

Fundamentos de Desenvolvimento com C# [25E1_2]

Página inicial do site > Meus cursos > Fundamentos de Desenvolvimento com C# [25E1_2] > Etapa 6 > Teste de Performance - TP3 [Obrigatório]

TESTE DE PERFORMANCE - TP3 [OBRIGATÓRIO]

Olá Samuel,

Chegamos em uma das etapas de preparação! A cada Teste de Performance (TP) você terá a oportunidade de praticar os conhecimentos adquiridos e receber feedbacks relevantes para o seu aprendizado.

Uso de IA: Sinal Vermelho

Todas as partes deste trabalho devem ser da autoria do aluno. Qualquer uso de ferramentas geradoras de IA, como ChatGPT, é proibido. O uso de IA gerativa será considerado má conduta acadêmica e estará sujeito à aplicação do código disciplinar, pois as tarefas deste trabalho foram elaboradas para desafiar o aluno a desenvolver conhecimentos de base, pensamento crítico e habilidades de resolução de problemas. O uso da tecnologia de IA limitaria sua capacidade de desenvolver essas competências e de atingir os objetivos de aprendizagem desta disciplina.

Competência: Escrever programas em C# que utilizem classes e objetos

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

Enunciado: Explique, de forma clara e objetiva, os conceitos de classe, objeto, campos (ou atributos) e métodos no contexto da Programação Orientada a Objetos em C#. Em seguida, crie um exemplo simples (pode ser em pseudocódigo ou C#) que ilustre esses conceitos na prática. Seu exemplo deve conter:

- Declaração de uma classe.
- Pelo menos dois campos (ou atributos) diferentes.
- Pelo menos um método que utilize ou manipule esses campos.
- Criação de um objeto a partir da classe declarada.

Observações:

- Use exemplos do dia a dia (por exemplo, "Carro", "Pessoas", "Livro") para tornar a explicação mais acessível.
- Relacione cada conceito do enunciado (classe, objeto, atributos, métodos) com trechos do seu exemplo prático.
- Seja objetivo, mas inclua detalhes suficientes para demonstrar compreensão do tema.

Considerações:

- Clareza e correção conceitual na explicação dos termos (classe, objeto, campos, métodos).
- Qualidade do exemplo prático e conexão com a teoria.
- Organização do texto e, se optar por C#, uso adequado da linguagem (nomes de classe iniciando em maiúscula, etc.).

Contexto para as próximas questões (Exercícios 2 a 6): Venda de Show

Imagine que você é responsável pela venda de ingressos para um grande evento musical. É preciso cadastrar ingressos com nome do show, preço e quantidade disponível, bem como atualizar essas informações sempre que algo mudar. Ao final, você deve exibir as informações do ingresso para ter um registro atualizado do sistema.

Exercício 2 Criando a Classe "Ingresso"

Enunciado: Com base na história de usuário acima, crie uma classe que represente um ingresso para o show. Sua tarefa é:

1. Declarar a classe e atributos
 - Crie uma classe chamada Ingresso.
 - Insira nela, pelo menos, os atributos: nomeDoShow (string), preco (double) e quantidadeDisponivel (int).
 - Explique brevemente por que cada atributo é importante no contexto de venda de shows.

Considerações:

- Declaração correta dos atributos e criação da classe Ingresso.
- Clareza na explicação da importância de cada atributo.
- Organização adequada do código (sem se preocupar ainda com métodos mais avançados).

Exercício 3 Métodos Básicos da Classe "Ingresso"

Enunciado: Dando continuidade ao contexto da venda de ingressos para um show, agora vamos aprimorar a classe Ingresso com métodos básicos:

1. Atualizar Preço
 - Crie um método chamado AlterarPreco(double novoPreco) que recebe um novo preço e atualiza o atributo preco.
2. Atualizar Quantidade
 - Crie um método chamado AlterarQuantidade(int novaQuantidade) que recebe uma nova quantidade e atualiza o atributo quantidadeDisponivel.
3. Exibir Informações
 - Crie um método chamado ExibirInformacoes(), que mostre (no console ou como retorno de string) o nomeDoShow, o preco atual e a quantidadeDisponivel do ingresso.

Considerações:

- Correta implementação dos métodos solicitados.
- Clareza e objetividade no código, demonstrando como cada método cumpre seu papel.
- Uso adequado dos atributos já criados na classe Ingresso.

Exercício 4 Testando a Classe "Ingresso"

Enunciado: Agora é hora de testar a classe Ingresso. Para isso:

1. Criar Classe de Teste
 - Crie uma nova classe (por exemplo, AppIngresso ou Program) com um método Main.
2. Instanciar um Ingresso
 - Dentro de Main, crie um objeto da classe Ingresso.
 - Atribua valores iniciais aos atributos (nomeDoShow, preco, quantidadeDisponivel).
3. Chamar os Métodos de Atualização
 - Chame AlterarPreco e AlterarQuantidade para modificar o preço e a quantidade disponíveis.
 - Exibir Informações.
 - Por fim, chame ExibirInformacoes() para confirmar se as alterações foram aplicadas corretamente.

Considerações:

- Criação correta de uma classe ou método Main para executar o código.
- Instanciação da classe Ingresso com valores iniciais nos atributos.
- Uso correto dos métodos de atualização e exibição de informações.
- Organização e legibilidade do código.

Exercício 5 Criando Métodos de Propriedade (Getters e Setters)

Enunciado: Ainda no contexto de venda de show, vamos criar métodos para ler e alterar os valores dos atributos sem acessar diretamente as variáveis. Esses métodos são os getters e setters.

1. Métodos "Get" (Leitura)
 - Crie, na classe Ingresso, um método para retornar o valor de cada atributo: -GetNomeDoShow(), GetPreco(), GetQuantidadeDisponivel(), etc.
2. Métodos "Set" (Atualização)
 - Crie métodos que recebam um novo valor para cada atributo: -SetNomeDoShow(string novoNome), SetPreco(double novoPreco), SetQuantidadeDisponivel(int novaQtd), etc.
3. Exemplo de Uso
 - Mostre (no Main ou em outra classe de teste) como invocar esses métodos, por exemplo, chamando SetPreco(200.0) e em seguida GetPreco() para confirmar a atualização.

Observações:

- Você não precisa, neste momento, usar modificadores de acesso (public/private). Basta ter métodos simples de leitura e escrita.
- Explique por que esses métodos (getters e setters) podem ser úteis, mesmo que não esteja usando encapsulamento total ainda.

Considerações:

- Criação dos métodos de leitura (getters) e de atualização (setters) para cada atributo.
- Demonstração clara de uso (chamada no Main ou classe de teste).
- Clareza e objetividade na explicação sobre função dos getters e setters no gerenciamento de ingressos.

Exercício 6 Adicionando Construtores à Classe "Ingresso"

Enunciado: Para facilitar a criação de novos ingressos no sistema, vamos definir construtores:

1. Criar Construtor
 - Na classe Ingresso, crie um construtor que receba parâmetros para inicializar obrigatoriamente nomeDoShow, preco e quantidadeDisponivel.
2. Exemplificar o Uso
 - No método Main ou em outra classe de teste, instancie um objeto do tipo Ingresso usando esse construtor.
 - Exiba as informações do objeto (usando ExibirInformacoes() ou getters) para confirmar que os valores foram atribuídos corretamente.
3. Justificativa
 - Explique por que usar um construtor facilita a criação de objetos em relação a chamar vários métodos Set... separadamente.

Considerações:

- Declaração correta de um construtor com parâmetros na classe Ingresso.
- Atribuição adequada dos atributos nomeDoShow, preco e quantidadeDisponivel.
- Demonstração clara do uso do construtor (instanciando um objeto e exibindo informações).
- Clareza na explicação da utilidade dos construtores.

Contexto para Exercícios 7, 8 e 9: Matrícula de Faculdade

Agora, mude de cenário: você vai criar um sistema simples de matrícula para uma faculdade. A ideia é cadastrar informações básicas do aluno e efetuar operações relacionadas à sua matrícula.

Exercício 7 Modelando uma Matrícula

Enunciado: Crie uma classe chamada 'Matricula' que contenha, no mínimo, os seguintes atributos:

- NomeDoAluno (string)
- Curso (string)
- NumeroMatricula (int) – identificador da matrícula
- Situacao (string) – por exemplo, "Ativa", "Trancada", "Concluida" etc.
- DataInicial (string) – registra a data em que a matrícula foi iniciada

Considerações:

- Criação de uma classe Matricula com atributos nomeDoAluno, curso, numeroMatricula, situacao e dataInicial.
- Implementação correta da classe Matricula.
- Atribuição correta dos atributos nomeDoAluno, curso, numeroMatricula, situacao e dataInicial.

Exercício 8 Criando Métodos na Classe de Matrícula

Enunciado: Agora que a classe 'Matricula' foi definida, vamos criar métodos para manipular seu estado:

1. Trancar Matrícula
 - Um método Trancar() que altera Situacao para "Trancada".
2. Reativar Matrícula
 - Um método Reativar() que altera Situacao para "Ativa".
3. Exibir Informações
 - Um método ExibirInformacoes() que mostre (no console ou como retorno de string) o nome do aluno, o curso, a situação atual e a data inicial da matrícula.

Observações:

- Escolha nomes adequados aos métodos (iniciando com letra maiúscula ou minúscula, conforme convenção do seu projeto, mas seja consistente).

Considerações:

- Implementação adequada dos métodos solicitados (trancar e reativar).
- Exibição clara das informações básicas da matrícula.
- Organização e clareza do código.

Exercício 9 Testando a Classe de Matrícula

Enunciado: Crie uma classe de teste (por exemplo, TestaMatricula) com um método Main para:

1. Instanciar um objeto Matricula
 - Na classe Matricula, crie um método para retornar o valor de cada atributo: -GetNomeDoAluno(), GetCurso(), GetNumeroMatricula(), GetSituacao(), GetDataInicial().
2. Atribuir valores aos atributos básicos (nome do aluno, curso, número da matrícula, data inicial etc.).
3. Chamar os métodos Trancar() e Reativar(), além de ExibirInformacoes(), mostrando no console como a situação da matrícula muda.

Observações:

- Mantenha as convenções de nomes e boas práticas de organização de código.
- Você pode colocar as classes em arquivos separados ou no mesmo arquivo, contanto que fique claro onde está o Main.

Considerações:

- Correta instânciação da classe Matricula e atribuição de valores iniciais.
- Chamadas adequadas aos métodos Trancar(), Reativar() e ExibirInformacoes().
- Clareza na evidência de que os métodos funcionam (ex: imprimir antes e depois).

Contexto para Exercícios 10, 11 e 12: Áreas e Volumes em C#

Você está desenvolvendo um sistema para calcular áreas (em figuras 2D) e volumes (em figuras 3D), agora utilizando C#. Não serão utilizados conceitos avançados, como herança: cada classe será independente.

Exercício 10 Definindo Classes de Formas Geométricas

Enunciado: Siga o passo a passo abaixo:

- Crie a classe Circulo com o atributo Raio (double).
- Crie a classe Esfera com o atributo Raio (double).

Considerações:

- Declaração correta das classes Circulo e Esfera.
- Presença e clareza na definição(s) atributo(s) necessário(s).
- Código organizado e fácil de ler, sem introduzir conceitos não ensinados (como herança).

Exercício 11 Criando Métodos de Cálculo

Enunciado: Agora você adicionará métodos para calcular:

1. Área do Círculo
 - Na classe Circulo, crie um método CalcularArea() que retorne Math.PI * (Raio * Raio).
2. Volume da Esfera
 - Na classe Esfera, crie um método CalcularVolume() que retorne (4.0 / 3.0) * Math.PI * (Raio * Raio * Raio).

Observações:

- Não utilize recursos avançados de C#, como herança ou interfaces – fique nos métodos simples.
- Lembre-se de retornar o resultado, não somente imprimir.

Considerações:

- Métodos de cálculo corretos e funcionais.
- Retorno adequado dos valores (pode usar double).
- Código bem organizado.

Exercício 12 Testando as Classes de Figuras

Enunciado: Crie uma classe de teste (por exemplo, TestaFiguras) com Main para:

1. Instanciar um objeto Circulo e outro da classe Esfera.
2. Definir valores para o círculo (por exemplo, 3.0 para o círculo e 5.0 para a esfera).
3. Chamar os métodos CalcularArea() e CalcularVolume().
4. Exibir os resultados no console, validando se funcionam corretamente.

Observações:

- Caso opte por arquivos separados, lembre-se de compilar todos e depois rodar a classe com Main.
- Não use conceitos como herança ou polimorfismo; limite-se a instanciar e chamar métodos.

Considerações:

- Criação de uma classe de teste com método Main.
- Instanciação correta de Círculo e Esfera, atribuindo Raio adequadamente.
- Chamadas corretas aos métodos de cálculo, exibindo resultados plausíveis.
- Organização e legibilidade do código, seguindo convenções de C#.

Assim que terminar, salve seu trabalho em PDF nomeando o arquivo conforme a regra "nome_sobrenome_DR1_TP3.pdf" e poste como resposta a este TP.