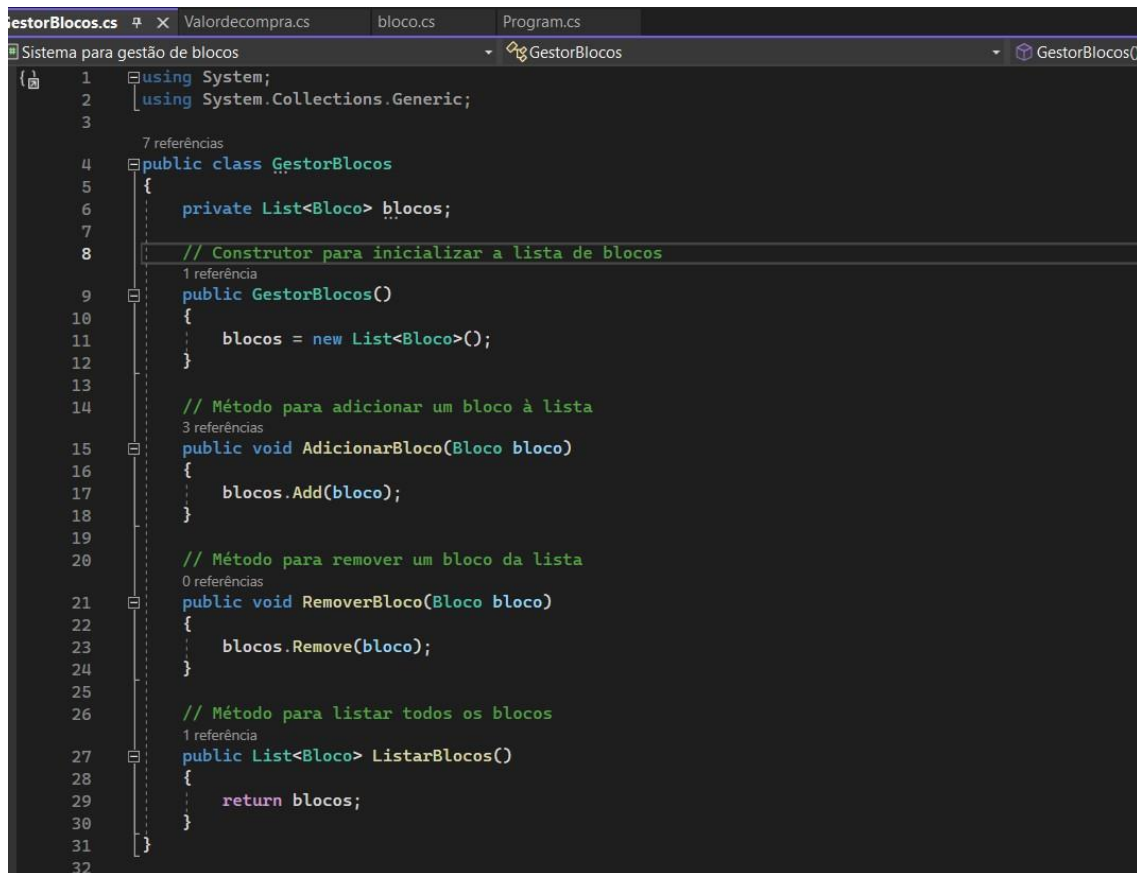


Alunos: Guilherme Sabino de Oliveira, Arthur Costalonga Colli

Link do git: <https://github.com/Guilherme1593/Sistema-para-gest-o-de-blocos.git>



```
1 using System;
2 using System.Collections.Generic;
3
4 public class GestorBlocos
5 {
6     private List<Bloco> blocos;
7
8     // Construtor para inicializar a lista de blocos
9     public GestorBlocos()
10     {
11         blocos = new List<Bloco>();
12     }
13
14     // Método para adicionar um bloco à lista
15     public void AdicionarBloco(Bloco bloco)
16     {
17         blocos.Add(bloco);
18     }
19
20     // Método para remover um bloco da lista
21     public void RemoverBloco(Bloco bloco)
22     {
23         blocos.Remove(bloco);
24     }
25
26     // Método para listar todos os blocos
27     public List<Bloco> ListarBlocos()
28     {
29         return blocos;
30     }
31 }
32
```

### GestorBlocos

A classe GestorBlocos é responsável por gerenciar uma coleção de blocos. Ela possui uma lista privada de objetos Bloco e métodos para adicionar, remover e listar blocos.

**blocos:** Lista privada que armazena os objetos Bloco.

**AdicionarBloco (Bloco bloco):** Adiciona um bloco à lista.

**RemoverBloco(Bloco bloco):** Remove um bloco da lista.

**ListarBlocos():** Retorna a lista de blocos atual.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Sistema_para_gestão_de_blocos
8 {
9     // Referências
10     internal class Valordecompra
11     {
12         // Referências
13         static void CadastrarBloco(GestorBlocos gestor)
14         {
15             Console.WriteLine("Número do Bloco: ");
16             string numero = Console.ReadLine();
17
18             double valorCompra, valorVenda;
19
20             do
21             {
22                 Console.WriteLine("Valor de Compra: ");
23                 } while (!double.TryParse(Console.ReadLine(), out valorCompra));
24
25             do
26             {
27                 Console.WriteLine("Valor de Venda: ");
28                 } while (!double.TryParse(Console.ReadLine(), out valorVenda));
29
30             Console.WriteLine("Nome do Bloco: ");
31             string nome = Console.ReadLine();
32
33             Console.WriteLine("Comprimento (mm): ");
34             double comprimento = double.Parse(Console.ReadLine());
35
36             Console.WriteLine("Largura (mm): ");
37             double largura = double.Parse(Console.ReadLine());
38
39             Console.WriteLine("Espessura (mm): ");
40             double espessura = double.Parse(Console.ReadLine());
41
42             Console.WriteLine("Material: ");
43             string material = Console.ReadLine();
44
45             // Cria um objeto Bloco com os dados fornecidos pelo usuário
46             Bloco bloco = new Bloco(nome, comprimento, largura, espessura, material);
47
48             // Define as propriedades de número, valor de compra e valor de venda
49             bloco.Numero = numero;
50             bloco.ValorCompra = valorCompra;
51             bloco.ValorVenda = valorVenda;
52
53             // Adiciona o bloco à lista gerenciada pelo GestorBlocos
54             gestor.AdicionarBloco(bloco);
55
56             Console.WriteLine("Bloco cadastrado com sucesso!");
57         }
58     }
59 }
```

A função ValorDeCompra é uma função que registra o valor de compra de um bloco específico. No contexto do programa, ela deve atualizar a propriedade ValorCompra de um objeto Bloco com o valor de compra informado pelo usuário.

Nesta função:

bloco: É o objeto Bloco ao qual você deseja atribuir o valor de compra.

valorCompra: É o valor de compra que você deseja registrar para o bloco.

Quando você chama essa função e passa um objeto Bloco e um valor de compra, ela atualiza a propriedade ValorCompra desse bloco com o valor fornecido.

```
Sistema para gestão de blocos Bloco
4 {
5     // Propriedades para armazenar informações sobre o bloco
6     public string Nome { get; set; }
7     public double Comprimento { get; set; }
8     public double Largura { get; set; }
9     public double Espessura { get; set; }
10    public string Material { get; set; }
11    public string? Numero { get; internal set; }
12    public double ValorCompra { get; internal set; }
13    public double ValorVenda { get; internal set; }
14
15    // Construtor para criar um objeto Bloco com dados iniciais
16    public Bloco(string nome, double comprimento, double largura, double espessura, string material)
17    {
18        Nome = nome;
19        Comprimento = comprimento;
20        Largura = largura;
21        Espessura = espessura;
22        Material = material;
23    }
24
25    // Método para calcular o volume do bloco
26    public double CalcularVolume()
27    {
28        return Comprimento * Largura * Espessura;
29    }
30
31    // Sobrescrevendo o método ToString para exibir informações do bloco
32    public override string ToString()
33    {
34        double lucro = CalcularLucro();
35        return $"{Nome} - Material: {Material}, Dimensões: {Comprimento}x{Largura}x{Espessura} mm, Lucro de Venda: {lucro:C}";
36    }
37
38    // Dentro da classe Bloco
39    public double CalcularLucro()
40    {
41        // Calcula o lucro subtraindo o custo de compra do valor de venda
42        return ValorVenda - ValorCompra;
43    }
44 }
45
46
47
```

## Bloco

A classe Bloco representa um bloco de mármore ou granito. Ela contém propriedades para armazenar informações sobre o bloco, como nome, comprimento, largura, espessura, material, número, valor de compra e valor de venda. Além disso, a classe possui métodos para calcular o volume do bloco e o lucro de venda.

Nome: Armazena o nome do bloco.

Comprimento, Largura, Espessura: Armazenam as dimensões do bloco

Material: Armazena o material do bloco.

Numero: Armazena o número do bloco.

ValorCompra: Armazena o valor de compra do bloco.

ValorVenda: Armazena o valor de venda do bloco.

CalcularVolume(): Calcula o volume do bloco com base nas dimensões.

CalcularLucro(): Calcula o lucro de venda subtraindo o valor de compra do valor de venda.

ToString(): Sobrescrito para exibir informações formatadas do bloco, incluindo o lucro de venda.

```
Sistema para gestão de blocos
1 using System;
2
3 namespace Sistema_para_gestão_de_blocos
4 {
5     0 referências
6     class Program
7     {
8         private static readonly object meuBloco;
9
10        0 referências
11        static void Main(string[] args)
12        {
13            GestorBlocos gestor = new GestorBlocos();
14
15            while (true)
16            {
17                Console.WriteLine("Menu:");
18                Console.WriteLine("1. Cadastrar Bloco");
19                Console.WriteLine("2. Listar Blocos");
20                Console.WriteLine("3. Sair");
21                Console.WriteLine("Escolha uma opção: ");
22
23                string opcao = Console.ReadLine();
24
25                switch (opcao)
26                {
27                    case "1":
28                        CadastrarBloco(gestor);
29                        break;
30
31                    case "2":
32                        Console.WriteLine("Lista de Blocos:");
33                        var blocos = gestor.ListarBlocos();
34                        foreach (var b in blocos)
35                        {
36                            Console.WriteLine(b);
37                        }
38                        break;
39
40                    case "3":
41                        Console.WriteLine("Saindo do programa.");
42                        Environment.Exit(0);
43                        break;
44
45                    default:
46                        Console.WriteLine("Opção inválida. Tente novamente.");
47                        break;
48                }
49            }
50
51            0 referências
52            private static void CalcularLucro(GestorBlocos gestor)
53            {
54                throw new NotImplementedException();
55            }
56
57            // Função para cadastrar um bloco com número, valor de compra e valor de venda
58            1 referência
59            static void CadastrarBloco(GestorBlocos gestor)
60            {
61                Console.WriteLine("Número do Bloco: ");
62                string numero = Console.ReadLine();
63
64                Console.WriteLine("Valor de Compra: ");
```

## Program.CS

A classe Program é a classe de entrada do programa. Ela contém o método Main, que é o ponto de partida da execução. No método Main, o programa exibe um menu de opções para o usuário, permitindo cadastrar blocos, listar blocos, calcular lucro e sair do programa.

Main(string[] args): O ponto de entrada do programa. Controla o menu principal e as interações com o usuário.

CadastrarBloco(GestorBlocos gestor): Função para cadastrar um bloco com informações fornecidas pelo usuário.

CalcularLucro(GestorBlocos gestor): Função para calcular o lucro de venda de um bloco com base no número do bloco fornecido pelo usuário.

```
1 using System;
2 using System.Collections.Generic;
3
4 public class GestorBlocos
5 {
6     private List<Bloco> blocos;
7
8     // Construtor para inicializar a lista de blocos
9     public GestorBlocos()
10     {
11         blocos = new List<Bloco>();
12     }
13
14     // Método para adicionar um bloco à lista
15     public void AdicionarBloco(Bloco bloco)
16     {
17         blocos.Add(bloco);
18     }
19
20     // Método para remover um bloco da lista
21     public void RemoverBloco(Bloco bloco)
22     {
23         blocos.Remove(bloco);
24     }
25
26     // Método para listar todos os blocos
27     public List<Bloco> ListarBlocos()
28     {
29         return blocos;
30     }
31 }
32
```

## GestorBlocos

A classe GestorBlocos é responsável por gerenciar uma coleção de blocos. Ela possui uma lista privada de objetos Bloco e métodos para adicionar, remover e listar blocos.

**blocos:** Lista privada que armazena os objetos Bloco.

**AdicionarBloco (Bloco bloco):** Adiciona um bloco à lista.

**RemoverBloco(Bloco bloco):** Remove um bloco da lista.

**ListarBlocos():** Retorna a lista de blocos atual.