Programação Orientada a Objetos

GCT052 - Aula 3.1 - Relações entre Classes

Prof. Dr. Bento Rafael Siqueira

Universidade Federal de Lavras
Departamento de Ciência da Computação

29 de outubro de 2025

Sumário

Por que Relações entre Classes?

- Classes não existem isoladas no mundo real
- Sistemas complexos requerem **interação** entre componentes
- As relações definem **como** os objetos se relacionam
- Importante para modelar sistemas reais corretamente

Principais Tipos de Relação

1. **Associação**: Relação genérica entre classes
2. **Agregação**: Relação "tem um"onde a parte pode existir sem o todo
3. **Composição**: Relação "tem um"onde a parte não existe sem o todo

Associação

Definição

Relacionamento genérico entre duas classes independentes, onde uma classe usa outra.

- Classe A usa ou conhece Classe B
- São **independentes**: existem separadamente
- Sem relação de "propriedade"
- A relação é **bidirecional** ou **unidirecional**

Exemplo do Mundo Real

- `Professor` $\leftrightarrow$ `Aluno` (bidirecional)

Associação - Exemplo em C#

```
1 public Professor(string nome)  Nome = nome; Alunos = new List<Aluno>();
2 public void AdicionarAluno(Aluno aluno)  Alunos.Add(aluno);
3 // Classe Aluno public class Aluno  public string Nome  get; set;  public string
  Matricula  get; set;
4 public Aluno(string nome, string matricula)  Nome = nome; Matricula = matricula;
```

Associação - Notação UML

Agregação

Definição

Relação "tem um"onde a parte pode existir independentemente do todo.

- Representa um relacionamento **mais forte** que associação
- Relação de "propriedade parcial"
- A parte **pode sobreviver** sem o todo
- Destruição do todo **não destrói** as partes

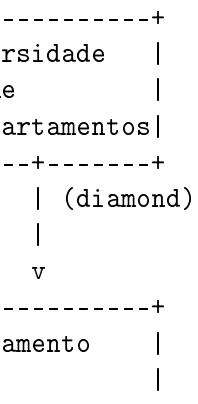
Exemplo do Mundo Real

- `Universidade "tem"Departamentos`

Agregação - Exemplo em C#

```
5 public Departamento(string nome, string sigla)  Nome = nome; Sigla = sigla;
6 // Classe Universidade (todo) public class Universidade  public string Nome  get; set;
  public List<Departamento> Departamentos  get; set;
7 public Universidade(string nome)  Nome = nome; Departamentos = new List<Departamento>();
8 public void AdicionarDepartamento(Departamento dept)  Departamentos.Add(dept);
```

Agregação - Notação UML



Composição

Definição

Relação "tem um"onde a parte **não pode existir** sem o todo.

- Relação **mais forte** que agregação
- Relação de "propriedade completa"
- A parte **não sobrevive** sem o todo
- Destruição do todo **destrói** as partes

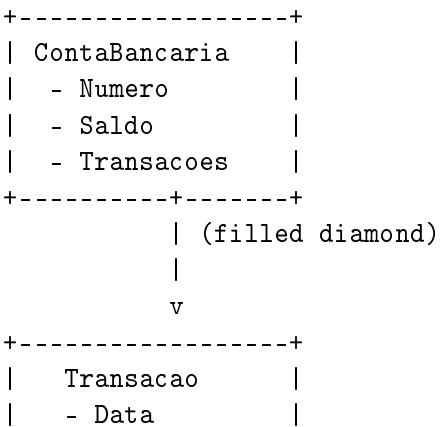
Exemplo do Mundo Real

- `Casa` "tem" `Salas` - Se a casa for demolida, as salas também são destruídas

Composição - Exemplo em C#

```
9  public Transacao(DateTime data, double valor, string tipo)  Data = data; Valor = valor;
   Tipo = tipo;
10 // Classe ContaBancaria (todo) public class ContaBancaria  public string Numero  get;
   set;  public double Saldo  get; set;  public List<Transacao> Transacoes  get; set;
11 public ContaBancaria(string numero)  Numero = numero; Saldo = 0; Transacoes = new
   List<Transacao>();
12 public void AdicionarTransacao(Transacao trans)  Transacoes.Add(trans); Saldo +=
   trans.Valor;
```

Composição - Notação UML



Comparação das Relações

Aspecto	Associação	Agregação	Composição
Relação	Usa/Conhece	"Tem um"	"Tem um"
Dependência	Independe	Leve	Forte
Existência	Totalmente independente	Parte pode existir sem	Parte não existe sem
Lifecycle	Separado	Separado	Controlado pelo todo
UML	Linha simples	Losango vazio	Losango preenchido
Exemplo	Professor-Aluno	Universidade-Dept	Casa-Sala

Regra de Ouro

- **Associação**: "usa"

Decidindo o Tipo de Relação

Perguntas para Decidir

1. **As classes podem existir independentemente?**
   - Sim → Associação
   - Não → Composição ou Agregação
2. **A parte tem sentido sem o todo?**
   - Sim → Agregação
   - Não → Composição
3. **O todo controla a existência da parte?**
   - Sim → Composição
   - Não → Agregação

Exemplo Completo 1: Biblioteca

Sistema de Biblioteca

- `Biblioteca` - Classe principal
- `Livro` - Pode existir sem biblioteca? **Sim** → Agregação
- `Pessoa` - Usa a biblioteca? **Sim** → Associação
- `Emprestimo` - Existe sem biblioteca? **Não** → Composição

```
1 rivate List<Livro> livros; // Agrega
2 ção private List<Pessoa> pessoas; // Associação private List<Emprestimo> emprestimos; // Composição
3 public class Livro   // Existe independente
4 public class Pessoa   // Existe independente
5 public class Emprestimo   // Não existe sem Biblioteca
```

Exemplo Completo 2: Sistema Bancário

Sistema Bancário

- `Banco` - Classe principal
- `Cliente` - Relação? **Associação**
- `Conta` - Relação? **Agregação** (conta pode fechar mas cliente existe)
- `Extrato` - Relação? **Composição** (extrato não existe sem conta)
- `Cartao` - Relação? **Composição** (cartão não existe sem conta)

```
6 public void AdicionarCliente(Cliente cliente)
7 public class Conta  private List<Transacao> transacoes; // Composição private Cartao cartao; // Composição
```

Boas Práticas no Uso de Relações

1. Composição para Dados Críticos

- Use composição para dados que devem ser sempre consistentes
- Exemplo: Transações de uma conta bancária

2. Agregação para Compartilhamento

- Use agregação quando vários objetos podem compartilhar partes
- Exemplo: Vários departamentos podem ter o mesmo professor

3. Associação para Colaboração

- Use associação quando classes trabalham juntas mas são independentes

Armadilhas Comuns

Erro 1: Usar Composição no Lugar de Agregação

- Errado: Pensar que toda relação "tem um"é composição
- Certo: Avalie se a parte sobrevive ao todo

Erro 2: Confundir Associação com Agregação

- Errado: Toda vez que há uma lista de objetos
- Certo: Agregação implica relação de "propriedade"

Erro 3: Não Considerar o Lifecycle

- Errado: Apenas olhar o código, não o significado

Exercício Proposto

Sistema de Restaurante

Desenhe as classes e defina os tipos de relacionamento para:

- `Restaurante` - O estabelecimento
- `Garcon` - Pessoal do restaurante
- `Cliente` - Quem come no restaurante
- `Cardapio` - Menu disponível
- `Prato` - Item do cardápio
- `Pedido` - Ordem feita pelo cliente
- `ItemPedido` - Item específico em um pedido

Resumo

Associação

- Relação mais fraca
- Classes independentes
- Representação: Linha simples

Agregação

- Relação intermediária
- Parte pode existir sem o todo
- Representação: Losango vazio

Fim

Qualquer Dúvida?

Contato: bento.siqueira@ufla.br

Material: github.com/californi/disciplinas-ufla