***Estrutura de um Projeto***

1. **Arquivo Program.cs:** Este é o **ponto de entrada** do programa, ou seja, é onde a execução começa. É o arquivo que contém o método Main, o qual o compilador identifica como o ponto de partida.
2. **Arquivos.cs:** As **classes** são geralmente definidas em arquivos.cs e contêm o **código** que descreve o comportamento e os dados dos objetos. Você pode ter várias classes em um projeto, cada uma em um arquivo.cs separado ou agrupadas.
3. **Método Main - “static void Main(string[] args)”:** é o ponto de partida de um projeto. Todo projeto de console em C# deve ter pelo menos um arquivo contendo este método e só é possível existir 1 método desse por projeto. Ele pode estar no Program.cs ou em qualquer outro arquivo, desde que seja o **ponto de entrada** do programa.
4. **Arquivo de Projeto (.csproj):** Este arquivo é usado para definir configurações do projeto. Ele contém informações como: Framework que o projeto usa; Dependênciase pacotes NuGet que o projeto precisa; Configurações de build e compilação; Referências a outros projetos ou bibliotecas.
5. ***Using:*** No início dos arquivos .cs, é comum encontrar declarações using. Elas são usadas para **importar namespaces** que fornecem funcionalidades adicionais ao código, como classes e métodos predefinidos. Por exemplo:

- using System: Importa classes básicas como Console para escrever saídas no terminal.

- using System.Collections.Generic: Importa estruturas de dados como listas (List), dicionários (Dictionary)

1. ***Namespace:*** é o local onde temos o projeto, geralmente é o nome do projeto;
2. ***Bin e Obj:***

- A pasta \*\*bin\*\* é onde os arquivos compilados são armazenados. Aqui você encontra os executáveis ou bibliotecas (.dll) geradas.

- A pasta \*\*obj\*\* é usada para armazenar arquivos temporários de compilação. Ela contém informações que o compilador usa para gerar o produto final.

1. ***Pasta Properties:*** a pasta Properties normalmente contém o arquivo launchSettings.json, que define as configurações de depuração do projeto.
2. ***Dependências (NuGet):*** Projetos C# podem usar pacotes NuGet, que são bibliotecas externas que adicionam funcionalidades ao projeto. Esses pacotes são gerenciados pelo NuGet Package Manager do Visual Studio.

***Classe***

* **O que é:** Uma classe é um molde ou blueprint para criar objetos (instâncias) que encapsulam dados e comportamentos relacionados. Em outras palavras, uma classe define um novo tipo de dado que pode conter atributos (variáveis) e métodos (funções).
* **Definição de uma Classe:** A definição básica de uma classe em C# é feita utilizando a palavra-chave class, seguida pelo nome da classe. O nome da classe deve começar com uma letra maiúscula e, por convenção, usa-se o PascalCase.
* **Componentes da classe:**

- Atributos;

- Construtores;

- Métodos;

***Atributo***

* **O que é:** São variáveis associadas a uma classe. Eles são usados para armazenar o estado ou características de um objeto;
* Cada objeto (ou instância) da classe terá seus próprios valores para esses atributos;
* **Método Construtor:** Para os atributos serem inicializados precisamos do método **construtor,** que é um método especial que é chamado automaticamente quando um novo objeto da classe Livro é criado. Ele inicializa os atributos da classe com os valores passados como parâmetros, ele obriga que seja passado um valor para os atributos no momento da instanciação;
* **Construtor Padrão:** quando não definimos um construtor, o compilador cria um automaticamente.

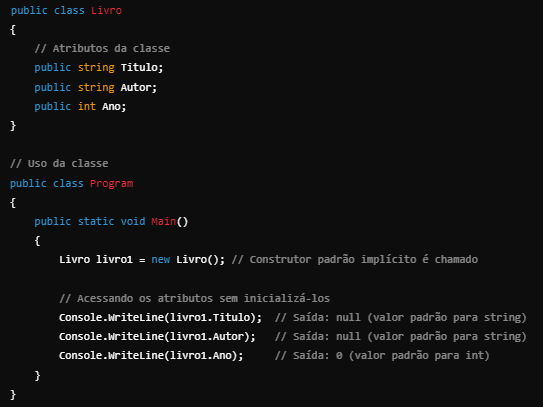
- Não recebe parâmetros.

- Inicializa os atributos da classe com seus valores padrão:

. Para tipos numéricos (int, float, double, etc.), o valor padrão é 0;

. Para tipos bool, o valor padrão é false.

. Para tipos de referência (como string ou objetos), o valor padrão é null.

 - Exemplo com construtor padrão:

* **Construtor Personalizado**: Você pode deixar o construtor implícito funcionar, mas, em muitos casos, é uma boa prática usar um **construtor personalizado** quando você deseja inicializar os atributos da classe com valores específicos logo no momento em que o objeto é criado. Isso facilita a inicialização e evita que os atributos fiquem com valores indefinidos ou inadequados.

- **Vantagens:**

**. Inicialização:** Garante que os atributos da classe estejam corretamente inicializados no momento da criação do objeto.

**. Facilidade:** Simplifica o código, pois permite que você defina valores diretamente no construtor, sem precisar fazer atribuições separadamente.

**. Validação:** Você pode adicionar validações no construtor para garantir que os dados fornecidos são válidos (por exemplo, evitar que o ano seja negativo).

- Exemplo do construtor personalizado:



* **Construtores:** Ambas as abordagens são válidas, mas o uso de um construtor personalizado ajuda a evitar que os atributos permaneçam com valores padrão indesejados, como null ou 0, e melhora a clareza do código.
* **Outros tópicos importantes:**

1. Como modificar o acesso aos atributos (public, private, protected).
2. Como usar atributos estáticos para compartilhar valores entre instâncias.

- Quando um atributo é estático, ele pertence à **classe** em si, e não a uma instância específica da classe. Isso significa que todas as instâncias da classe compartilham o mesmo valor para esse atributo.

- Exemplo:



1. A diferença entre atributos readonly e const para controlar imutabilidade.



1. A importância das propriedades automáticas para encapsulamento.
2. Como inicializar atributos diretamente.
3. Como os atributos interagem com herança.

***Métodos***

* São as funções da nossa classe;
* Eles encapsulam um bloco de código que executa uma tarefa específica e podem ser reutilizados ao longo do programa;
* **Componentes do método:**

- **Modificador de Acesso**: Define a visibilidade do método, como public, private, protected.

**- Tipo de Retorno**: Indica o tipo de dado que o método vai devolver. Pode ser um tipo primitivo (int, string, etc.), um tipo de objeto ou void (sem retorno).

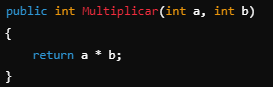
- **Nome do Método**: O nome que será usado para chamar o método. Deve ser significativo e seguir a convenção PascalCase.

- **Parâmetros**: Entre parênteses, os parâmetros são dados que podem ser passados para o método.

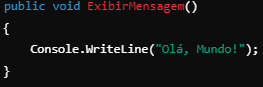
- **Retorno**: Para métodos que retornam algo, a palavra-chave return é usada para devolver o valor.

* **Tipos de métodos:**

1. Métodos com Retorno: Esses métodos devolvem um valor do tipo especificado.



1. Métodos sem retorno: Esses métodos não devolvem nenhum valor. Eles executam uma ação e terminam.



1. Métodos Estáticos: Métodos marcados como static pertencem à **classe** em si, e não a uma instância dela. Eles podem ser chamados sem criar um objeto da classe.
2. Métodos de Instância: São métodos que dependem de uma **instância** da classe para serem chamados. Eles podem acessar os atributos e modificar o estado de um objeto.
3. Métodos de Extensão: Permitem "adicionar" novos métodos a uma classe existente sem modificar seu código original.

* Resumo

- Métodos são blocos de código que executam uma tarefa específica.

- Eles podem ser chamados em uma instância de uma classe ou diretamente na classe se forem estáticos.

- Métodos podem retornar valores, aceitar parâmetros, e serem sobrecarga ou sobrescritos em subclasses.

- Conhecer os diferentes tipos de métodos (estáticos, instância, sobrecarga, abstratos, virtuais) permite escrever código mais flexível e reutilizável.

***Objeto/Instância***

* Um **objeto** é uma **instância** de uma classe;
* Quando você define uma classe, está criando um "molde" ou "modelo" para objetos. A partir desse molde, você pode criar quantos objetos quiser.
* Cada objeto tem seus próprios valores de atributos, mas compartilha o mesmo comportamento (métodos) definido na classe.
* **Criar um objeto:** Para criar um objeto, usamos a palavra-chave **“new”,** que chama o **construtor** da classe. O **construtor** inicializa os atributos e retorna uma nova instância da classe.
* Cada instância de uma classe ocupa um espaço diferente na memória. Mesmo que dois objetos tenham os mesmos valores de atributos, eles ainda são considerados objetos distintos.
* **Componentes de um Objeto:**

- **Atributos**: Cada instância de uma classe tem seus próprios valores para os atributos, mesmo que esses atributos pertençam à mesma classe.

- **Métodos**: Os métodos definidos na classe são compartilhados por todas as instâncias, mas podem alterar os atributos de cada objeto individualmente.

* **Resumo:**

- Um **objeto** é uma instância de uma classe, que contém seus próprios valores de atributos e compartilha o comportamento definido pelos métodos da classe.

- A criação de um objeto se dá pelo uso do operador new, que chama o **construtor** da classe.

- Objetos são passados por referência, permitindo que métodos modifiquem o estado do objeto original.