadas:
  - Uma para definir os atributos e métodos dos clientes e outra para os atributos e métodos dos cartões de crédito.
  - Para expressar o relacionamento entre cliente e cartão de crédito, podemos adicionar um atributo do tipo Cliente na classe CartaoDeCredito.

```
public class CartaoCredito {
    public int numero;
    public String dataValidade;
    public Cliente cliente;
}
```

```
public class Cliente {
    int codigo;
    String nome;
}
```



Referências a objetos como parâmetros

- Da mesma forma que podemos passar valores primitivos como parâmetro para um método ou construtor, também podemos passar valores não primitivos (referências).
- Considere um método na classe Conta que implemente a lógica de transferência de valores entre contas.

Referências a objetos como parâmetros

- Esse método deve receber como argumento, além do valor a ser transferido, a referência da conta que receberá o dinheiro.

```
public boolean transfere(Conta destino, double valor) {
    if (this.saca(valor)) {
        destino.deposita(valor);
        return true;
    }

    return false;
}
```

- Na chamada do método transfere(), devemos ter duas referências de contas: uma para chamar o método e outra para passar como parâmetro.
  - Quando a variável destino é passada como parâmetro, somente a referência armazenada nessa variável é enviada para o método transfere() e não o objeto em si.

Considerações Finais

- Conceitos importantes a respeito do relacionamento entre diferentes classes e objetos.
- Associação, agregação e composição são implementados de forma similar em linguagens de programação orientadas a objetos.
- É possível identificar estes relacionamentos através de diagramas UML.

Exercício

- Implemente as classes Produto, Pedido e ItemPedido conforme a especificação no slide anterior.
  - Você precisará implementar alguns atributos e/ou métodos privados para controlar os itens de um pedido.
- Implemente o programa principal criando alguns produtos, adicionando alguns itens ao pedido, e obtendo o valor total da compra.