**Linguagem escolhida:** C#=

**Padrão de nomenclatura:**

Nomes:

1. Abreviações somente quando necessário ou caso o nome da variável fique muito extenso, mais vale um nome longo do que um incompreensível.
2. Não usar proposições para simplificar a padronização e diminuir o tamanho dos nomes, “notadoAluno\_d” -> “notaAluno\_d”
3. Nomes curtos são preferíveis.
4. Não usar caracteres especiais ou acentuação.

Exemplo usuarioTruck

**Componentes:**

|  |  |  |
| --- | --- | --- |
| **Form** | frm | frmFileOpen |
| **UserControl** | usc | uscLogin |
| **TextBox** | txt | txtGetText |
| **Label** | lbl | lblTitle |
| **ComboBox** | cbo | cboCity |
| **Image** | img | imgIcon |
| **Picturebox** | pic | picHeader |
| **Grid** | grd | grdPrices |
| **GridView** | grv | grvClients |
| **Radio Button** | rbt | rbtRequerid |
| **LinkLabel** | lkl | lklSite |
| **CheckBox** | chk | chkSex |
| **ListBox** | lst | lstPrices |
| **Button** | btn | btnClientSave |
| **TreeView** | trv | trvMenu |
| **DropDownList** | Ddl | ddlVendedores |

**Métodos / Rotinas**:

**Denominação:**

1. Prefixo:
   1. Todos os métodos devem começar com a primeira letra maiúscula, sendo privados ou públicos.
2. Sufixo:
   1. Caso eles retornem valor, os mesmos devem ser seguidos com a primeira letra do tipo da variável de retorno, “\_d” para double por exemplo.

**Inicio e finalização:**

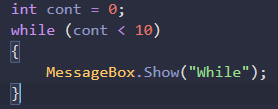
1. Inicio:
   1. Sempre devem iniciar com um comentário, fazendo uma descrição resumida da finalidade do método, indicando o porquê dele, no fim da descrição, identificar o nome do(s) autor(es) com o sobrenome abreviado com as primeiras letras.
2. Fim:
   1. Fazer um comentário fim com o nome do método, isso ajuda em métodos muito extensos.

**Estrutura de repetição while:**

1. Regras do while:

a. nunca deve ser criado uma variável dentro de um while.

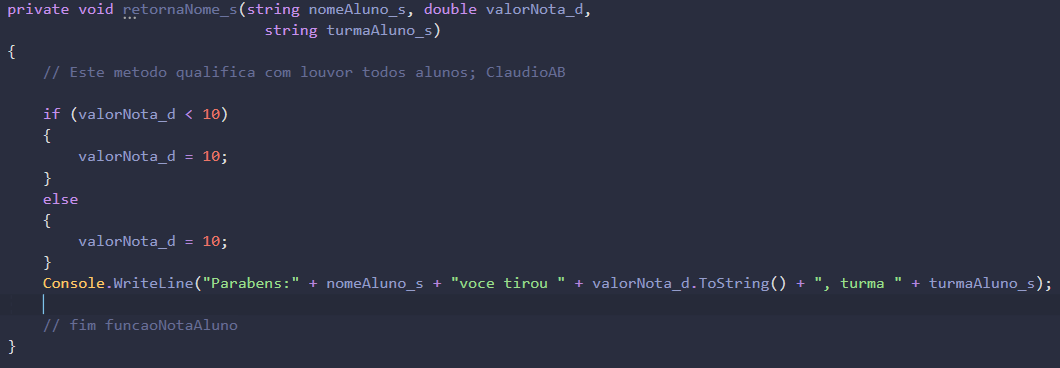
**Exemplo:**

****

**Indentação Métodos:**

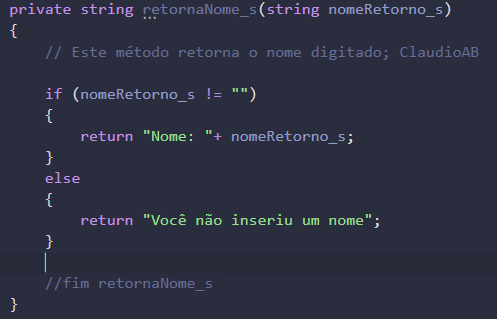
1. **Corpo do método:**
   1. Após a declaração do método, o início de abertura do método deve começar na próxima linha.
   2. Após o caractere de inicialização “{“, o início do método deve começar na próxima linha com uma tabulação de espaçamento, indicando o inicio do código em si.
2. **Rotinas condicionais ou de repetição**
   1. Caso a declaração da condição seja muito extensa, pode ser iniciada uma nova linha após o caractere de separação, priorizando deixar as condições/parâmetros no mesmo nível de indentação para facilitar a compreensão do código.
   2. Para rotinas ( condicionais, repetição e outros), a regra é a mesma que para o corpo do método, a condição deve ser seguida de uma nova linha com “{“ para iniciar a rotina.
   3. Após o caractere de inicialização da rotina “{”, o código deve começar com um tab de espaçamento.
   4. Para finalização, o caractere “}” deve estar na mesma indentação do caractere de inicialização

**Exemplo:**



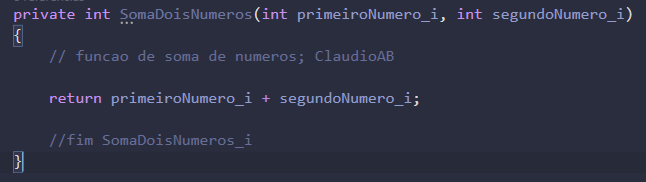
**Indentação padrão:**

1. Inicio:
   1. Sempre respeitando a hierarquia do código, todo código que estiver dentro de uma rotina, cláusula, método ou qualquer condição que tenha o caractere de início de código “{” deve ser iniciado em uma linha após o caractere inicializador seguido de um tab de espaçamento.
   2. O caractere de inicialização sempre deve proceder as declarações em uma linha
2. Fim:
   1. Caractere de finalização sempre deve estar no mesmo nível de identação do caractere de inicialização.
3. Declaração de parâmetros e condições, quando houver mais de um ou uma, o caractere de separação deve ser precedido por um espaço.



1. Blocos lógicos devem ser procedidos de uma nova linha para facilitar a compreensão do código.

Exemplo:

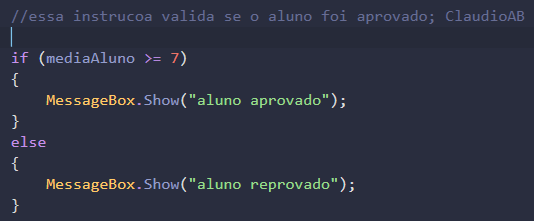


**Instrução** **logica if:**

1. Corpo da instrução

a. o comentário deve ser descrito antes da instrução.

b. não deve haver espaço entre os colchetes e o código que estão dentro da instrução if.

**Exemplo: **

**Denominação de variáveis:**

1. Prefixo:
   1. Toda variável deve iniciar com a letra minúscula caso seja privada ou seja de uso interno do método, caso seja pública deve iniciar com a primeira letra maiúscula, como nos métodos.
2. Sufixo:
   1. Devem terminar com o a primeira letra do tipo de variável usada seguido de underline, “\_s” para string no ocaso.
3. Nome:
   1. Quando a variável tiver mais de uma palavra a segunda letra deve sempre começar com a letra maiúscula.

**Exemplo:**

private int valorContagem\_i;

public string NomeAluno\_s ;

**Orientação a objeto:**

**Estrutura da Classe:**

1. Encapsulamento:

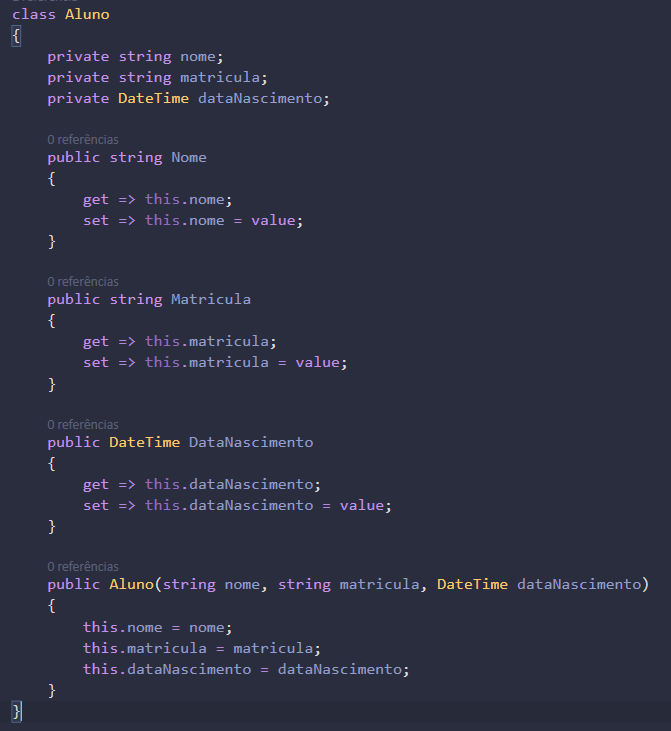
a. toda variável dentro da classe será privada, partindo do pressuposto que a restrição deve fluir do mais restrito ao menos restrito.

b. as variáveis privadas com visibilidade publica devem ter metodos get e set especificando a visibilidade das mesmas, variáveis privadas de uso interno da classe não necessitam dos get e set.

2. Construtores:

a. O construtor deve sempre ser declarado, mesmo que não haja argumentos

b. Quando o construtor da classe tiver argumento, não há necessidade de declarar um construtor sem argumentos.

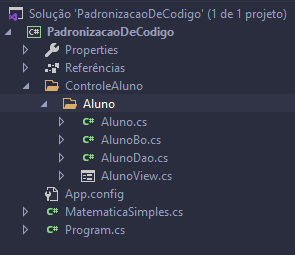
**Exemplo:**  

**Estrutura do projeto:**

1. Encapsulamento das pastas:

a. O projeto precisa ter algumas pastas para melhor organização, inicialmente precisamos criar uma pasta controle seguido do nome da entidade, por exemplo ControleAluno, em seguida criaremos uma pasta com o nome da entidade, como por exemplo Aluno, apos isso criaremos todas as classes, Aluno, AlunoBo, AlunoDao e o form AlunoView.

**Exemplo:**

****