Exercícios

Vamos “Codar OO”...

# Exercício

## Cenário: Gestão de Pessoas no “IBGE” ->

* + Escreva um algoritmo que receba a idade de um grupo de pessoas, calcule e imprima a porcentagem de cada faixa etária em relação ao total de pessoas.
  + As faixas etárias são:
    - 01 - 15 anos - A
    - 16 - 30 anos - B
    - 31 - 45 anos - C
    - 46 - 60 anos - D
    - >= 61 anos - E
  + Para encerrar a entrada de dados, digite uma idade <= 0.

List<int> idades = new List<int>();

//List<int> idades = new();

List<float> A = [];

List<float> B = [];

List<float> C = [];

List<float> D = [];

List<float> E = [];

Console.WriteLine("Para encerrar, digite uma idade <= 0\n");

while (true)

{

Console.Write("Idade: ");

int idd = int.Parse(Console.ReadLine());

if (idd <=0)

break;

idades.Add(idd);

if (idd == 1 || idd <= 15)

{

A.Add(idd);

}

else if (idd == 16 || idd <= 30)

{

B.Add(idd);

}

else if (idd == 31 || idd <= 45)

{

C.Add(idd);

}

else if (idd == 46 || idd <= 60)

{

D.Add(idd);

}

else {

E.Add(idd);

}

}

float total = idades.Count;

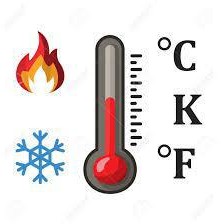
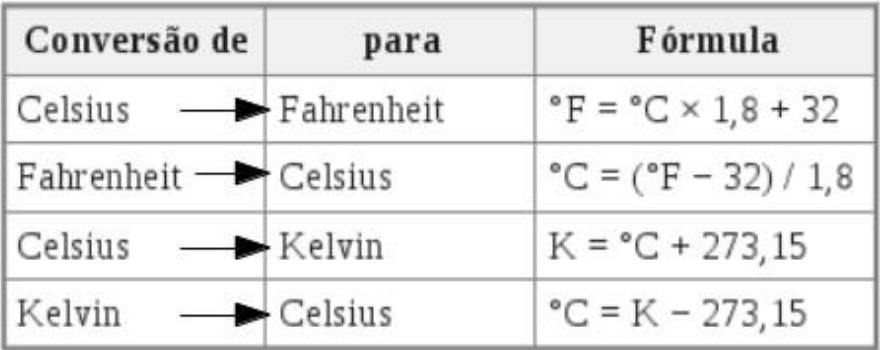
Console.WriteLine($"\nTotal digitado: {total}");

Console.WriteLine($"\nFaixas Etárias:\n\n•\t01 - 15 anos - A - {(A.Count/total)\*100}%\r\n•\t16 - 30 anos - B - {(B.Count/total)\*100}%\r\n•\t31 - 45 anos - C - {(C.Count/total)\*100}%\r\n•\t46 - 60 anos - D - {(D.Count/total)\*100}%\r\n•\t>= 61 anos - E - {(E.Count/total)\*100}%\r\n");

# Exercício

## Cenário: Verão Capixaba 01 ->

* + Faça um algoritmo para realizar as conversões de temperatura, de acordo com as fórmulas abaixo:



Console.WriteLine("\n1. Celsius p Fahrenheit");

Console.WriteLine("2. Fahrenheit p Celcius");

Console.WriteLine("3. Celsius p Kelvin");

Console.WriteLine("4. Kelvin p Celsius");

Console.Write("\nDigite uma opção: ");

int n = int.Parse(Console.ReadLine());

switch (n)

{

case 1:

Console.Write("\nCelsius: ");

float i = float.Parse(Console.ReadLine());

Console.WriteLine($"\n°F = {i \* 1.8 + 32}");

break;

case 2:

Console.Write("\nFahrenheit: ");

float f = float.Parse(Console.ReadLine());

Console.WriteLine($"\n°C = {(f - 32) / 1.8}");

break;

case 3:

Console.Write("\nCelsius: ");

float c = float.Parse(Console.ReadLine());

Console.WriteLine($"\nK = {c + 273.15}");

break;

case 4:

Console.Write("\nFahrenheit: ");

float p = float.Parse(Console.ReadLine());

Console.WriteLine($"\n°C = {p - 273.15}");

break;

default:

Console.WriteLine("\nDigite um número válido!");

break;

# Exercício

## Cenário: Verão Capixaba 02 ->

* + Os comerciantes das Praias de Vila-Velha querem saber a temperatura média do verão capixaba.
  + Para isso é necessário escrever um algoritmo que leia a temperatura diária no verão, durante 15 dias, e exiba a temperatura média da estação.
  + Deve impedir a entrada de dados de uma temperatura fora da estação.
  + Sabe-se que as temperaturas, na estação de verão Capixaba, ficam acima de 28 °C.

\\ fzr while (true)

List<float> dias = new List<float>();

float total = 0;

for (int i = 1; i <= 15; i++)

{

Console.Write($"Digite a temperatura do dia {i}: ");

float temp = float.Parse(Console.ReadLine());

if (temp <= 28)

{

Console.WriteLine("\nDigite uma temperatura válida! (> 28)");

break;

}

else

{

dias.Add(temp);

total += temp;

}

}

float media = total / dias.Count;

Console.WriteLine($"\nA Média da Estação é: {media}ºC");

# Exercício

## Cenário: Gestão de Recursos na “Supermercado da Mãe” ->

* + Faça um algoritmo que leia o preço e o código de 20 mercadorias presentes em uma cesta básica. Os códigos obedecem à tabela abaixo:
    - ‘l’ ou ‘L’ : limpeza
    - ‘a’ ou ‘A’ : alimentação
    - ‘h’ ou ‘H’ : higiene
  + Calcule e imprima:
    - O preço total da cesta básica; e
    - Para cada código, o preço total das mercadorias presentes na cesta.

List<float> cesta = new List<float>();

List <float> L = [];

List<float> A = [];

List<float> H = [];

float preco;

for (int i = 1; i <= 6; i++)

{

Console.Write($"Digite o preço da Mercadoria {i}: ");

preco = float.Parse(Console.ReadLine());

cesta.Add(preco);

while (true)

{

Console.Write($"Digite o código da Mercadoria {i}: ");

char cod = char.Parse(Console.ReadLine());

Console.WriteLine();

if (cod == 'l' || cod == 'L')

{

L.Add(preco);

break;

}

else if (cod == 'a' || cod == 'A')

{

A.Add(preco);

break;

}

else if (cod == 'h' || cod == 'H')

{

H.Add(preco);

break;

}

else

{

Console.WriteLine("Digite um código válido!\n");

}

}

}

float total = cesta.Sum();

float totalL = L.Sum();

float totalA = A.Sum();

float totalH = H.Sum();

Console.WriteLine($"O preço total da cesta é: R${total:F2}\n");

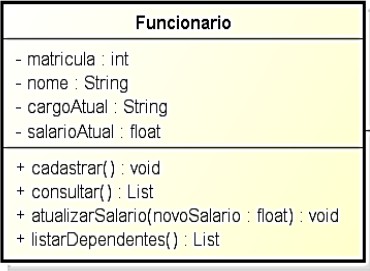
Console.WriteLine($"O preço total da Limpeza é: R${totalL:F2}");

Console.WriteLine($"O preço total da Alimentação é: R${totalA:F2}");

Console.WriteLine($"O preço total da Higiene é: R${totalH:F2}");

# Exercício

* Cenário: Gestão de RH de uma Empresa -> Crie uma classe de acordo com a representação abaixo.



* Os métodos simplesmente imprimem um texto (*String*) dizendo que eles fazem e os parâmetros que eles recebem.
* Logo depois crie uma classe que utilize os atributos e métodos implementados.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CodarOO

{

internal class Funcionario

{

// Atributos

private int matricula;

private string nome;

private string cargoAtual;

private float salarioAtual;

// Propriedades (para acesso aos atributos)

public int Matricula { get => matricula; set => matricula = value; }

public string Nome { get; set; }

public string CargoAtual { get; set; }

public float SalarioAtual { get; set; }

public void cadastrar(int m, string n, string c, float s)

{

this.matricula = m;

this.nome = n;

this.cargoAtual = c;

this.salarioAtual = s;

Console.WriteLine($"Funcionário {nome} cadastrado com sucesso");

}

public void consultar()

{

Console.WriteLine("\n--- Consulta de Funcionário ---");

Console.WriteLine($"Matrícula: {matricula}");

Console.WriteLine($"Nome: {nome}");

Console.WriteLine($"Cargo Atual: {cargoAtual}");

Console.WriteLine($"Salário Atual: {salarioAtual:C}");

}

public void atualizarSalario(float novoSalario)

{

this.salarioAtual = novoSalario;

Console.WriteLine($"Salário Atualizado com Sucesso: {novoSalario}");

}

public void listarDependentes()

{

Console.WriteLine($"\nListando dependentes de {nome}: \n...");

}

}

}

namespace CodarOO

{

internal class Program

{

static void Main(string[] args)

{

Funcionario f = new Funcionario();

f.cadastrar(123, "Daphne", "Chefe", 1800);

f.consultar();

f.atualizarSalario(2000);

f.listarDependentes();

}

}

}

Exercício

* Cenário: Uma Loja de Roupas online -> Você foi contratado para desenvolver um software para gerenciar as vendas de uma loja de roupas online. Para isso, você precisa implementar uma classe Produto e instanciar objetos dessa classe para representar cada item de roupa disponível na loja.
* Crie uma classe Produto com os seguintes atributos:
  + nome (String): o nome do produto.
  + preco (double): o preço do produto.
  + estoque (int): a quantidade de produtos em estoque.
  + Implemente Construtores
  + Implemente métodos para:
    - Obter e atualizar os dados da classe
    - Um método para calcular o valor total de produtos no estoque
* Crie uma classe Loja para instanciar e manipular 3 produtos diferentes dessa loja.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CodarOO

{

internal class Produto

{

private String nome;

private double preco;

private int estoque;

public Produto(string nome, double preco, int estoque)

{

this.nome = nome;

this.preco = preco;

this.estoque = estoque;

}

public void obterP()

{

Console.WriteLine($"Produto:\n{nome}\n{preco}\n{estoque}\n");

}

public void atualizarN(string n)

{

this.nome = n;

Console.WriteLine("Nome Atualizado com Sucesso\n");

obterP();

}

public void atualizarP(double p) {

this.preco = p;

Console.WriteLine("Preço atualizado com Sucesso\n");

obterP();

}

public void atualizarE(int e)

{

if (e > 0)

{

this.estoque += e;

}

else

{

this.estoque -= e;

}

Console.WriteLine("Estoque atualizado com Sucesso\n");

obterP();

}

public double valorTotal()

{

return preco \* estoque;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace CodarOO

{

internal class Loja

{

private string nome;

public Loja(string n)

{

this.nome = n;

}

}

}

namespace CodarOO

{

internal class Program

{

static void Main(string[] args)

{

Produto f = new Produto("Mel", 10, 2);

Produto f1 = new Produto("Açúcar", 20, 3);

Produto f2 = new Produto("Celular", 1000, 1);

Loja l = new Loja("Baggage Brazil");

f.obterP();

f1.obterP();

f2.obterP();

f.atualizarN("Mel Doce");

f1.atualizarP(22);

f2.atualizarE(2);

Console.WriteLine(value: $"Valor Total da Loja: {f.valorTotal() + f1.valorTotal() + f2.valorTotal()}");

}

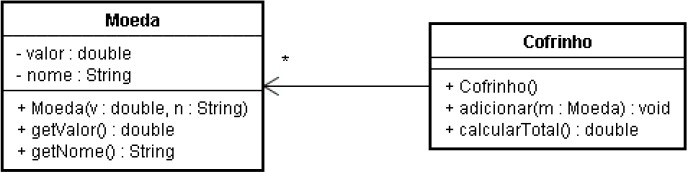
}

}

# Exercício

1/2

* + Cenário: Cofrinho do Tio Patinhas -> Implementar um cofrinho de moedas com a capacidade de receber moedas e calcular o total depositado no cofrinho.
    - Implementa uma coleção de Moeda como uma lista.
    - Faça um classe de teste



# Exercício

2/2

* Altere a classe Cofrinho de modo que ela implemente métodos para:
  + Contar o número de moedas armazenadas
  + Contar o número de moedas de um determinado valor
  + Informar qual a moeda de maior valor

# Exercício

* Cenário: Google Tradutor -> Escreva uma aplicação “Dicionário” com as seguintes funções:
  + Adicionar uma palavra e seu significado ao dicionário
  + Recuperar uma palavra e seus significados no dicionário
  + Remover um significado da Palavra
  + Pesquisar se o significado está contido no lista de significados da palavra
* Observe a seguinte modelagem: Uma Palavra do Dicionário tem Vários Significados

|  |
| --- |
| Palavra |
| +nome: String  +significados: List<Significado> |
| +adicionarSignificado(Significado texto): void  +recuperarSignificado(): List<Significado>  +removerSignificado(Significado texto): void  +pesquisarSignificado(Significado texto): boolean |

|  |
| --- |
| Significado |
| +texto: String  +palavra: Palavra |
|  |