UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE PROCESSAMENTO DE ENERGIA ELÉTRICA

Exercícios

Curso de Engenharia de Controle e Automação

DPEE1090 - Programação orientada a objetos para automação

Prof. Renan Duarte

1º semestre de 2024

Exercícios

https://bit.ly/3xRPuEC

Exercício 1

Desenvolva uma classe chamada `Triangle` para representar um triângulo equilátero, ou seja, um trilátero regular com três lados de mesmo comprimento.

A classe possui um único atributo denominado `side`, do tipo `double`, que representa o lado do triângulo equilátero e cujo valor **deve ser maior ou igual a zero e menor ou igual a vinte**.

A classe possui dois construtores: o primeiro configura o lado do triângulo equilátero com o valor padrão 1.0, e o segundo recebe como parâmetro o lado do triângulo equilátero.

O lado do triângulo equilátero pode ser obtido e alterado pelo usuário por meio dos métodos `GetSide()` e `SetSide()`, respectivamente.

A classe também apresenta os métodos `Area()` e `Perimeter()`, que retornam a área e o perímetro do triângulo equilátero, respectivamente.

$$Area = \frac{\sqrt{3}}{4} l^2 \qquad Perimeter = 3l$$

Exercícios

https://bit.ly/4baDP25

Exercício 2

Escreva uma classe auxiliar para a validação de um CPF digitado pelo usuário. O CPF deve ser digitado no formato 11144477735 (somente números) e salvo no formato de string.

Algoritmo de validação do CPF:

1. Calcular o primeiro dígito verificador: separamos os primeiros 9 dígitos do CPF (111444777) e multiplicamos cada um dos números, da direita para a esquerda por números crescentes a partir do número 2:

Dígito	1	1	1	4	4	4	7	7	7
Multiplicador	10	9	8	7	6	5	4	3	2
Resultado	10	9	8	28	24	20	28	21	14

Somamos todos os resultados e dividimos por 11, salvando o quociente e o resto:

10+9+8+28+24+20+28+21+14 = 162 → **Quociente = 14, Resto = 8**

Exercícios

https://bit.ly/4baDP25

Exercício 2 - Continuação

- Se o resto da divisão for menor que 2, então o dígito é igual a 0
- Se o resto da divisão for maior ou igual a 2, então o dígito verificador é igual a 11 menos o resto da divisão (11 - resto)

No nosso exemplo, o dígito verificador é 3

3. Calcular o segundo dígito verificador: separamos os primeiros 10 dígitos do CPF (1114447773) e multiplicamos cada um dos números, da direita para a esquerda por números crescentes a partir do número 2

Dígito	1	1	1	4	4	4	7	7	7	3
Multiplicador	11	10	9	8	7	6	5	4	3	2
Resultado	11	10	9	32	28	24	35	28	21	6

Somamos todos os resultados e dividimos por 11, salvando o quociente e o resto:

11+10+9+32+28+24+35+28+21+6 = 204 → **Quociente = 18, Resto = 6**

Exercícios

Exercício 2 - Continuação

- Se o resto da divisão for menor que 2, então o dígito é igual a 0
- Se o resto da divisão for maior ou igual a 2, então o dígito verificador é igual a 11 menos o resto da divisão (11 - resto)

No nosso exemplo, o dígito verificador é 5

4. Comparar os dois dígitos verificadores calculados com os dois dígitos verificadores do CPF (11144477735)

Apenas se ambos forem iguais, o CPF é válido

Exercícios

https://bit.ly/3UwKjmk

Exercício 3

Crie uma classe chamada `Invoice` que possa ser utilizada por uma loja de suprimentos de informática para representar uma fatura de um item vendido na loja. Uma fatura deve incluir as seguintes informações como atributos:

- O código do item faturado
- A descrição do item
- A quantidade comprada do item
- O preço unitário do item

Sua classe deve ter um construtor que inicialize os quatro atributos. Se a quantidade não for positiva, ela deve ser configurada como 0. Se o preço por item não for positivo ele deve ser configurado como 0.0. Forneça `getters` e `setters` para cada variável de instância.

Além disso, forneça um método chamado `GetInvoiceAmount` que calcula o valor da fatura (isso é, multiplica a quantidade pelo preço por item) e depois retorna o valor e um método ToString que imprime todos os dados da fatura, incluindo o total.

Escreva um aplicativo de teste que demonstra as capacidades da classe `Invoice`.

Exercícios

https://bit.ly/3Ugpmuv

Exercício 4

Escreva uma classe `NumeroComplexo`, que representa um número complexo (R + jI). A classe deve fornecer as seguintes operações:

a) Construtor com valores das partes real e imaginária: formato (R, I)

b) Construtor apenas com parte real (parte imaginária é definida como zero)

c) Métodos para ler e definir cada um dos atributos do número

d) Método somar, que recebe outro número complexo e o adiciona ao número complexo que recebeu a mensagem: (a+bi)+(c+di) = (a+c)+(b+d)i

e) Método subtrair, que recebe outro número complexo e o subtrai do número complexo que recebeu a mensagem: (a+bi)−(c+di) = (a−c)+(b−d)i

f) Um método para verificar se dois números são iguais

g) Um método que gere e retorne a representação string do número complexo no formato (R, I)

Implementa um programa para teste da classe

Dúvidas?

renan.duarte@gedre.ufsm.br

GEDRE – Prédio 10 – CTLAB