Para os exercícios abaixo, utilize todos os conceitos de orientação a objetos vistos em salas de aula. Considere o uso correto de arquitetura de código e boas práticas de programação. Não esqueça de refletir sobre o problema, desenhando a solução antes de codificar.

Utilize o Visual Studio para os exercícios, codificando na linguagem C#. Separe classes e interfaces em arquivos separados, e não esqueça de encapsular atributos. Também faça uma simulação do programa no arquivo principal do projeto (Program.cs).

Exercício 1

Você foi contratado para desenvolver um sistema de gestão de funcionários para uma empresa. O sistema deve ser capaz de lidar com diferentes tipos de funcionários, como funcionários regulares, gerentes e diretores. Cada tipo de funcionário tem diferentes atributos e comportamentos.

- Crie uma classe abstrata chamada Funcionario com os seguintes atributos e métodos:
    - Atributos: nome, idade e salarioBase.
    - Métodos: CalcularSalario() (método abstrato que será implementado nas subclasses).
- Crie as seguintes subclasses de Funcionario:
    - FuncionarioRegular: Representa um funcionário regular com salário fixo.
    - Gerente: Representa um gerente com salário fixo e uma bonificação adicional.
    - Diretor: Representa um diretor com salário fixo e uma participação nos lucros da empresa.
- Utilize herança para herdar os atributos e métodos da classe Funcionario nas subclasses, ajustando conforme necessário.
- Crie uma interface chamada AumentoSalario com o método AumentarSalario(porcentagem).

- Implemente a interface AumentoSalario nas classes Gerente e Diretor, de forma que cada uma delas possa ter um método para aumentar seus salários.
- Utilize polimorfismo ao chamar o método CalcularSalario() para cada tipo de funcionário, garantindo que o cálculo seja feito de acordo com as regras específicas de cada categoria de funcionário.
- Crie um programa principal que demonstre o funcionamento do sistema, instanciando diferentes tipos de funcionários e chamando métodos para calcular salários e aumentar salários.

Exercício 2

Você foi contratado para desenvolver um sistema de gestão de figuras geométricas para uma escola de matemática. O sistema deve ser capaz de lidar com diferentes tipos de figuras geométricas, como círculos, quadrados e triângulos. Cada tipo de figura tem diferentes atributos e comportamentos.

- Crie uma classe abstrata chamada FiguraGeometrica com os seguintes métodos abstratos:
    - CalcularArea(): Método para calcular a área da figura.
    - CalcularPerimetro(): Método para calcular o perímetro da figura.
- Implemente as seguintes subclasses de FiguraGeometrica:
    - Circulo: Representa um círculo com o atributo raio.
    - Quadrado: Representa um quadrado com o atributo lado.
    - Triangulo: Representa um triângulo com os atributos lado1, lado2 e lado3.
- Utilize herança para herdar os métodos abstratos da classe FiguraGeometrica nas subclasses, ajustando conforme necessário.
- Crie uma interface chamada Desenho com o método Desenhar(), que deve imprimir uma representação visual da figura na tela.
- Implemente a interface Desenho nas classes Circulo, Quadrado e Triangulo, de forma que cada uma delas possa ser desenhada na tela de acordo com suas características específicas.
- Crie um programa principal que demonstre o funcionamento do sistema, instanciando diferentes tipos de figuras geométricas, calculando suas áreas e perímetros, e desenhando cada uma delas na tela.

Exercício 3

Você foi contratado para desenvolver um sistema de gestão de figuras geométricas para uma escola de matemática. O sistema deve ser capaz de lidar com diferentes tipos de

figuras geométricas, como círculos, quadrados e triângulos. Cada tipo de figura tem diferentes atributos e comportamentos.

- Crie uma classe abstrata chamada ContaBancaria com os seguintes métodos abstratos:
    - Depositar(valor): Método para depositar um valor na conta.
    - Sacar(valor): Método para sacar um valor da conta.
    - CalcularJuros(): Método para calcular os juros da conta (implementação específica em cada tipo de conta).
- Implemente as seguintes subclasses de ContaBancaria:
    - ContaCorrente: Representa uma conta corrente com atributos adicionais como saldo e taxaManutencao.
    - ContaPoupanca: Representa uma conta poupança com atributos adicionais como saldo e taxaJuros.
    - ContaInvestimento: Representa uma conta de investimento com atributos adicionais como saldo e rendimentoAnual.
- Utilize herança para herdar os métodos abstratos da classe ContaBancaria nas subclasses, ajustando conforme necessário.
- Crie uma interface chamada MovimentacaoConta com os métodos depositar() e sacar(), que serão implementados em cada tipo de conta.
- Implemente a interface MovimentacaoConta nas classes ContaCorrente, ContaPoupanca e ContaInvestimento, de forma que cada uma delas possa realizar operações de depósito e saque de acordo com suas características específicas.
- Crie um programa principal que demonstre o funcionamento do sistema, instanciando diferentes tipos de contas bancárias, realizando operações de depósito, saque e cálculo de juros.

Exercício 4

Suponha que você está desenvolvendo um sistema de gestão de animais para um zoológico. O sistema deve ser capaz de lidar com diferentes tipos de animais, como mamíferos, aves e répteis. Cada tipo de animal tem diferentes características e comportamentos.

- Crie uma classe abstrata chamada Animal com o atributo espécie, e os seguintes métodos abstratos:
    - EmitirSom(): Método para que o animal emita um som característico.
    - Movimentar(): Método para simular o movimento do animal.
- Implemente as seguintes subclasses de Animal:
    - Mamifero: Representa um mamífero com atributos corPelagem e tipoAlimentacao.
    - Ave: Representa uma ave com atributos como corPenas e tipoVoo.
    - Reptil: Representa um réptil com atributos como comprimento e habitat.

- Utilize herança para herdar os métodos abstratos da classe Animal nas subclasses, ajustando conforme necessário.
- Crie uma interface chamada Visitavel com o método ExibirInformacoes(), que será implementado em cada tipo de animal.
- Implemente a interface Visitavel nas classes Mamifero, Ave e Reptil, de forma que cada uma delas possa exibir suas informações específicas quando visitadas no zoológico.
- Crie a classe Zoologico, que possui uma lista de Animal. Encapsule a lista de Animal de forma que apenas Zoologico consiga adicionar ou remover animais. Crie, também, os seguintes métodos:
    - ObservarAnimal(int): exibe as informações do animal no índice especificado
    - QuantidadeAnimais(): exibe a quantidade de animais do zoológico.
- Crie a classe abstrata Visitante, que possui o método abstrato Visitar(Zoologico).
- Crie a classe VisitanteLento e VisitanteRapido, que herdam de Visitante e sobrescrevam o método Visitar. VisitanteLento deverá olhar os animais de 1 em 1, enquanto o VisitanteRapido deverá olhar os animais de 2 em 2
- Crie um programa principal que demonstre o funcionamento do sistema, instanciando diferentes tipos de animais, fazendo-os emitir som, movimentar-se e exibindo suas informações quando visitados no zoológico.

Exercício 5

Suponha que você está desenvolvendo um sistema para gerenciar produtos em um estoque de uma loja. O sistema deve ser capaz de lidar com diferentes tipos de produtos, como eletrônicos, roupas e alimentos. Cada tipo de produto tem diferentes características e comportamentos.

- Crie uma classe abstrata chamada Produto com os seguintes métodos abstratos:
    - CalcularPreco(): Método para calcular o preço do produto.
    - ExibirInformacoes(): Método para exibir as informações do produto.
- Implemente as seguintes subclasses de Produto:
    - Eletronico: Representa um produto eletrônico com atributos como marca, modelo e preco.
    - Roupa: Representa uma peça de roupa com atributos como tipo, tamanho e preco.
    - Alimento: Representa um alimento com atributos como nome, dataValidade e preco.
- Utilize herança para herdar os métodos abstratos da classe Produto nas subclasses, ajustando conforme necessário.
- Crie uma classe abstrata chamada Loja, que possui o atributo marca os métodos chamados VenderProduto(Produto) e VenderProduto(Produto, quantidade)
- Crie uma classe chamada Loja1, que possui um dicionário onde a chave é o tipo do produto, e o valor é um inteiro que determina a quantidade daquele produto no estoque.

- Crie uma classe chamada Loja2, que possui uma lista de Produto
- Crie uma interface chamada Estocável, que possui um método chamado Estocar(Produto). Implemente a interface Estocável em Loja1 e Loja2
- Crie um programa principal que demonstre o funcionamento do sistema, instanciando diferentes tipos de Produto, e instancie Lojas, estocando produtos e vendendo os produtos.

Exercício 6

Você foi contratado para desenvolver um sistema para instituições de ensino alocar alunos em disciplinas.

- Um aluno deve ter nome, cpf, e deve poder se matricular em uma disciplina
- Um professor também possui nome e cpf
- Uma disciplina possui uma lista de alunos, um professor, uma relação das médias dos alunos, e uma média base para identificar se um aluno foi aprovado ou não. Deve existir uma forma de calcular a média de todos os alunos a partir de suas notas. Da mesma forma, deve ser possível, a partir da disciplina, saber se um aluno foi aprovado ou não.
- Uma disciplina de exatas possui notas de provas e notas de trabalhos.
- Uma disciplina de humanas possui notas de trabalhos e uma lista de presença que identifica um aluno e sua taxa de participação na aula, indo de 0 a 100.
- Cada disciplina possui pesos diferentes para calcular se um aluno foi aprovado ou não. Uma aula de Cálculo, por exemplo, usa 20% das notas dos trabalhos e 80% das notas das provas, enquanto uma aula de Desenho Artístico possui 50% das notas dos trabalhos e 50% da participação em sala de aula
- Deve ser possível, a partir de cada Disciplina, inserir as notas dos alunos e gerar suas médias, imprimindo um relatório mostrando cada aluno com sua respectiva média e se está aprovado ou não. Ao final do relatório, adicione a taxa de aprovação de alunos. Somente o professor da disciplina é capaz de realizar tais operações.

Exercício 7

Você foi contratado pela Rebu, um novo aplicativo de transporte que faz aluguel de carros para transporte de passageiros sob demanda. Você deverá fazer um sistema que permite que pessoas solicitem por motoristas para as levarem para um destino.

Uma pessoa possui nome, CPF e avaliação de 0 a 5. Ela pode ter uma corrida atual e um histórico de corridas.

Uma corrida pertence a uma pessoa, e tem data, um endereço de origem, um endereço de destino e um motorista.

Um motorista é uma pessoa que pode aceitar uma corrida ou não. Se estiver uma corrida atual, pode finalizar uma corrida ou não. Ao finalizar uma corrida, deve ser computado a distancia percorrida e calcular o total ganho pelo motorista.
Existem dois tipos de motoristas

- Motoristas padrão cobram R$ 2 por km percorrido
- Motoristas VIP cobram R$ 4 por km percorrido.

Um passageiro é uma pessoa que pode solicitar uma corrida. Ela possui uma localização atual e, ao solicitar uma corrida, deve informar se quer uma corrida com motorista padrão ou motorista VIP.

Existem passageiros que podem falar em inglês e só podem solicitar corridas de motoristas que falam em inglês. Um motorista que fala inglês deve iniciar a corrida falando "Hello".

Exercício 8

Você foi contratado para criar o mais novo jogo de RPG do mercado. A empresa LevelDown quer fazer um jogo de fantasia, no qual um jogador controla um personagem em diferentes aventuras.

Um personagem possui um nome, nível, quantidade de HP, quantidade de Mana, valor de ataque e valor de defesa. Um personagem também pode atacar outro personagem.

Um personagem pode ter 3 tipos diferentes:

- Guerreiro:
    - Valor de Ataque: 20
    - Valor de Defesa: 40
    - Quantidade de HP: 50
    - Quantidade de Mana: 20
- Arqueiro:
    - Valor de Ataque: 40
    - Valor de Defesa: 30
    - Quantidade de HP: 40
    - Quantidade de Mana: 30
- Mago:
    - Valor de Ataque: 50,
    - Valor de Defesa: 30
    - Valor de HP: 20
    - Quantidade de Mana: 50

Ao atacar um personagem, deve ser reduzido do HP do personagem atacado o valor do ataque do personagem atacante menos o valor da defesa do personagem atacado. Se o HP do personagem atacado reduzir para 0, então o personagem atacado desmaia.

Cada personagem possui uma habilidade especial que consome 10 de mana.

- Um Guerreiro ataca duas vezes
- Um Mago causa 1.5x de dano
- Um Arqueiro tem 30% de chance de causar um acerto crítico (2x de dano).

Crie, também, formas de se jogar contra uma IA. Para isso, crie estratégias para se jogar automaticamente.

Faça uma aplicação de formulários que permita que o jogador veja a interface do jogo.

Exercício 9

Crie um Jogo de Batalha Naval orientado a objetos.

Exercício 10

Crie um Jogo de UNO! orientado a objetos.