Universidade Federal de Uberlândia
Faculdade de Computação – Gestão da Informação – Prof. Daniel A. Furtado
4º Trabalho de Programação Orientada a Objetos
Classes, Objetos, Modificadores de Acesso e Encapsulamento

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 28-55)

Exercício 1

Crie um projeto no Visual Studio Community e defina uma classe em C# para representar **veículos** conforme especificações a seguir:

- Deve haver atributos **privados** para armazenar o **modelo**, o **ano** e a **velocidade** do veículo:
- O construtor deve permitir que o usuário informe apenas o **modelo** e o **ano** no momento de criar os objetos;
- A velocidade **inicial** de todo veículo deve ser **0**;
- O **modelo** e o **ano** do veículo devem ser expostos ao código externo apenas para **leitura** por meio de métodos **get** (GetModelo/GetAno);
- A velocidade deve ser exposta para **leitura** e **atualização**, porém não deve ser permitida uma velocidade negativa;
- Em nenhum local do código interno da classe deve haver apresentação de mensagens ao usuário.

Crie um programa principal para ilustrar o uso da classe. O programa deve criar dois objetos da classe. O primeiro objeto deve representar um veículo modelo “ABC” e ano 2012. O modelo e o ano do segundo veículo devem ser solicitados ao usuário. O código deve ilustrar o uso de todos os métodos definidos na classe.

Exercício 2

Crie um projeto no Visual Studio Community e defina uma classe em C# para modelar um **cilindro** (forma geométrica no espaço), conforme especificações a seguir:

- Deve haver atributos **privados** do tipo **double** para armazenar o **raio** da base do cilindro e a **altura**;

- Deve haver um **construtor** que permita a criação de novos cilindros informando o **raio** da base e a **altura**. O construtor não deve permitir a criação de cilindros com raio ou altura nulos ou negativos;
- Deve haver um segundo construtor, sem parâmetros, que permita a criação de novos cilindros com tamanho padrão igual a 1 (raio e altura iguais a 1);
- Deve haver um método **privado** que calcule e retorne a área da base do cilindro, dado pela fórmula $area\ base = \pi r^2$;
- Deve haver um método **público** que calcule e retorne o volume do cilindro, dado pela fórmula $volume = área\ da\ base \cdot altura$. O método definido anteriormente para calcular a área da base deve ser chamado. Nenhuma mensagem deve ser apresentada pelo método;
- Deve haver um método **público** para calcular e retornar a área lateral do cilindro, dado pela forma $área\ lateral = 2\pi rh$. Nenhuma mensagem deve ser apresentada pelo método;
- O atributo raio deve ser exposto para acesso externo para fins de **leitura** (GetRaio);
- O atributo altura deve ser exposto para acesso externo para **leitura** e **atualização**, mas não deve permitir um novo valor que seja 0 ou negativo;
- Deve haver um método **Mostrar** para que o objeto apresente, em uma única mensagem, todas as suas características: raio da base, altura, área lateral e volume;
- Deve haver um método de nome **EhIgualA** que receba um outro objeto do tipo **Cilindro** como parâmetro e retorne **true** caso o objeto atual possua raio e altura iguais ao do cilindro passado como parâmetro. Caso contrário, deverá retornar **false**. Veja o slide 42.

Crie um programa principal para ilustrar o uso da classe. O programa deve declarar um array de cilindros para armazenar até 5 objetos (var arrayCilindros = new Cilindro[5]). Em seguida, deve criar 5 objetos do tipo Cilindro utilizando um laço **for**. Cada objeto criado deve ser armazenado na próxima posição do *array* de cilindros. O raio e a altura de cada cilindro devem ser solicitados ao usuário.

Após criação dos objetos, acrescente um segundo **for** para somar os volumes dos 5 cilindros do array. O resultado deve ser apresentado ao usuário.

Acrescente um terceiro **for** para somar as áreas laterais dos cilindros.

Acrescente um quarto **for** para apresentar todas as informações de todos os cilindros usando o método **Mostrar**.

Por fim, o programa deve mostrar uma mensagem indicando se o **primeiro** cilindro do array é igual ao **segundo** cilindro em termos de raio e altura. Deve ser utilizado o método **EhIgualA**.

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**.