Universidade Federal de Uberlândia
Faculdade de Computação – Gestão da Informação – Prof. Daniel A. Furtado
3° Trabalho de Programação Orientada a Objetos
Introdução à Classes e Objetos

Instruções Gerais

- Esta atividade deve ser realizada **individualmente**;
- Esteja atento às **observações sobre plágio** apresentadas no final desde documento;
- Trabalhos com implementações utilizando trechos de códigos retirados de sites da Internet, gerados por ferramentas de IA (como ChatGPT, Gemini e similares), copiados de trabalhos de semestres anteriores, serão anulados;
- O trabalho deve ser entregue até a data/hora definida pelo professor. Não deixe para enviar o trabalho nos últimos instantes, pois eventuais problemas relacionados à eventos adversos como instabilidade de conexão, congestionamento de rede etc., não serão aceitos como motivos para entrega da atividade por outras formas ou em outras datas;
- Trabalhos enviados por e-mail ou pelo MS Teams **não serão considerados**.

Material de Apoio

https://furtado.prof.ufu.br/site/teaching/POO/POO-Modulo2-Fundamentos.pdf (slides 1-36)

Exercício 1

Crie um novo projeto no Visual Studio Community e defina uma classe em C# para representar veículos. Deve haver atributos para armazenar a **marca**, o **modelo**, o **ano** e a **velocidade** do veículo. O construtor deve permitir a criação de novos objetos informando apenas a **marca**, o **modelo** e o **ano**.

- A velocidade inicial de todo veículo deve ser 0.
- Deve haver um método para acelerar o veiculo e outro para pará-lo instantaneamente.
- A classe deve incluir também um método para apresentação dos dados do veículo.

Crie um programa principal para ilustrar o uso da classe. O programa deve criar pelo menos dois objetos da classe e deve utilizar todos os métodos implementados.

Exercício 2

Crie um novo projeto no Visual Studio Community e defina uma classe em C# para modelar um vetor geométrico no espaço R^3 conforme especificações a seguir:

- A classe deve ter atributos do tipo **double** para armazenar as coordenadas A, B e C do vetor;
- A classe deve ter um construtor que permita a criação de novos vetores informando as coordenadas A, B e C;
- A classe deve ter um segundo construtor, sem parâmetros, que permita a criação de novos vetores com coordenadas iguais a 1. O construtor não deve solicitar dados ao usuário;
- A classe deve ter um método que retorne o **módulo** do vetor, dado pela fórmula $\sqrt{A^2 + B^2 + C^2}$. O método não deve imprimir na tela o resultado do cálculo. Deve calcular e

retornar o valor. Para calcular a raiz quadrada de um número, utilize o método Math.Sqrt(número);

- A classe deve ter um método para **normalizar** o vetor. Para isso, o método deve atualizar o valor de suas coordenadas dividindo cada uma delas pelo módulo do vetor. Para calcular o módulo, utilize o método definido anteriormente;
- A classe deve ter um método que exiba as coordenadas do vetor em uma string bem formatada.

Crie um programa principal para ilustrar o uso da classe. O programa deve criar pelo menos dois vetores, usando os dois construtores, e deve utilizar todos os métodos implementados.

Exercício 3

Crie um novo projeto no Visual Studio Community e defina uma classe em C# para modelar uma conta bancária. A classe deve possuir atributos para armazenar:

- o número da conta (string),
- o número da agência (string),
- o nome do titular da conta (string) e
- o saldo da conta (decimal).

A classe deve ter um método construtor que permita a criação de novas contas informando o número da conta, a agência e o nome do titular. O saldo não deve ser passado como parâmetro na criação da conta, pois toda conta deve ter um saldo inicial igual a zero.

A classe deve possuir:

- Um método **Depositar** com um parâmetro **valor** que permita fazer um depósito na conta. O método deve **acrescentar** o valor passado como parâmetro no saldo da conta. O método não deve solicitar dados ao usuário nem imprimir mensagens na tela.
- Um método **Retirar** com um parâmetro **valor** que permita fazer um "saque" na conta. O método deve subtrair o valor passado como parâmetro do saldo da conta caso haja saldo suficiente. Caso contrário, o método deve exibir a mensagem "Operação não realizada: saldo insuficiente.".
- Um método **Mostrar** para apresentar todas as informações da conta na forma de uma string adequadamente formatada.

Para ilustrar o uso da classe, crie um programa principal que exiba, dentro de um laço do-while (conforme trabalho anterior), um menu com as seguintes opções:

1 – Criar nova conta
2 – Depositar
3 – Sacar
4 – Mostrar dados da conta
5 – Sair

Quando o usuário escolher a opção 1, o programa deve solicitar os dados da nova conta e então criar um novo objeto para representar a respectiva conta.
Para a opção 2, o programa deve solicitar ao usuário o valor a ser depositado e em seguida chamar o respectivo método do objeto para efetuar o depósito. A opção 3 deve ter funcionamento análogo. A opção 4 deve apresentar todos os dados da conta, inclusive o saldo corrente.

Entrega

Compacte as pastas dos projetos dos exercícios em um único arquivo no formato **zip** e envie pelo Sistema **SAAT** até a data limite indicada pelo professor em sala de aula. Caso não tenha o **Visual Studio Community**, coloque o código criado para cada exercício em um arquivo .cs (exercicio1.cs, exercicio2.cs etc.), compacte todos os arquivos e envie o arquivo compactado.

Sobre Eventuais Plágios

Este é um trabalho individual. Os alunos envolvidos em qualquer tipo de plágio, total ou parcial, seja entre equipes ou de trabalhos de semestres anteriores ou de materiais disponíveis na Internet (exceto os materiais de aula disponibilizados pelo professor), serão duramente penalizados (art. 196 do Regimento Geral da UFU). Todos os alunos envolvidos terão seus **trabalhos anulados** e receberão **nota zero**.