Programação Orientada a Objetos

GCT052 - Aula 2.3 - Classes e Objetos

Prof. Dr. Bento Rafael Siqueira

Universidade Federal de Lavras
Departamento de Ciência da Computação

10 de setembro de 2025

Sumário

O que são Classes e Objetos?

- **Classe**: É um molde ou template que define as características e comportamentos de um tipo de objeto
- **Objeto**: É uma instância específica de uma classe, com valores únicos para seus atributos
- **Analogia**: Classe = Planta de uma casa, Objeto = Casa construída

Exemplo do Dia a Dia:

- **Classe**: Carro
- **Atributos**: Marca, modelo, cor, ano, placa
- **Métodos**: Ligar, acelerar, frear, parar
- **Objetos**: Meu carro (Fiat, Palio, branco, 2020, ABC-1234)

Por que usar Classes e Objetos?

1. **Organização**: Código mais estruturado e fácil de entender
2. **Reutilização**: Uma classe pode gerar múltiplos objetos
3. **Manutenção**: Mudanças na classe afetam todos os objetos
4. **Abstração**: Foco no que é importante, escondendo detalhes
5. **Encapsulamento**: Proteção dos dados internos

Vantagens:

- Código mais limpo e legível
- Facilita trabalho em equipe
- Reduz duplicação de código
- Permite modelagem do mundo real

Estrutura de uma Classe

- **Atributos (Campos)**: Características ou propriedades da classe
- **Métodos**: Ações ou comportamentos que a classe pode realizar
- **Construtor**: Método especial para criar objetos
- **Modificadores de Acesso**: Controlam quem pode acessar os membros

Exemplo Básico:

- **Classe**: Pessoa
- **Atributos**: Nome, idade, email
- **Métodos**: Falar, andar, comer
- **Construtor**: Criar uma nova pessoa

Modificadores de Acesso

1. **public**: Acessível de qualquer lugar
2. **private**: Acessível apenas dentro da própria classe
3. **protected**: Acessível na classe e nas classes filhas
4. **internal**: Acessível apenas dentro do mesmo assembly

Boas Práticas:

- Atributos geralmente são **private**
- Métodos públicos para interação externa
- Use **properties** para acesso controlado aos atributos

Exemplo 1: Classe Conta Bancária - Estrutura

Atributos:

- Número da conta
- Nome do titular
- Saldo
- Tipo da conta

Métodos:

- Depositar
- Sacar
- Verificar saldo
- Transferir

Exemplo 1: Classe Conta Bancária - Objetos

Objetos possíveis:

- Conta de João (12345, João Silva, R$ 1.500,00, Corrente)
- Conta de Maria (67890, Maria Santos, R$ 3.200,00, Poupança)

Características dos Objetos:

- Cada objeto tem seus próprios valores
- Podem executar os mesmos métodos
- Estado independente entre objetos

Exemplo 2: Classe Animal - Estrutura

Atributos:

- Nome
- Espécie
- Idade
- Peso
- Cor

Métodos:

- Comer
- Dormir
- Mover
- Fazer som

Exemplo 2: Classe Animal - Objetos

Objetos possíveis:

- Meu gato (Mimi, Gato, 3 anos, 4kg, Branco)
- Cachorro do vizinho (Rex, Cachorro, 5 anos, 15kg, Marrom)

Comportamentos dos Objetos:

- Cada animal tem características únicas
- Comportamentos podem variar por espécie
- Estado muda com ações (comer, dormir)

Exemplo 3: Classe Produto - Estrutura

Atributos:

- Código
- Nome
- Preço
- Quantidade em estoque
- Categoria

Métodos:

- Calcular desconto
- Atualizar estoque
- Verificar disponibilidade
- Aplicar promoção

Exemplo 3: Classe Produto - Objetos

Objetos possíveis:

- Notebook (NB001, Dell Inspiron, R$ 2.500,00, 10, Eletrônicos)
- Livro (LB002, C# Completo, R$ 89,90, 25, Livros)

Funcionalidades dos Objetos:

- Controle de estoque
- Cálculo de preços
- Gestão de categorias
- Aplicação de descontos

Estrutura Básica de uma Classe em C#

Elementos principais:

- **Atributos**: `private string nome; private int idade;`
- **Construtor**: `public NomeDaClasse(string nome, int idade)`
- **Métodos**: `public void Falar()`
- **Properties**: `public string Nome { get; set; }`

Exemplo completo: Ver arquivo `Exemplos/ContaBancaria.cs`

Criando e Usando Objetos

Passos para usar objetos:

1. **Criar objetos**: `NomeDaClasse obj = new NomeDaClasse();`
2. **Usar métodos**: `obj.Falar(); obj.Comer();`
3. **Acessar properties**: `obj.Nome = "João";`
4. **Cada objeto é independente**: Valores únicos para cada instância

Exemplo prático: Ver arquivo `Exemplos/Program.cs`

Exemplo Completo: Classe Conta Bancária

Arquivo: Exemplos/ContaBancaria.cs

- Implementação completa da classe ContaBancaria
- Atributos: numero, titular, saldo, tipo
- Métodos: Depositar, Sacar, VerificarSaldo, Transferir
- Construtor para inicializar a conta
- Properties para acesso controlado

Características:

- Validação de valores negativos
- Controle de saldo insuficiente
- Mensagens informativas
- Código bem documentado

Exemplo Completo: Classe Animal

Arquivo: Exemplos/Animal.cs

- Implementação da classe Animal
- Atributos: nome, especie, idade, peso, cor
- Métodos: Comer, Dormir, Mover, FazerSom
- Construtor com parâmetros opcionais
- Método ToString() para exibição

Características:

- Validação de dados de entrada
- Comportamentos específicos por espécie
- Cálculo de idade em meses
- Interface amigável

Exemplo Completo: Classe Produto

Arquivo: Exemplos/Produto.cs

- Implementação da classe Produto
- Atributos: codigo, nome, preco, estoque, categoria
- Métodos: CalcularDesconto, AtualizarEstoque, VerificarDisponibilidade
- Construtor com validações
- Properties com validação

Características:

- Cálculo automático de descontos
- Controle de estoque
- Validação de preços
- Categorização de produtos

Exercício 1: Classe Livro

Crie uma classe Livro com:

Atributos:

- Título
- Autor
- ISBN
- Páginas
- Preço
- Ano de publicação

Métodos:

- Calcular preço com desconto
- Verificar se é livro novo (últimos 2 anos)
- Exibir informações do livro

Exercício 2: Classe Retângulo

Crie uma classe Retângulo com:

Atributos:

- Largura
- Altura

Métodos:

- Calcular área
- Calcular perímetro
- Verificar se é quadrado
- Desenhar retângulo (usando asteriscos)

Tempo: 10 minutos

Exercício 3: Classe Estudante - Estrutura

Crie uma classe Estudante com:

Atributos:

- Nome
- Matrícula
- Curso
- Notas (array de 4 notas)

Métodos:

- Calcular média
- Verificar se passou (média ¿= 7.0)
- Adicionar nota
- Exibir boletim

Exercício 3: Classe Estudante - Implementação

Tempo: 20 minutos

Dicas para implementação:

- Use arrays para armazenar notas
- Implemente validação de notas (0-10)
- Calcule média aritmética simples
- Formate a saída do boletim

Teste com diferentes estudantes!

Nomenclatura

- **Classes**: PascalCase (ex: ContaBancaria, Produto)
- **Métodos**: PascalCase (ex: Depositar, CalcularSaldo)
- **Atributos**: camelCase (ex: nome, saldoAtual)
- **Constantes**: UPPER_CASE (ex: TAXA_JUROS)

Exemplos:

- **Bom**: `public class ContaBancaria`
- **Ruim**: `public class contabancaria`
- **Bom**: `private string nomeTitular`
- **Ruim**: `private string NomeTitular`

Encapsulamento

- **Sempre** use modificadores de acesso apropriados
- Atributos devem ser **private**
- Use **properties** para acesso controlado
- Evite atributos **public**

Exemplo:

- **Bom**: `private double saldo;` + `public double Saldo { get; set; }`
- **Ruim**: `public double saldo;`

Vantagens:

- Controle de acesso aos dados
- Validação automática

Documentação e Comentários - Boas Práticas

- Use **XML comments** para documentar classes e métodos
- Comente lógica complexa
- Use nomes descritivos (evite comentários óbvios)
- Documente parâmetros e valores de retorno

Benefícios:

- Facilita manutenção do código
- Melhora compreensão para outros desenvolvedores
- Gera documentação automática

Documentação e Comentários - Exemplo

Exemplo de XML Comments:

- `/// <summary>`
- `/// Deposita um valor na conta`
- `/// </summary>`
- `/// <param name="valor≫Valor a ser depositado</param>`
- `/// <returns>True se o depósito foi realizado</returns>`

Resultado:

- IntelliSense mostra a documentação
- Facilita uso da classe por outros desenvolvedores

Tópicos Relacionados

1. **Herança**
   - Classes pai e filhas
   - Reutilização de código
   - Polimorfismo
2. **Interfaces**
   - Contratos de comportamento
   - Múltiplas implementações
   - Abstração avançada
3. **Polimorfismo**
   - Múltiplas formas
   - Sobrescrita de métodos
   - Ligação tardia

Leitura Recomendada

- **Microsoft Docs**: Classes and Objects in C#
- **Clean Code**: Robert C. Martin
- **Effective C#**: Bill Wagner
- **Pro C# 10**: Andrew Troelsen

Prática Recomendada:

1. Implemente os exemplos da pasta `Exemplos`
2. Resolva os exercícios propostos
3. Crie suas próprias classes
4. Pratique com projetos reais

Dúvidas e Discussão

Obrigado pela atenção!

Pontos-Chave da Aula:

1. **Classes** são moldes que definem características e comportamentos
2. **Objetos** são instâncias específicas de classes
3. **Encapsulamento** protege os dados e controla o acesso
4. **Boas práticas** facilitam manutenção e legibilidade