**UNIVERSIDADE POSITIVO**

**TÓPICOS ESPECIALS EM DESENVOLVIMENTO DE SOFTWARE**

**Prof Escobar**

**Avaliação prática de Orientação a Objetos em C#**

* Responda às questões teóricas e desenvolva o código para as questões neste documento;
* Quando a questão solicitar implementação de código, deve-se implementar, também, classes consumidoras e que permitam os testes das classes desenvolvidas;
* Os prompts de GPT, Copilot ou qualquer outra IA generativa devem ser entregues junto com a sua avaliação;
* Pontos possíveis: 2,5;
* Realização individual;
* Data de entrega: 02/05/2024, 23h59 impreterivelmente.

Realize o que se pede.

1. Explique o que é herança em orientação a objetos e como ela é implementada em C#. Dê um exemplo prático de como a herança pode ser utilizada em um sistema de gerenciamento de funcionários.

Herança é um conjunto de características que uma classe ganha ao ser criada e adicionada a baixo de uma superclasse, ela serve pra otimizar o código e especializar as classes. Um exemplo pode ser pessoa e aluno

public class Pessoa{

    public int idade { get; set; }

    public string nome { get; set; }

    public string pai { get; set; }

    public string mae { get; set; }

}

public class Estudante

{    public string matricula { get; set; }

}

1. O que são interfaces em orientação a objetos e qual é a sua importância na construção de sistemas em C#? Dê um exemplo de como uma interface pode ser utilizada para garantir a interoperabilidade entre diferentes classes em um sistema de pagamento online.

São “contratos” onde diz a assinatura de um método.

Pode ser útil pra ter varias classes com a mesma assinatura.

Um exemplo é autorização de passagem de produtos com autenticador.

{chat prompt (não estou conseguindo pagar pedagio esqueci como usa interface)}

public interface IProduto{

public bool autenticador(string fornecedor);

}  
  
public class Revenda

{

public string fornecedor;

public bool setProduto(IProduto produto)

{

if(produto.autenticador(fornecedor) == true){

return true;

}else{

return false;

}

}

}

public class Produto : IProduto{

public string fornecedor {get;set;}

public bool autenticador(string fornecedor){

return this.fornecedor==fornecedor;

}

}

using MySql.Data.MySqlClient;

using System;

using System.Data;

class Program

{

static void Main()

{

Revenda revenda = new Revenda();

revenda.fornecedor = "ambev";

Produto coca = new Produto();

coca.fornecedor ="ambev";

Console.WriteLine(revenda.setProduto(coca));

}

}

1. Diferencie sobrecarga e sobreposição (ou sobrescrita) e forneça exemplos de cada um em C#.

{chat prompt(oque achou de minha resposta?)}

Sobreposição é quando reescrevo uma função que minha classe herdou.

Sobrecarga é quando escrevo a função passando parâmetros distintos, evitando assim o uso de if else

Um exemplo simples é:

public int gerador(int num){

return num +10;

}

public string gerador(string num){

return num + " dez";

}

1. Crie uma classe Produto que represente um produto em uma loja online. A classe deve ter os seguintes atributos: Nome, Preco, QuantidadeEmEstoque. Implemente métodos para adicionar e remover unidades do estoque, e um método para calcular o valor total do produto em estoque. Os métodos devem alterar o estado do objeto instanciado

// Implemente métodos para adicionar e remover unidades do estoque, e um método para calcular o valor total do produto em estoque. Os métodos devem alterar o estado do objeto instanciado

namespace Model

{

public class Produto : IProduto

{

public string fornecedor { get; set; }

public string nome { get; set; }

public int quantidadeEmEstoque { get; set; }

public double preco { get; set; }

public void adicionar(int qtd)

{

if (qtd > 0)

{

this.quantidadeEmEstoque += qtd;

Console.WriteLine( "Adicionado");

}

else

{

Console.WriteLine( "Não é possível adicionar quantidade negativa");

}

}

public void remover(int qtd)

{

if (qtd < 0)

{

qtd \*=-1;

}

this.quantidadeEmEstoque -= qtd;

Console.WriteLine("Removido {0} itens", qtd);

}

public double calcularValorTotal()

{

return this.quantidadeEmEstoque \* this.preco;

}

public bool autenticador(string fornecedor)

{

return this.fornecedor == fornecedor;

}

}

}

1. Implemente um sistema para uma locadora de filmes. Crie uma classe Filme com os seguintes atributos: Titulo, Genero, Duracao, Disponivel (indicando se o filme está disponível para locação). Implemente métodos para registrar a locação de um filme, registrar a devolução de um filme e verificar se um filme está disponível para locação. Não há necessidade de implementar rotinas de bancos de dados.

{gpt prompt(public void addFilme(Filme filme){

this.filmes.Add(filme);

} tacerto?)}

1. Implemente uma hierarquia de classes para representar diferentes tipos de veículos, como carros, motos e bicicletas. Cada classe deve herdar da classe base Veiculo. Adicione atributos específicos para cada tipo de veículo, como número de portas para carros, cilindrada para motos e número de marchas para bicicletas.
2. Desenvolva um sistema de gerenciamento de contas bancárias com diferentes tipos de contas, como conta corrente, conta poupança e conta empresarial. Utilize herança para representar a relação entre as classes de contas e implemente métodos específicos para cada tipo de conta, como calcular juros para contas poupança e verificar saldo mínimo para contas empresariais.  
   {chatgpt prompt(pra eu fazer uma injeçã de dependencia num construtor de uma classe que herda é necessario então escrever o construtor novamente?)}
3. Crie uma aplicação para uma escola de idiomas que oferece diferentes tipos de cursos, como inglês, espanhol e francês. Utilize herança para representar a relação entre as classes de cursos e implemente métodos específicos para cada tipo de curso, como calcular a média de notas para cursos avançados e gerar certificados para cursos concluídos.
4. Desenvolva um sistema de processamento de pagamentos com diferentes métodos de pagamento, como cartão de crédito, boleto bancário e transferência bancária. Crie uma interface IMetodoPagamento com métodos para realizar o pagamento e verificar o status do pagamento. Implemente classes para cada método de pagamento que herdem dessa interface e forneça a lógica específica para cada método de pagamento.
5. Projete um sistema de reservas de voos para uma companhia aérea. Crie uma interface IReserva com métodos para reservar um voo, cancelar uma reserva e verificar o status da reserva. Implemente classes para diferentes tipos de reservas, como reserva de voo regular, reserva de voo com upgrade de classe e reserva de voo para grupos grandes. Utilize a interface para garantir a interoperabilidade entre os diferentes tipos de reservas.