Fundamentos de Desenvolvimento com C#

[25E1_2]

TP03

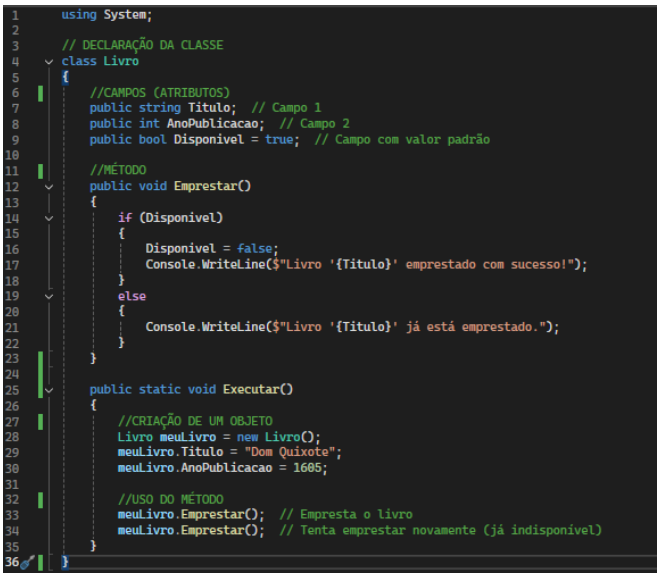
Exercícios C#

Larissa Xavier Araújo

Link para código no github: https://github.com/larisssaxavier/AtividadeTresDR2

Exercício 1: Conceitos de Classe, Objeto, Campos e Métodos (C#)

- **Classe:** É um modelo ou molde que define a estrutura e o comportamento de um objeto. Representa uma categoria de coisas do mundo real.
- **Objeto:** É uma instância concreta de uma classe, ou seja, um item específico criado a partir do molde (classe).
- **Campos (Atributos):** São variáveis que armazenam o estado (dados) de um objeto.
- **Métodos:** São funções que definem as ações/comportamentos do objeto. Podem manipular campos ou executar tarefas.



```
using System;

// DECLARAÇÃO DA CLASSE
class Livro
{
    //CAMPOS (ATRIBUTOS)
    public string Titulo;  // Campo 1
    public int AnoPublicacao;  // Campo 2
    public bool Disponivel = true;  // Campo com valor padrão

    //MÉTODO
    public void Emprestar()
    {
        if (Disponivel)
        {
            Disponivel = false;
            Console.WriteLine($"Livro '{Titulo}' emprestado com sucesso!");
        }
        else
        {
            Console.WriteLine($"Livro '{Titulo}' já está emprestado.");
        }
    }

    public static void Executar()
    {
        //CRIAÇÃO DE UM OBJETO
        Livro meuLivro = new Livro();
        meuLivro.Titulo = "Dom Quixote";
        meuLivro.AnoPublicacao = 1605;

        //USO DO MÉTODO
        meuLivro.Emprestar();  // Empresta o livro
        meuLivro.Emprestar();  // Tenta emprestar novamente (já indisponível)
    }
}
```

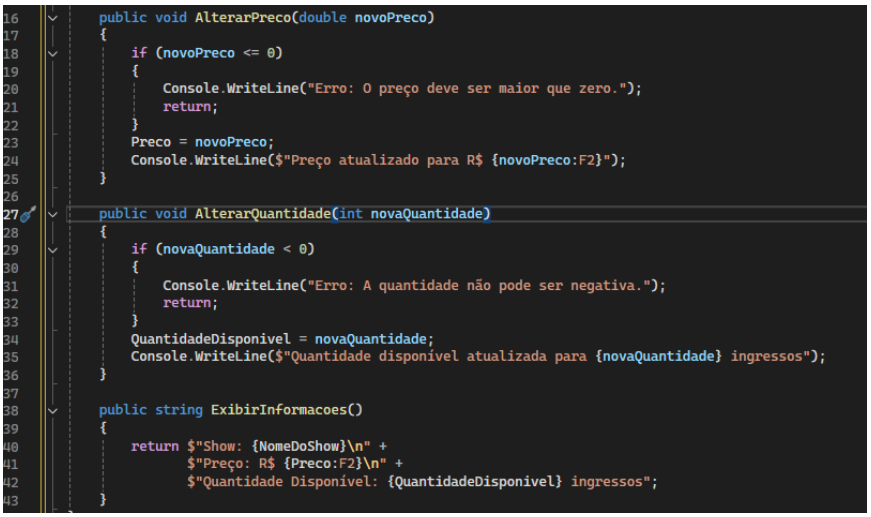
Exercício 2: Criando a Classe "Ingresso"

```
using System;

class Ingressos
{
    public string NomeDoShow { get; set; }       // Nome do evento musical
    public double Preco { get; set; }            // Preço unitário do ingresso
    public int QuantidadeDisponivel { get; set; } // Quantidade em estoque
}
```

- Nome do show identifica a qual show o ingresso pertence e evita erros de confusão entre eventos diferentes.
- Preço armazena o valor do ingresso e é utilizado em cálculos financeiros.
- Quantidade disponível controla o estoque dos ingressos e evita que sejam vendidos quantidades maiores que o disponível.

Exercício 3: Métodos Básicos da Classe "Ingresso"



```
public void AlterarPreco(double novoPreco)
{
    if (novoPreco <= 0)
    {
        Console.WriteLine("Erro: O preço deve ser maior que zero.");
        return;
    }
    Preco = novoPreco;
    Console.WriteLine($"Preço atualizado para R$ {novoPreco:F2}");
}

public void AlterarQuantidade(int novaQuantidade)
{
    if (novaQuantidade < 0)
    {
        Console.WriteLine("Erro: A quantidade não pode ser negativa.");
        return;
    }
    QuantidadeDisponivel = novaQuantidade;
    Console.WriteLine($"Quantidade disponível atualizada para {novaQuantidade} ingressos");
}

public string ExibirInformacoes()
{
    return $"Show: {NomeDoShow}\n" +
           $"Preço: R$ {Preco:F2}\n" +
           $"Quantidade Disponível: {QuantidadeDisponivel} ingressos";
}
```

Exercício 4: Testando a Classe "Ingresso"

```
public static void Teste()
{
    Console.WriteLine("\n-- TESTE DA CLASSE INGRESSO --");

    //Instanciação do ingresso com valores iniciais
    Ingressos showRock = new Ingressos(
        nomeDoShow: "Asking Alenxandria 2025",
        preco: 680.00,
        quantidadeDisponivel: 10000
    );

    //Exibição das informações iniciais
    Console.WriteLine("\n--- Informações Iniciais ---");
    Console.WriteLine(showRock.ExibirInformacoes());

    //Teste de atualização de preço
    Console.WriteLine("\n--- Atualizando Preço ---");
    showRock.AlterarPreco(800.00);
    showRock.AlterarPreco(-230.00);

    //Teste de atualização de quantidade
    Console.WriteLine("\n--- Atualizando Quantidade ---");
    showRock.AlterarQuantidade(7620);
    showRock.AlterarQuantidade(-4560);

    //Exibição das informações finais
    Console.WriteLine("\n--- Informações Atualizadas ---");
    Console.WriteLine(showRock.ExibirInformacoes());
}
```

```
-- TESTE DA CLASSE INGRESSO --

--- Informações Iniciais ---
Show: Asking Alenxandria 2025
Preço: R$ 680,00
Quantidade Disponível: 10000 ingressos

--- Atualizando Preço ---
Preço atualizado para R$ 800,00
Erro: O preço deve ser maior que zero.

--- Atualizando Quantidade ---
Quantidade disponível atualizada para 7620 ingressos
Erro: A quantidade não pode ser negativa.

--- Informações Atualizadas ---
Show: Asking Alenxandria 2025
Preço: R$ 800,00
Quantidade Disponível: 7620 ingressos
```

Exercício 5: Criando Métodos de Propriedade (Getters e Setters)

```
public Ingressos(string nomeDoShow, double preco, int quantidadeDisponivel)
{
    _nomeDoShow = nomeDoShow;
    _preco = preco;
    _quantidadeDisponivel = quantidadeDisponivel;
}

public string GetNomeDoShow() => _nomeDoShow;
public double GetPreco() => _preco;
public int GetQuantidadeDisponivel() => _quantidadeDisponivel;
```

```
public void SetNomeDoShow(string novoNome)
{
    _nomeDoShow = novoNome;
}

public void SetPreco(double novoPreco)
{
    _preco = novoPreco;
}

public void SetQuantidadeDisponivel(int novaQtd)
{
    _quantidadeDisponivel = novaQtd;
}
```

```
//TESTE DE GETTERS E SETTERS
Console.WriteLine("\n--- SEGUNDO TESTE ---");

Ingressos festival = new Ingressos("Psica", 345, 8500);
Console.WriteLine(festival.ExibirInformacoes());

//Teste dos métodos Set
Console.WriteLine("\n--- Atualizando Valores ---");
festival.SetNomeDoShow("Psica 2025");
festival.SetPreco(450.00);
festival.SetQuantidadeDisponivel(4500);

//Teste dos métodos Get
Console.WriteLine("\n--- Valores Atualizados via Getters ---");
Console.WriteLine($"Nome: {festival.GetNomeDoShow()}");
Console.WriteLine($"Preço: R$ {festival.GetPreco():F2}");
Console.WriteLine($"Quantidade: {festival.GetQuantidadeDisponivel()}");

Console.WriteLine("\n--- Informações Consolidadas ---");
Console.WriteLine(festival.ExibirInformacoes());
}
```

Exercício 6: Adicionando Construtores à Classe "Ingresso"

```
public Ingressos(string nomeDoShow, double preco, int quantidadeDisponivel)
{
    if (string.IsNullOrWhiteSpace(nomeDoShow))
        throw new ArgumentException("Nome do show é obrigatório");

    if (preco <= 0)
        throw new ArgumentException("Preço deve ser positivo");

    if (quantidadeDisponivel < 0)
        throw new ArgumentException("Quantidade não pode ser negativa");

    _nomeDoShow = nomeDoShow;
    _preco = preco;
    _quantidadeDisponivel = quantidadeDisponivel;
}
```

```
123         //TESTE DE CONSTRUTORR
124         Console.WriteLine("=== CRIAÇÃO DE INGRESSO COM CONSTRUTOR ===");
125
126         //Criação do objeto usando o construtor
127         Ingresso festival = new Ingresso(
128             nomeDoShow: "Rock in Rio 2024",
129             preco: 399.90,
130             quantidadeDisponivel: 15000
131         );
132
133         //Exibição das informações
134         Console.WriteLine("\n--- Ingresso Criado ---");
135         Console.WriteLine(festival.ExibirInformacoes());
136     }
```

Construtores ajudam a fornecer os dados essenciais para a criação do objeto, todas as regras de validação ficam em um só lugar e garante que o objeto sempre será criado em um estado válido.

Exercício 7: Modelando uma Matrícula

```
// Construtor
public void Matricula(string nomeDoAluno, string curso, int numeroMatricula, string situacao, string dataIni
{
    NomeDoAluno = nomeDoAluno;
    Curso = curso;
    NumeroMatricula = numeroMatricula;
    Situacao = situacao;
    DataInicial = dataInicial;
}

// Método para exibir as informações da matrícula
public void ExibirDados()
{
    Console.WriteLine("----- Dados da Matrícula -----");
    Console.WriteLine($"Nome do Aluno: {NomeDoAluno}");
    Console.WriteLine($"Curso: {Curso}");
    Console.WriteLine($"Número da Matrícula: {NumeroMatricula}");
    Console.WriteLine($"Situação: {Situacao}");
    Console.WriteLine($"Data Inicial: {DataInicial}");
}
public void Executar()
{
    Matricula matricula1 = new Matricula(
        "Larissa Xavier",
        "Engenharia da Computação",
        12345,
        "Ativa",
        "01/08/2024"
    );

    matricula1.ExibirDados();
}
```

Exercício 8: Criando Métodos na Classe de Matrícula

```
21      // Método para trancar a matrícula
22      public void Trancar()
23      {
24          Situacao = "Trancada";
25          Console.WriteLine("Matrícula trancada com sucesso.");
26      }
27
28      // Método para reativar a matrícula
29      public void Reativar()
30      {
31          Situacao = "Ativa";
32          Console.WriteLine("Matrícula reativada com sucesso.");
33      }
34
35      // Método para exibir as informações da matrícula
36      public void ExibirDados()
37      {
38          Console.WriteLine("----- Dados da Matrícula -----");
39          Console.WriteLine($"Nome do Aluno: {NomeDoAluno}");
40          Console.WriteLine($"Curso: {Curso}");
41          Console.WriteLine($"Número da Matrícula: {NumeroMatricula}");
42          Console.WriteLine($"Situação: {Situacao}");
43          Console.WriteLine($"Data Inicial: {DataInicial}");
```

Exercício 9: Testando a Classe de Matrícula

```
public static void Executar()
{
    Matricula matricula1 = new Matricula("Larissa Xavier", "Engenharia da computação", 123456, "Ativa", "01/

    matricula1.ExibirDados();

    Console.WriteLine("\n->TESTE");

    // Exibindo situação inicial
    Console.WriteLine("Situação inicial da matrícula:");
    matricula1.ExibirDados();

    // Trancando a matrícula
    matricula1.Trancar();
    Console.WriteLine("Após trancar a matrícula:");
    matricula1.ExibirDados();

    // Reativando a matrícula
    matricula1.Reativar();
    Console.WriteLine("Após reativar a matrícula:");
    matricula1.ExibirDados();
}
```

```
->TESTE
Situação inicial da matrícula:
----- Dados da Matrícula -----
Nome do Aluno: Larissa Xavier
Curso: Engenharia da computação
Número da Matrícula: 123456
Situação: Ativa
Data Inicial: 01/08/2024
Matrícula trancada com sucesso.
Após trancar a matrícula:
----- Dados da Matrícula -----
Nome do Aluno: Larissa Xavier
Curso: Engenharia da computação
Número da Matrícula: 123456
Situação: Trancada
Data Inicial: 01/08/2024
Matrícula reativada com sucesso.
Após reativar a matrícula:
----- Dados da Matrícula -----
Nome do Aluno: Larissa Xavier
Curso: Engenharia da computação
Número da Matrícula: 123456
Situação: Ativa
Data Inicial: 01/08/2024
```

Exercício 10: Definindo Classes de Formas Geométricas

```
using System;

class FormasGeometricas
{
    public class Circulo
    {
        public double Raio { get; set; }

        public Circulo(double raio)
        {
            if (raio <= 0)
                throw new ArgumentException("Raio deve ter valor positivo");
            Raio = raio;
        }
    }

    public class Esfera
    {
        public double Raio { get; set; }

        public Esfera(double raio)
        {
            if (raio <= 0)
                throw new ArgumentException("Raio deve ter valor positivo");
            Raio = raio;
        }
    }
}
```

Raio é a medida necessária para representar a geometria tanto do círculo quanto da esfera e permite realizar os cálculos de área, perímetro, superfície e volume.

Exercício 11: Criando Métodos de Cálculo

```
    public class Circulo
    {
        public double Raio { get; set; }

        public Circulo(double raio)
        {
            if (raio <= 0)
                throw new ArgumentException("Raio deve ter valor positivo");
            Raio = raio;
        }

        public double CalcularArea()
        {
            return Math.PI * Math.Pow(Raio, 2);
        }
    }

    public class Esfera
    {
        public double Raio { get; set; }

        public Esfera(double raio)
        {
            if (raio <= 0)
                throw new ArgumentException("Raio deve ter valor positivo");
            Raio = raio;
        }

        public double CalculaVolume()
        {
            return (4.0 / 3.0) * Math.PI * Math.Pow(Raio, 3);
        }
    }
```

Exercício 12: Testando as Classes de Figuras

```
39      public static void Executar()
40      {
41          Circulo circulo = new Circulo(2.5);
42          double areaCirculo = circulo.CalcularArea();
43          Console.WriteLine($"\nRaio do círculo: {circulo.Raio}");
44          Console.WriteLine($"Área do círculo: {areaCirculo:F2}");
45
46          Esfera esfera = new Esfera(6);
47          double volumeEsfera = esfera.CalculaVolume();
48          Console.WriteLine($"\nRaio da esfera: {esfera.Raio}");
49          Console.WriteLine($"Volume da esfera: {volumeEsfera:F2}");
50
51      }
```

```
Área do círculo: 19,63
Volume da esfera: 904,78
```