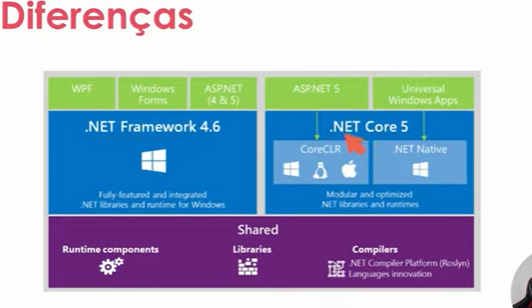


.NET – é uma plataforma de desenvolvimento

**.net Framework – versão legada do .net – está integrado somente com a plataforma Windows.**

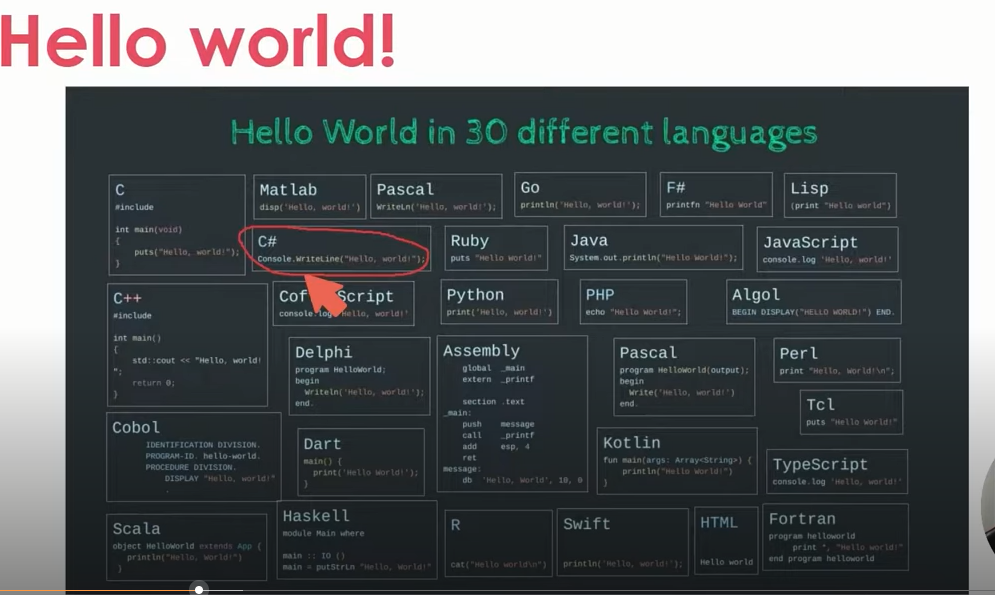
**.net core ou .net – é a versão atual. – funciona em multiplataformas ( Windows, Linux, mac os). Escrever um único código e ele executar multiplataforma.**





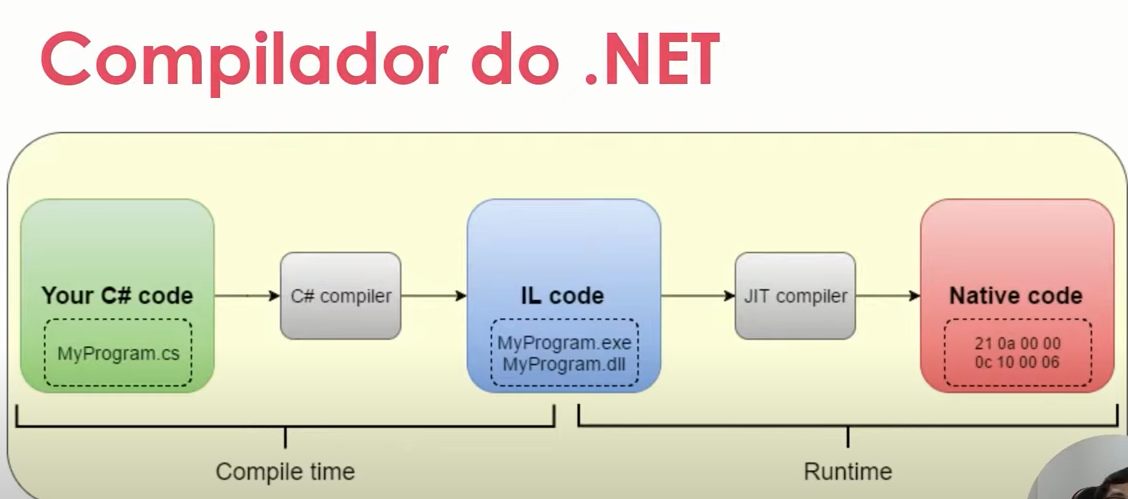
Framework, versão legada que vai até a 4.8

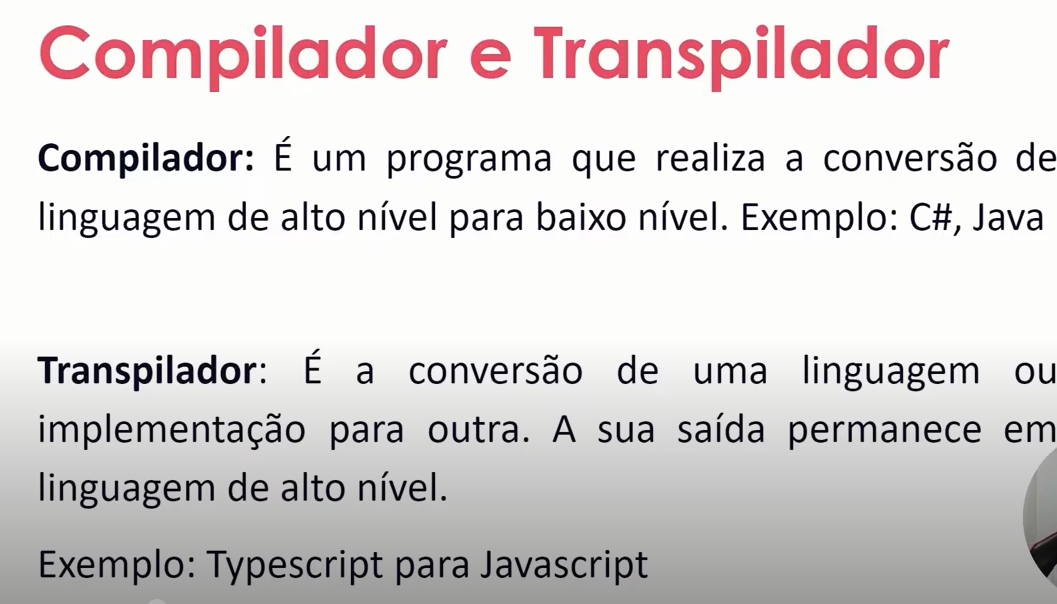
**Compilador .NET**

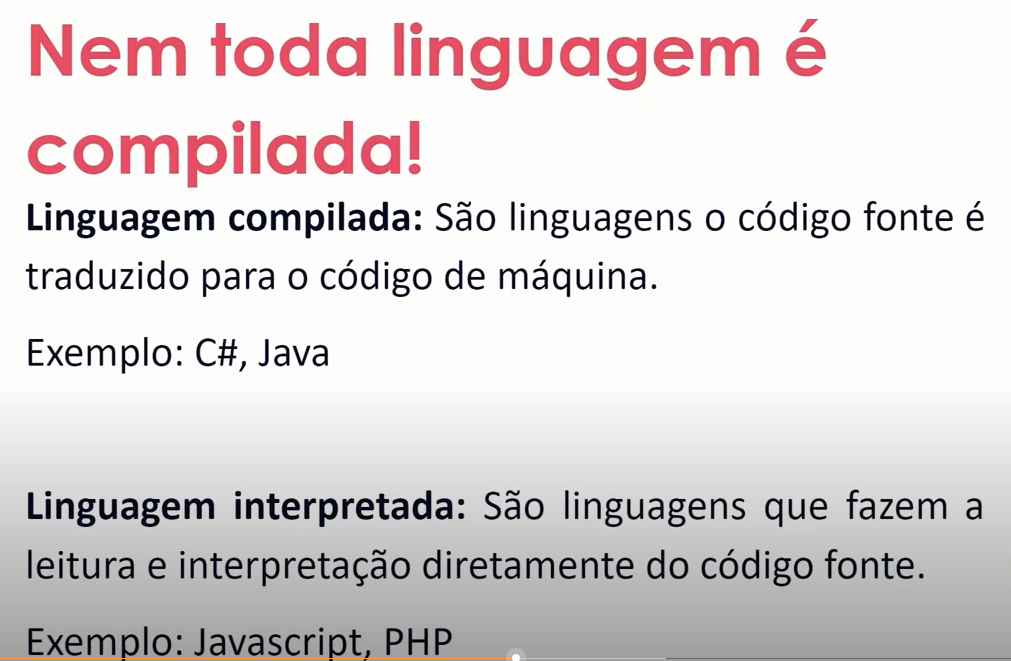


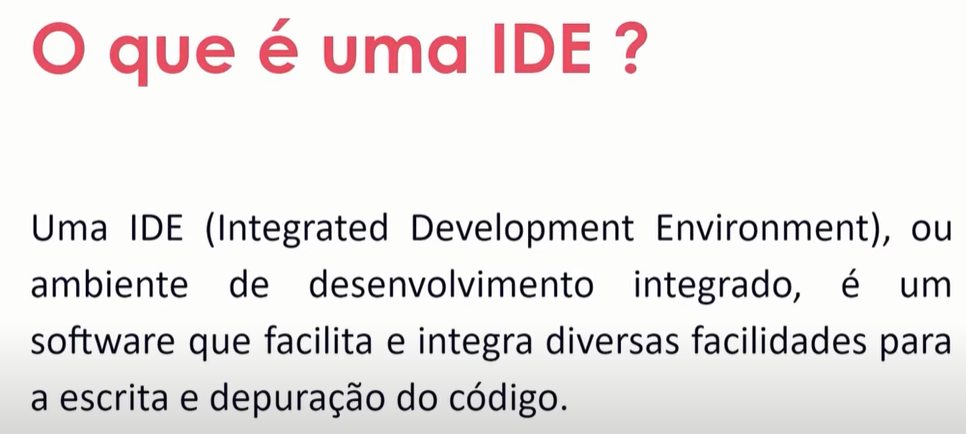


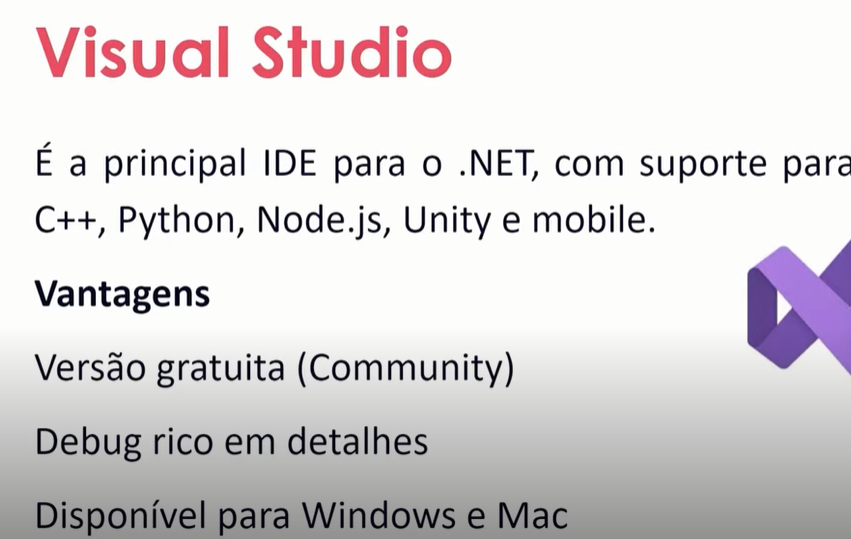


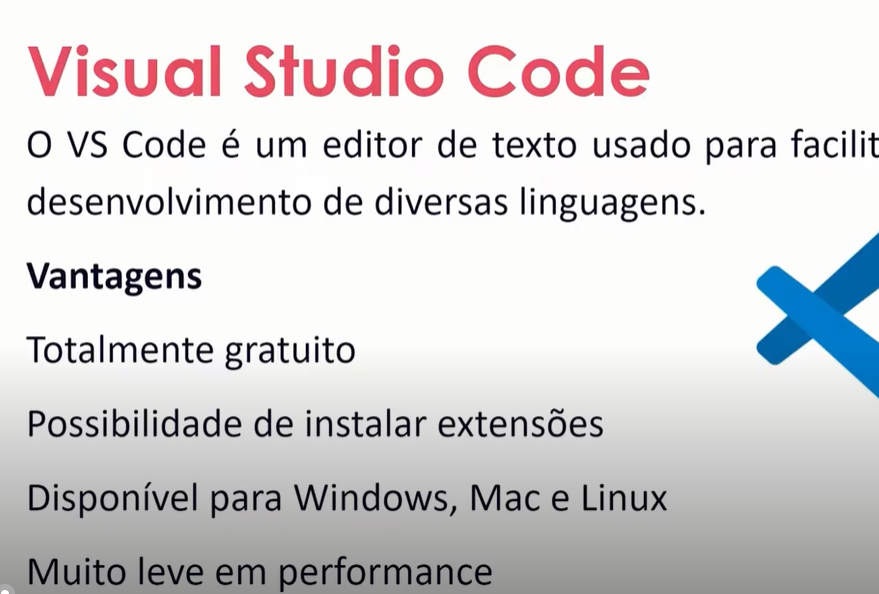




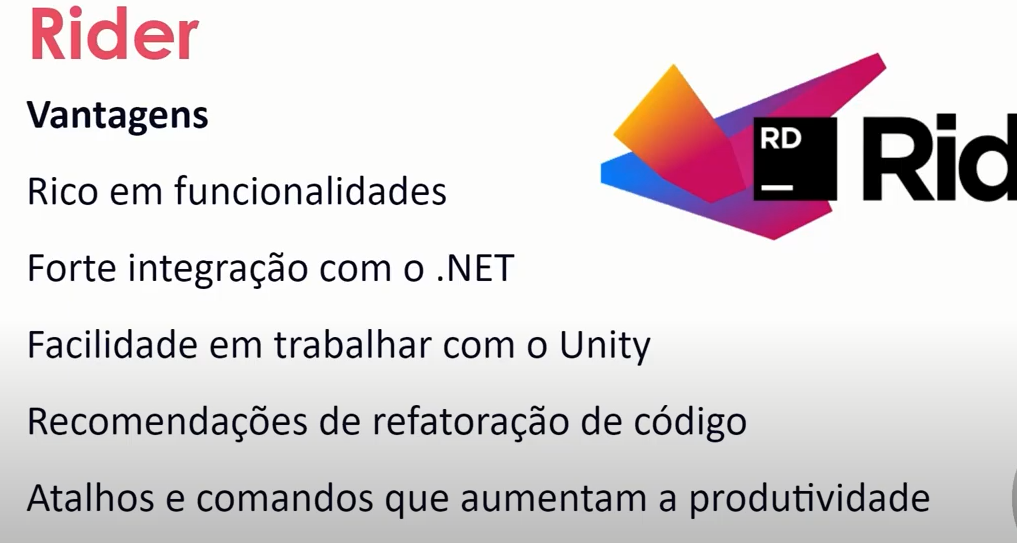






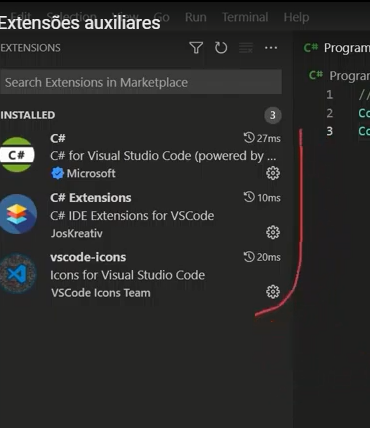


Sempre necessário instalar a extensão da linguagem que vamos utilizar.

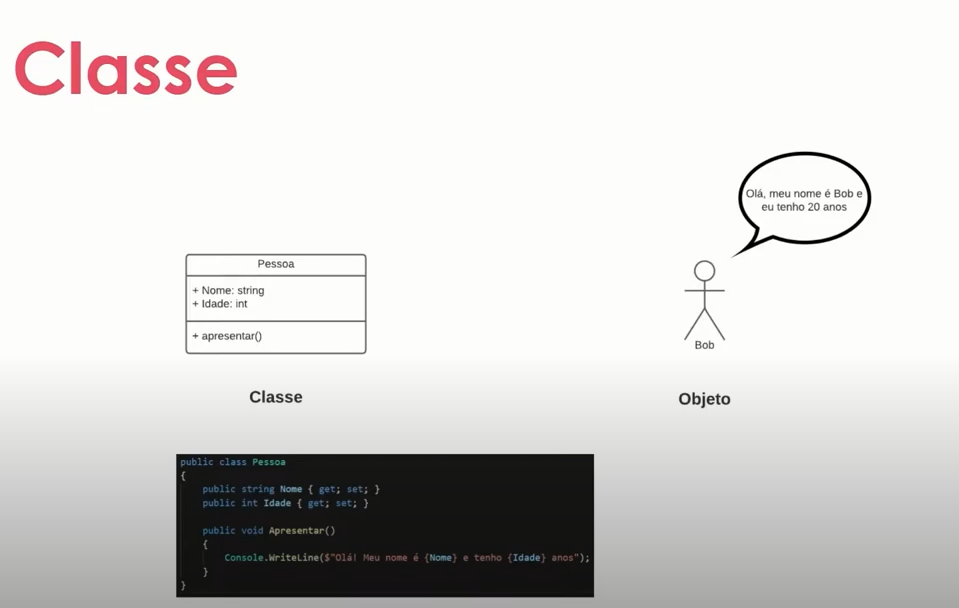




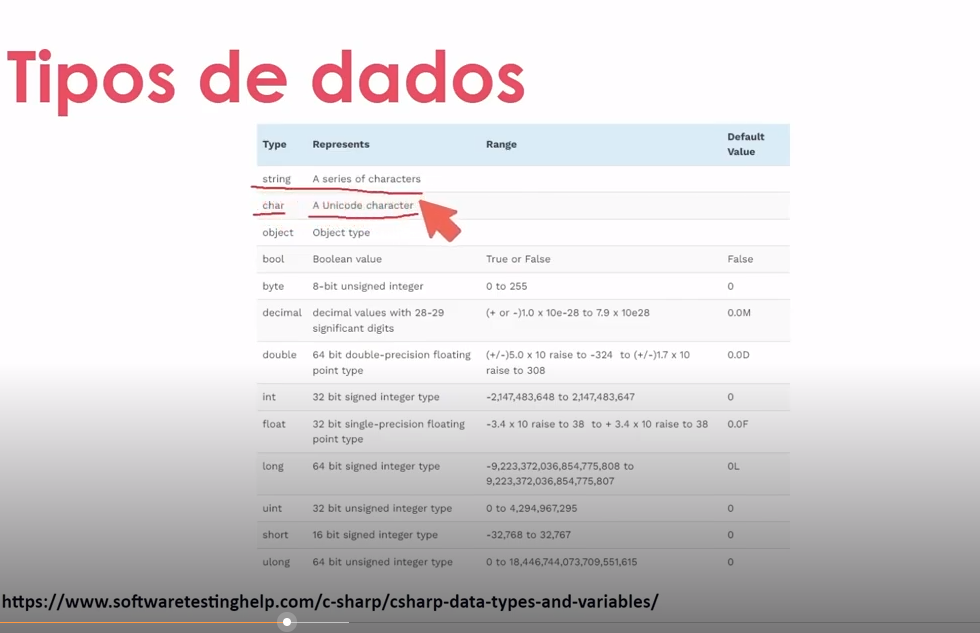
Extensões recomendadas para instalar:



ENTENDENDO A ESTRUTURA DE UM PROJETO:



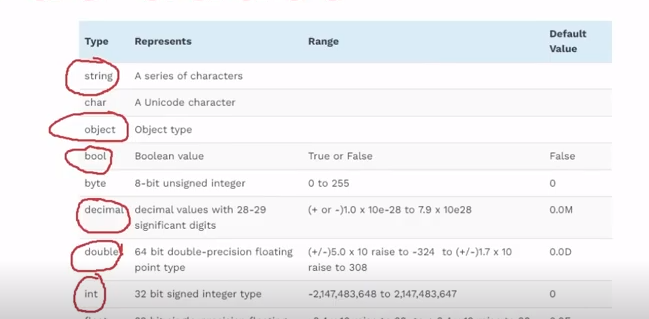
CLASSE – conceito representado do mundo real.



Mais utilizados - int e long – para representar números inteiros.

Decimal(valores monetários) , double, float

Mais utilizados geral:

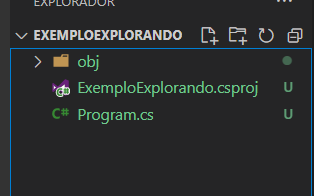


Criando do zero:

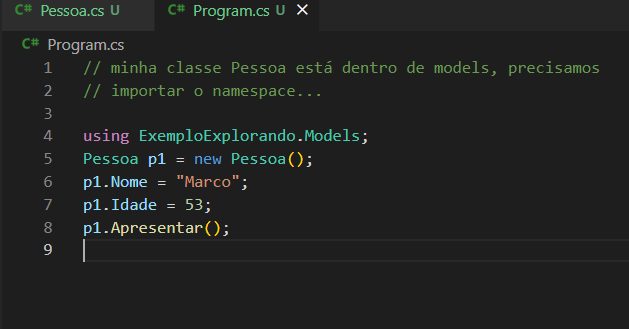
Criamos uma nova pasta no explorer..

No vscode - file – new pasta....

No terminal : dotnet new console  
Resultado:



Program.cs



Pessoa.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace ExemploExplorando.Models

{

    public class Pessoa

     // classe pessoa com 2 propriedades(possui get e set)

    {

        private string \_nome;

        public string Nome

        {

            get

            {

                return \_nome.ToUpper();

            }

            set

            {

                if (value == "")  //value eh o argumento que esta recebendo o Nome

                {

                    throw new ArgumentException("o nome não pode ser vazio");

                }

                \_nome = value;

            }

        }

        public int Idade { get; set; }

        public void Apresentar()

        {

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

        }

    }

}

Ou.. utilizando **body expression**. – se o código tiver uma única linha:

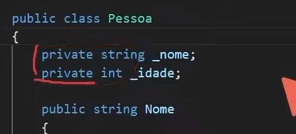
get

            {

                return \_nome.ToUpper();

            }

Get => \_nome.ToUpper();

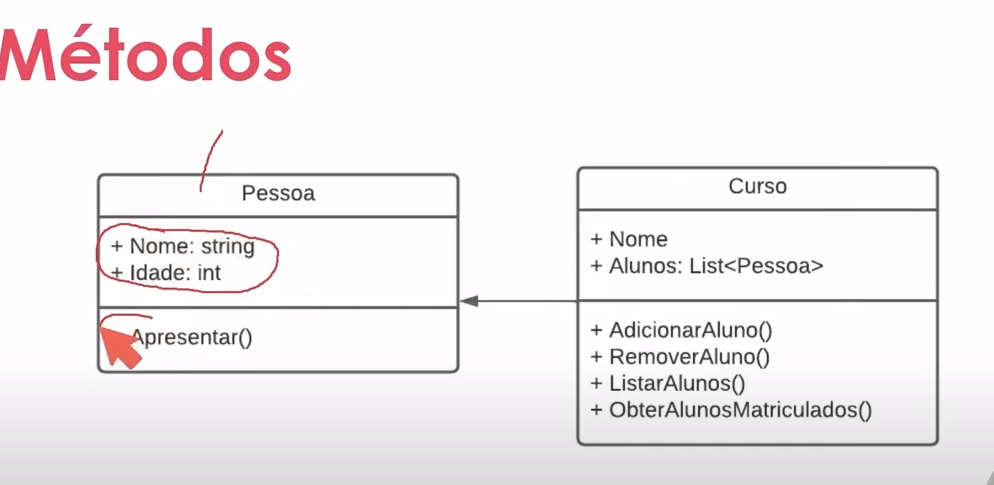


Private – só pode ser acessado de dentro da classe pessoa, e só pode ser modificado através da propriedade que criamos. E com isto, temos a garantia de que estamos fazendo as devidas validações tanto no set quanto no get.

\*\* encapsulamento – proteger para fazer validações antes de atribuir e já bloquear na hora.

PROPRIEDADE – VAI TER GET OU SET.. AI VEMOS A DIFERENÇA:   
public int Idade { get; set; }

Public –



**Nome da classe - Pessoa**

**Atributos /propriedades – Nome, Idade**

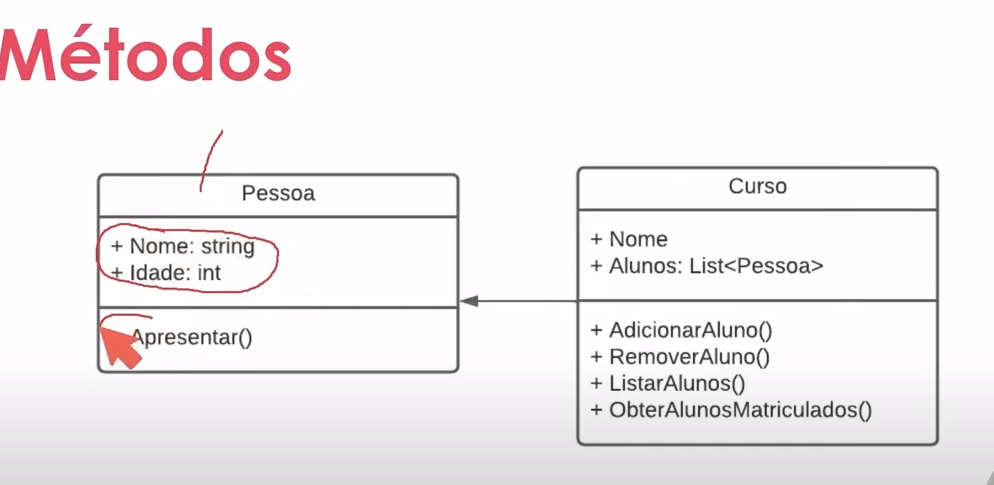
**Métodos – bloco de código que contém uma série de instruções. Ação que a classe vai fazer, o que ela faz, o que ela permite fazer, permite reaproveitar o código. – Apresentar**

**Classe – Curso**

**Atributos/propriedades – Alunos – List<Pessoa> ( é a nossa classe Pessoa).....**

**Métodos – AdicionarAluno(), RemoverAluno(),...**

**Para imprimir:**



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Program.cs**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**// minha classe Pessoa está dentro de models, precisamos importar o namespace...**

**using ExemploExplorando.Models;**

**//Pessoa p1 = new Pessoa();**

**//p1.Nome = "Marco";**

**//p1.Sobrenome = "Cini";**

**//p1.Idade = 53;**

**//p1.Apresentar();**

**Pessoa p1 = new Pessoa();**

**p1.Nome = "Marco";**

**p1.Sobrenome = "Cini";**

**Pessoa p2 = new Pessoa();**

**p2.Nome = "Julio";**

**p2.Sobrenome = "Cini";**

**Curso cursoDeIngles = new Curso();**

**cursoDeIngles.Nome = "Ingles";**

**cursoDeIngles.Alunos = new List<Pessoa>();**

**cursoDeIngles.AdicionarAluno(p1);**

**cursoDeIngles.AdicionarAluno(p2);**

**cursoDeIngles.ListarAlunos();**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Pessoa.cs**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Threading.Tasks;**

**namespace ExemploExplorando.Models**

**{**

**public class Pessoa**

**// classe pessoa com 2 propriedades(possui get e set)**

**{**

**private string \_nome;**

**private int \_idade;**

**public string Nome**

**{**

**get**

**{**

**return \_nome.ToUpper();**

**}**

**set**

**{**

**if (value == "") //value eh o argumento que esta recebendo o Nome**

**{**

**throw new ArgumentException("o nome não pode ser vazio");**

**}**

**\_nome = value;**

**}**

**}**

**public int Idade**

**{**

**get => \_idade;**

**// return \_idade - mesma coisa**

**set**

**{**

**if (value < 0)**

**{**

**throw new ArgumentException("A Idade nao pode ser menor que zero");**

**}**

**\_idade = value;**

**}**

**}**

**public string Sobrenome { get; set; }**

**public string NomeCompleto => $"{Nome} {Sobrenome}".ToUpper();**

**// juntar o nome e sobrenome.**

**// get - apenas retorna o valor, nao conseguimos atribuir valor nenhum.**

**public void Apresentar()**

**{**

**Console.WriteLine($"Nome: {NomeCompleto}, Idade: {Idade}");**

**}**

**}**

**}**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Curso.cs**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**using System;**

**using System.Collections.Generic;**

**using System.Linq;**

**using System.Threading.Tasks;**

**namespace ExemploExplorando.Models**

**{**

**public class Curso**

**{**

**public string Nome { get; set; }**

**// lista eh uma colecao do tipo Pessoa.**

**public List<Pessoa> Alunos { get; set; }**

**// void nao retorna nada. ou seja nao precisa do return.**

**public void AdicionarAluno(Pessoa aluno)**

**{**

**Alunos.Add(aluno);**

**}**

**// tipo int, necessita retornar um valor..**

**public int ObterQuantidadeDeAlunosMatriculados()**

**{**

**int quantidade = Alunos.Count;**

**return quantidade;**

**}**

**public void RemoverAluno(Pessoa aluno)**

**{**

**Alunos.Remove(aluno);**

**}**

**// ou**

**// public bool RemoverAluno(Pessoa aluno)**

**// {**

**// Alunos.Remove(aluno);**

**// return true;**

**// }**

**public void ListarAlunos() {**

**Console.WriteLine($"Alunos do curso de: {Nome}");**

**foreach (Pessoa aluno in Alunos)**

**{**

**Console.WriteLine(aluno.NomeCompleto);**

**}**

**}**

**}**

**}**

**CONSTRUTORES**

**MANIPULANDO VALORES ----**

**- CONCATENAÇÃO (OU INTERPOLAÇÃO) DE STRINGS**

$"Alunos do curso de: {Nome}"

public void ListarAlunos()

        {

            Console.WriteLine($"Alunos do curso de: {Nome}");

            for (int count = 0; count < Alunos.Count; count ++)

            {

                Console.WriteLine($" nº {count+1} - {Alunos[count].NomeCompleto}");

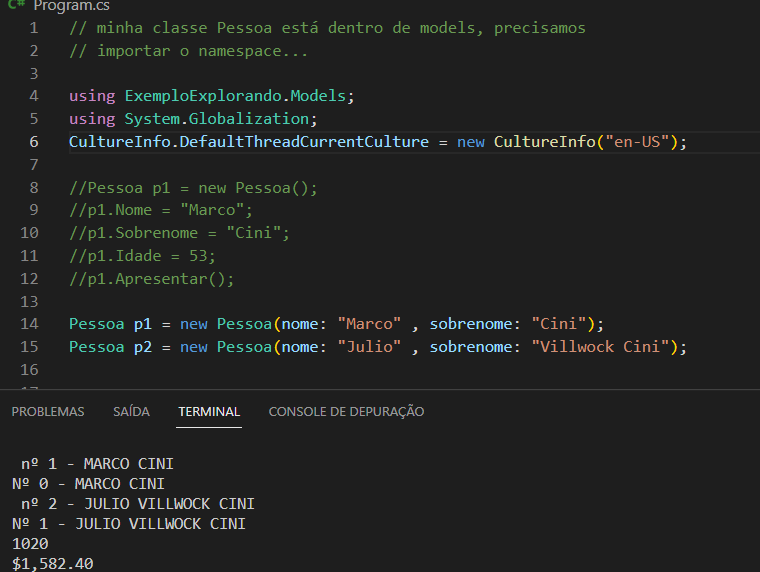
                // ou

                string texto = "Nº " + count + " - " + Alunos[count].NomeCompleto;

                Console.WriteLine(texto);

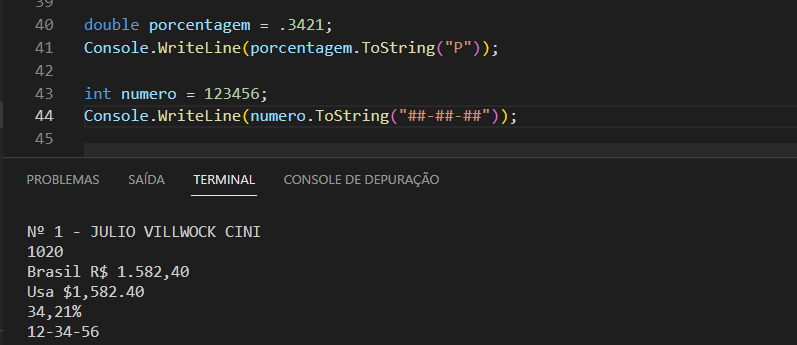
**CONCATENAÇÃO DE VALORES / formatando valores monetários:**

**Ou pt-BR**



**PORCENTAGEM:**



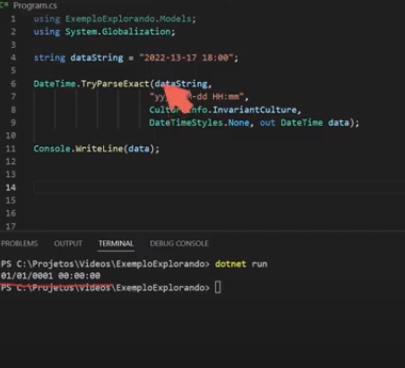


**FORMATANDO O TIPO DATETIME**



// HH = 24h   HH - 12H

**INTRODUCAO EXCEPTIONS:**



**Data passada incorreta, exibiu um valor padrão mas não encerrou o programa.**

// parse - converter de string para date

DateTime data1 = DateTime.Parse("20/04/2023 14:39");

Console.WriteLine(data1);

string dataString = "20-04-2023 18:00";

bool sucesso = DateTime.TryParseExact(dataString,

"dd-MM-yyyy HH:mm",

CultureInfo.InvariantCulture,

DateTimeStyles.None, out DateTime data3);

if (sucesso)

{

    Console.WriteLine($"Conversão com sucesso! Data: {data3}");

}

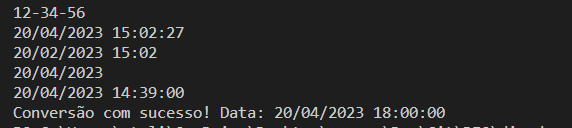
else

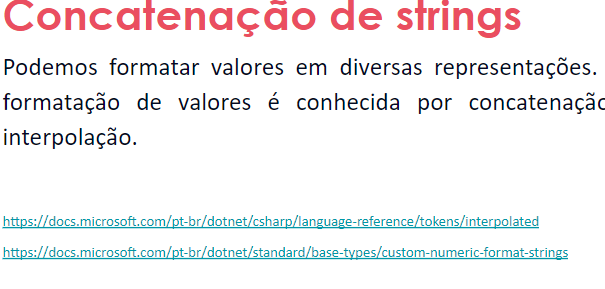
{

    Console.WriteLine($"{dataString} não é uma data válida");

}

**Resultado:**





**Try - catch – finally**

//tratamento de excecoes, trata e segue a execucao.

try

{

    string[] linhas = File.ReadAllLines("Models/Arquivos/arquivo\_Leitura.txt");

    foreach(string linha in linhas)

    {

    Console.WriteLine(linha);

    }

}

catch(FileNotFoundException ex)

{

    Console.WriteLine("arquivo nao encontrado");

}

catch(DirectoryNotFoundException ex)

{

    Console.WriteLine("caminho da pasta nao encontrado");

}

catch(Exception ex)

{

    Console.WriteLine("Ocorreu uma exceção genérica. Mensage de erro:" + ex.Message);

}finally

{

}

Console.WriteLine("Chegou até aqui sem erros");

{

    public class ExemploExcecao

    {

        public void Metodo1()

        {

            Metodo2();

        }

        public void Metodo2()

        {

            Metodo3();

        }

        public void Metodo3()

        {

            Metodo4();

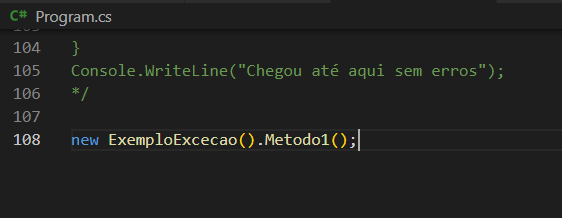
        }

        public void Metodo4()

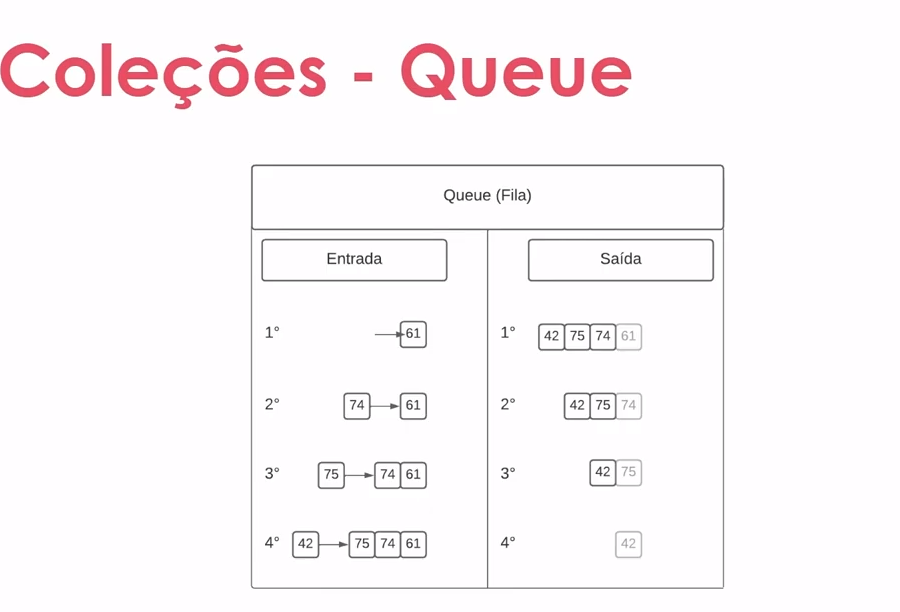
        {

            throw new Exception("Ocorreu uma excecao");

        }



**QUEUE – FILAS**



**FIFO – first in first out**

Queue<int> fila = new Queue<int>();

fila.Enqueue(2);

fila.Enqueue(4);

fila.Enqueue(6);

fila.Enqueue(8);

foreach(int item in fila)

{

    Console.WriteLine(item);

}

// remover um elemento da fila -sempre remove o primeiro elemento que entrou.

// regra da fila.

 Console.WriteLine($"Removendo o elemento: {fila.Dequeue()}");

 foreach(int item in fila)

{

    Console.WriteLine(item);

}

**PILHA - STACK**

**Lifo – last in first out – ultimo que entrou é o primeiro a sair.**

// PILHA  - REMOVE O ELEMENTO DO TOPO - FIFO

Stack<int> pilha = new Stack<int>();

pilha.Push(4);

pilha.Push(6);

pilha.Push(8);

pilha.Push(10);

foreach(int item1 in pilha)

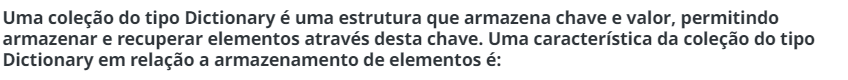
{

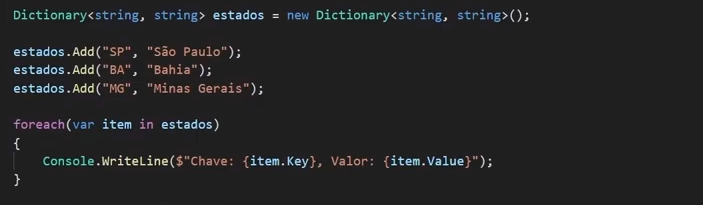
    Console.WriteLine(item1);

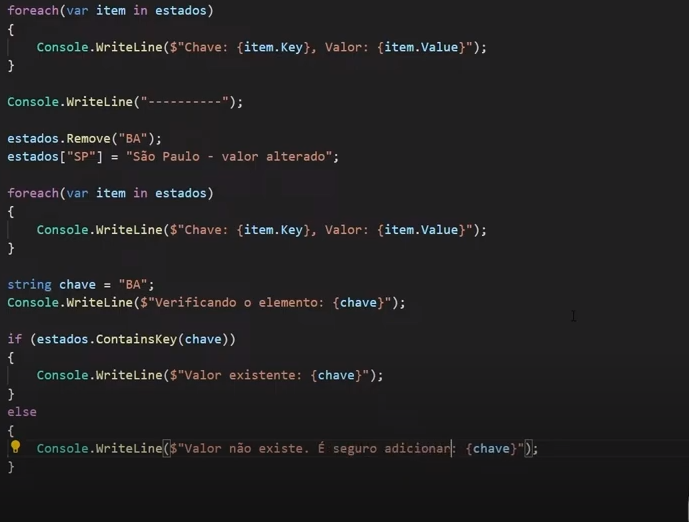
}

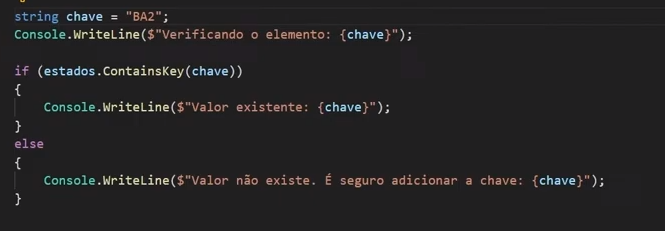
Console.WriteLine($"removendo elemento do topo:{pilha.Pop()}");

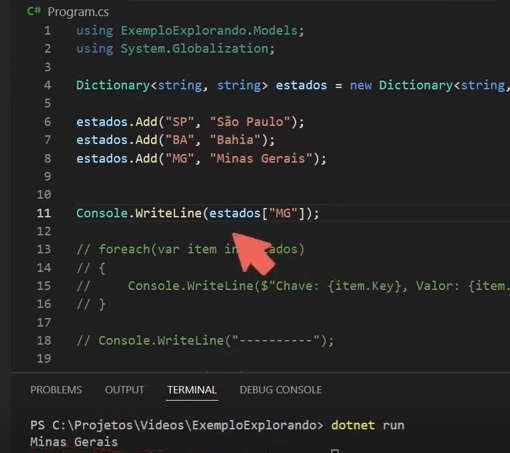
**DICTIONARY - GARANTE QUE ELEMENTOS SEJAM UNICOS DE ACORDO COM O PRIMEIRO ELEMENTO – QUE É ÚNICA.**

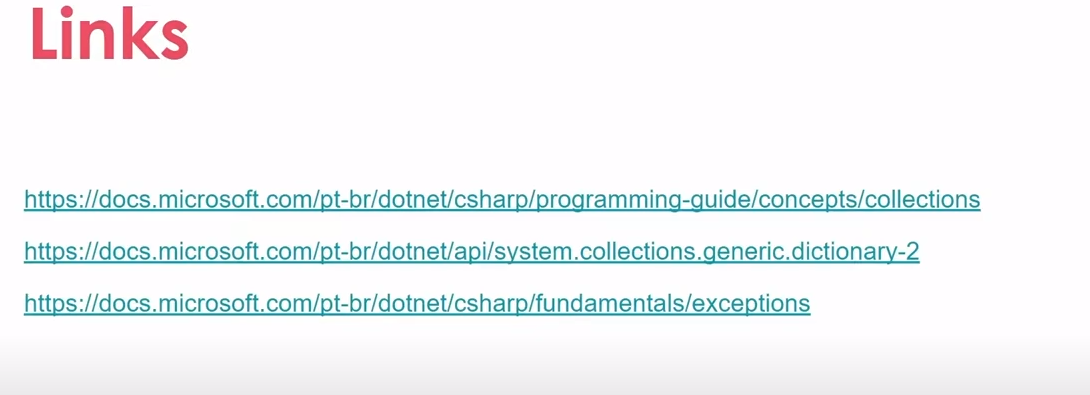




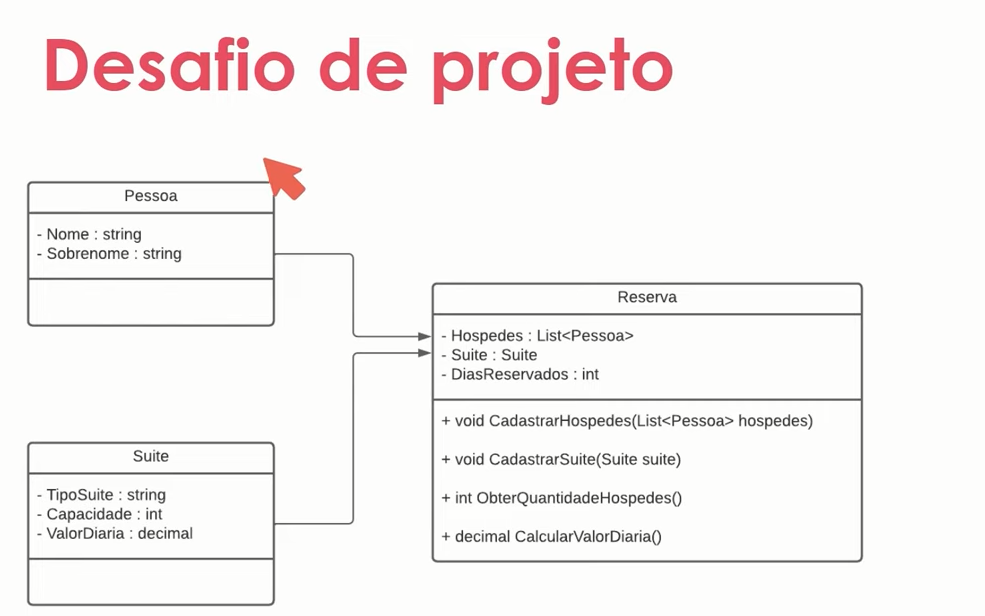








**DESAFIO DE PROJETO**



**Anotações gerais:**

**3 classes – Pessoa, Reserva, Suite.**

**Reserva – propriedades:  
- hospedes(lista de pessoas – 1 ou mais pessoas  
- suíte  
- diasReservados  
acoes da classe reserva:**

**Calcular Valor diária :**

**Se reservar a suíte num período igual ou maior que 10 dias, você receberá 10% de desconto e este valor deve retornar do CalcularValorDiaria()**

**- validar a capacidade da suíte – reservar para o nro de pessoas que tem na capacidade.**

**\*\*código que vamos precisar alterar é na classe reserva**

