# 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.

**Aluno:** Guilherme Rodrigues

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.**

   Em orientação a objetos a Herança permite que uma ou mais classes compartilhem seus atributos, métodos e outras seções da classe entre si. A ligação entre essas classes, estabelece um relacionamento hierárquico. No geral adota-se uma classe "mestra" denominada classe base e as classes que herdam os atributas dela são chamadas de subclasses ou classes derivadas. Em C# a herança é implementada com a utilização do termo ": " (dois pontos) entre a subclasse e a classe base.

   public class Subclasse **:** ClasseBase { }

   **Exemplo:** Possuímos a classe base "Pessoa" com os atributos nome e CPF. A partir disso podemos definir subclasses como "Funcionário" que possuirá os atributos cargo e salário, a subclasse "Terceiro" que possuirá os atributos id e nota_fiscal. Lembrando que ambas as subclasses herdam os atributos da classe base nome e CPF.

```
public class Pessoa {
    private string nome {get; set;}
    private int CPF {get; set}
}
```

```
public class Funcionario : Pessoa {
    private string cargo {get; set;}
    private double salário {get; set}
}
```

```
public class Terceiro : Pessoa {
    private int id {get; set;}
    private int nota_fiscal {get; set}
}
```

2. **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.**

Em orientação a objetos interface se estabelece como uma forma de "contrato", onde definimos, em uma classe do tipo interface, métodos, propriedades e eventos de forma abstrata, sem a necessidade de implementação em cada membro. A importância da utilização de interfaces em C# ou outras linguagens é que além delas permitirem o desenvolvimento de código modular, de fácil manutenção, também permitem que classes diferentes compartilhem comportamentos / ações comuns, facilitando implementações específicas por outras semelhantes.

**Exemplo:** Possuimos a interface IPagamento e as classes PagamentoCredito e PagamentoBoleto. Ambas as classes herdam da interface IPagamento o método Pagar().

```
public interface IPagamento{
    public void Pagar( double valor );
}
```

```
Public class PagamentoCredito : IPagamento {
    public void Pagar( double valor ){
        // implementar o método
    }
}
```

```
Public class PagamentoBoleto : IPagamento {
    public void Pagar( double valor ){
        // implementar o método
    }
}
```

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

Sobrecarga é a capacidade de termos vários métodos com o mesmo nome, mas com diferentes tipos ou número de parâmetros no método. Já a sobreposição ou sobrescrita é a capacidade de alterarmos a implementação de um método de uma subclasse em relação ao método definido na classe base, mas sem alterar o tipo e número de parâmetros.

**Exemplo:**

***Sobrecarga:***

```
public class Veiculo{
    public string Abastecer(double capacidade){
        // implementar o método
    }
    public string Abastecer(double capacidade, string combustivel){
        // implementar o método
    }
```

***Sobreposição:***

```
public class Geometria{
    public virtual double CalculoArea();
        return 0;
}
public class Quadrado : Geometria{
    public double lado {get; set;}
    public override double CalculoArea(){
        return lado * lado;
    }
}
```

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

- **Link do repositório com os códigos do Ex04 ao Ex10:**

  https://github.com/guilherme-rods/Avaliacao_Csharp_1bim.git