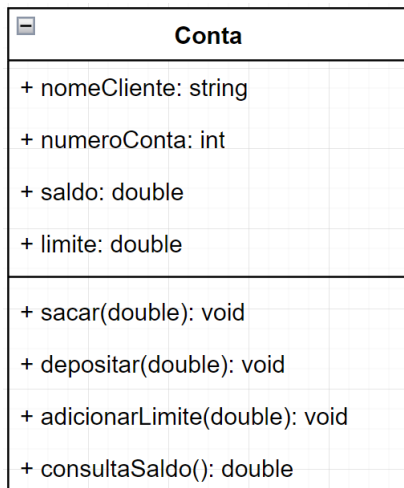


## SW-I – Prof. Anderson Vanin

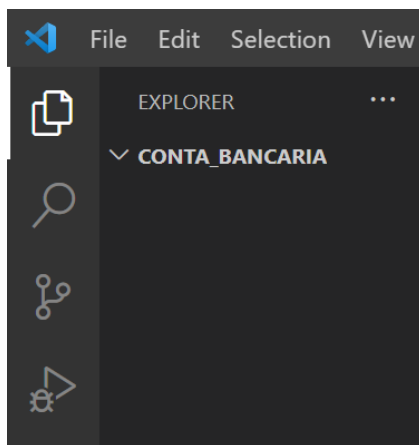
### Exercício guiado 02 – Conta Bancária

Neste exercício vamos criar uma classe responsável por operações bancárias simples como: depositar, sacar e consultar saldo.

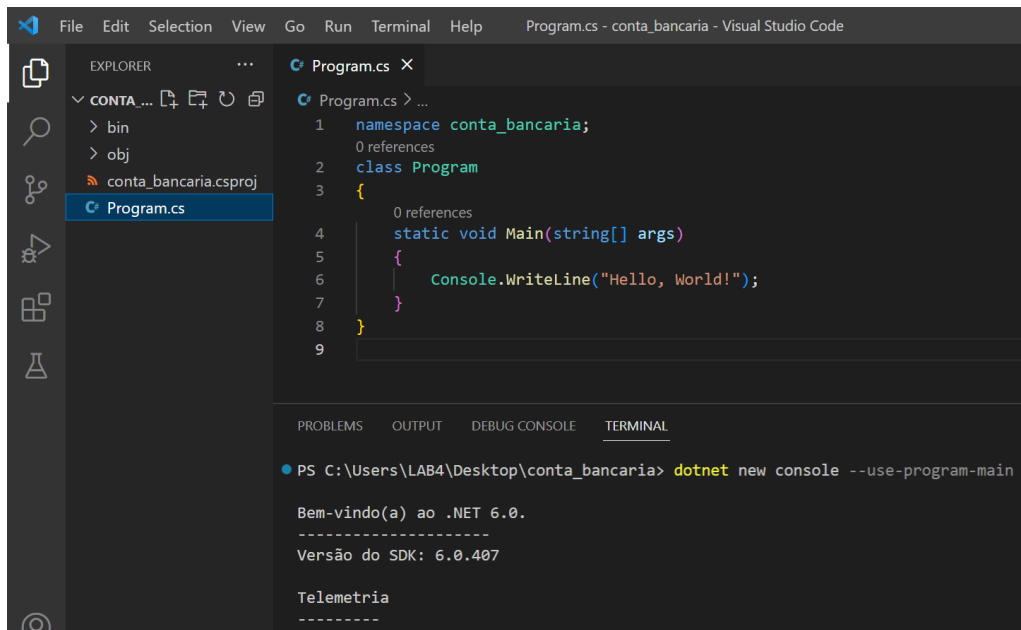
Para começar vamos examinar o diagrama de classe Conta.



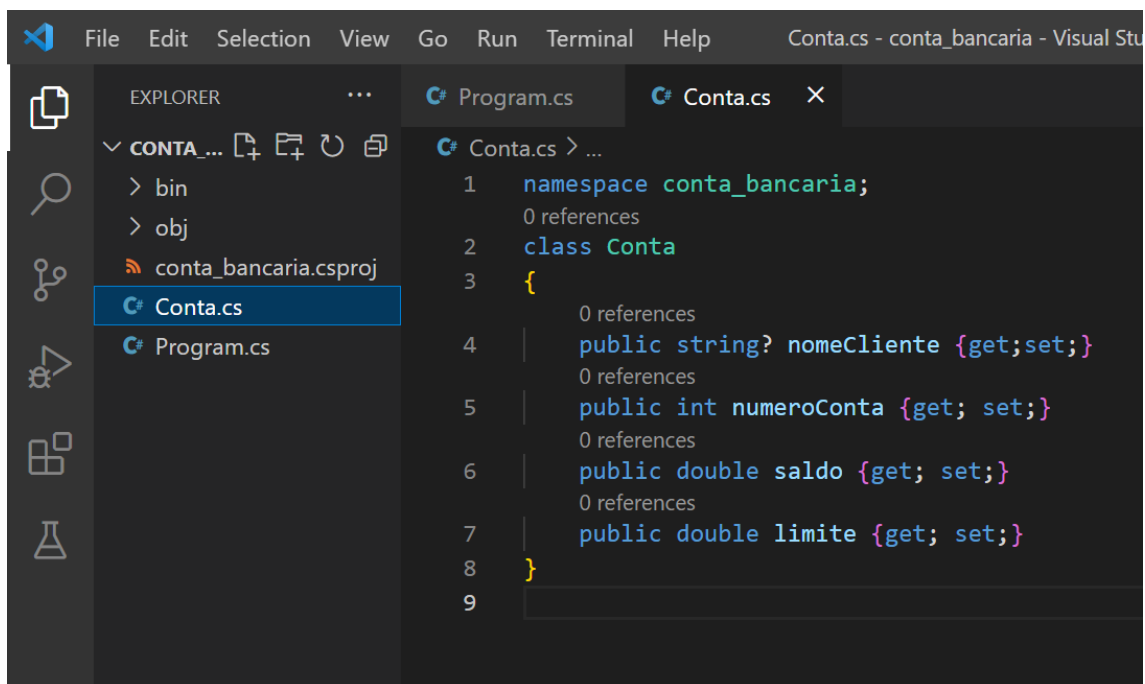
1- Crie uma pasta vazia chamada `conta_bancaria`



2- Crie a estrutura inicial do projeto em C#

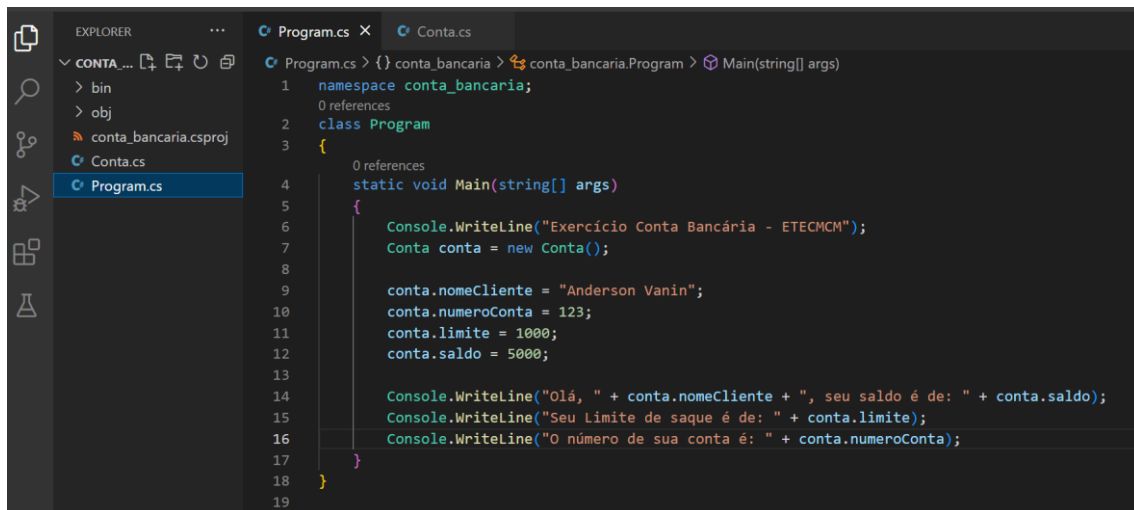


3- Crie uma classe chamada Conta e insira os atributos.



Repare que neste exemplo iremos utilizar o que chamamos de métodos assessores (get e set) que servem para recuperar e setar valores em atributos respectivamente.

4- Agora vamos para a nossa classe principal, atribuir alguns valores para nome, número, saldo e limite e depois mostrar no console os valores atribuídos.



```
1 namespace conta_bancaria;
2 class Program
3 {
4     static void Main(string[] args)
5     {
6         Console.WriteLine("Exercício Conta Bancária - ETECMCM");
7         Conta conta = new Conta();
8
9         conta.nomeCliente = "Anderson Vanin";
10        conta.numeroConta = 123;
11        conta.limite = 1000;
12        conta.saldo = 5000;
13
14        Console.WriteLine("Olá, " + conta.nomeCliente + ", seu saldo é de: " + conta.saldo);
15        Console.WriteLine("Seu Limite de saque é de: " + conta.limite);
16        Console.WriteLine("O número de sua conta é: " + conta.numeroConta);
17    }
18 }
19
```

```
PS C:\Users\LAB4\Desktop\conta_bancaria> dotnet run
Exercício Conta Bancária - ETECMCM
Olá, Anderson Vanin, seu saldo é de: 5000
Seu Limite de saque é de: 1000
O número de sua conta é: 123
PS C:\Users\LAB4\Desktop\conta_bancaria> 
```

5- Agora vamos criar um método para realizar um depósito e um para consultar o saldo. Lembre-se que o valor depositado, deve ser adicionado ao saldo disponível em conta e que ao consultar o saldo, o valor deve ser somado ao limite da conta. Exemplo: se o cliente tiver em saldo R\$1000 e ter um limite de R\$500, fazendo um depósito de R\$200, o saldo final da conta deste cliente deverá ser de R\$ 1700 (Saldo + Limite + Depósito).

```
C# Conta.cs > {} conta_bancaria > conta_bancaria.Conta
1 namespace conta_bancaria;
  2 references
2 class Conta
3 {
  2 references
4     public string? nomeCliente {get;set;}
  2 references
5     public int numeroConta {get; set;}
  4 references
6     public double saldo {get; set;}
  3 references
7     public double limite {get; set;}
8
9     //Método para realizar um depósito
  0 references
10    public void depositar (double valor){
11        this.saldo += valor;
12    }
13
14    //Método ConsultaSaldo - retorna um saldo disponível + limite
  0 references
15    public double ConsultaSaldo(){
16        return this.saldo + this.limite;
17    }
18 }
19
```

6- Volte para a classe principal e acesse as informações utilizando os métodos criados.

```
Program.cs x Conta.cs
Program.cs > {} conta_bancaria > conta_bancaria.Program > Main(string[] args)
1 namespace conta_bancaria;
  0 references
2 class Program
3 {
  0 references
4     static void Main(string[] args)
5     {
6         Console.WriteLine("Exercício Conta Bancária - ETECMCM");
7         Conta conta = new Conta();
8
9         conta.nomeCliente = "Anderson Vanin";
10
11        conta.limite = 500;
12
13        conta.depositar(200);
14
15        //Vamos chamar o método ConsultaSaldo e receber em uma variável local chamada saldo o valor disponível em saldo desse cliente
16        double saldo = conta.ConsultaSaldo();
17        //Agora para visualizar:
18        Console.WriteLine("Seu saldo é de: " + saldo);
19
20    }
21 }
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

0 número de sua conta é: 123

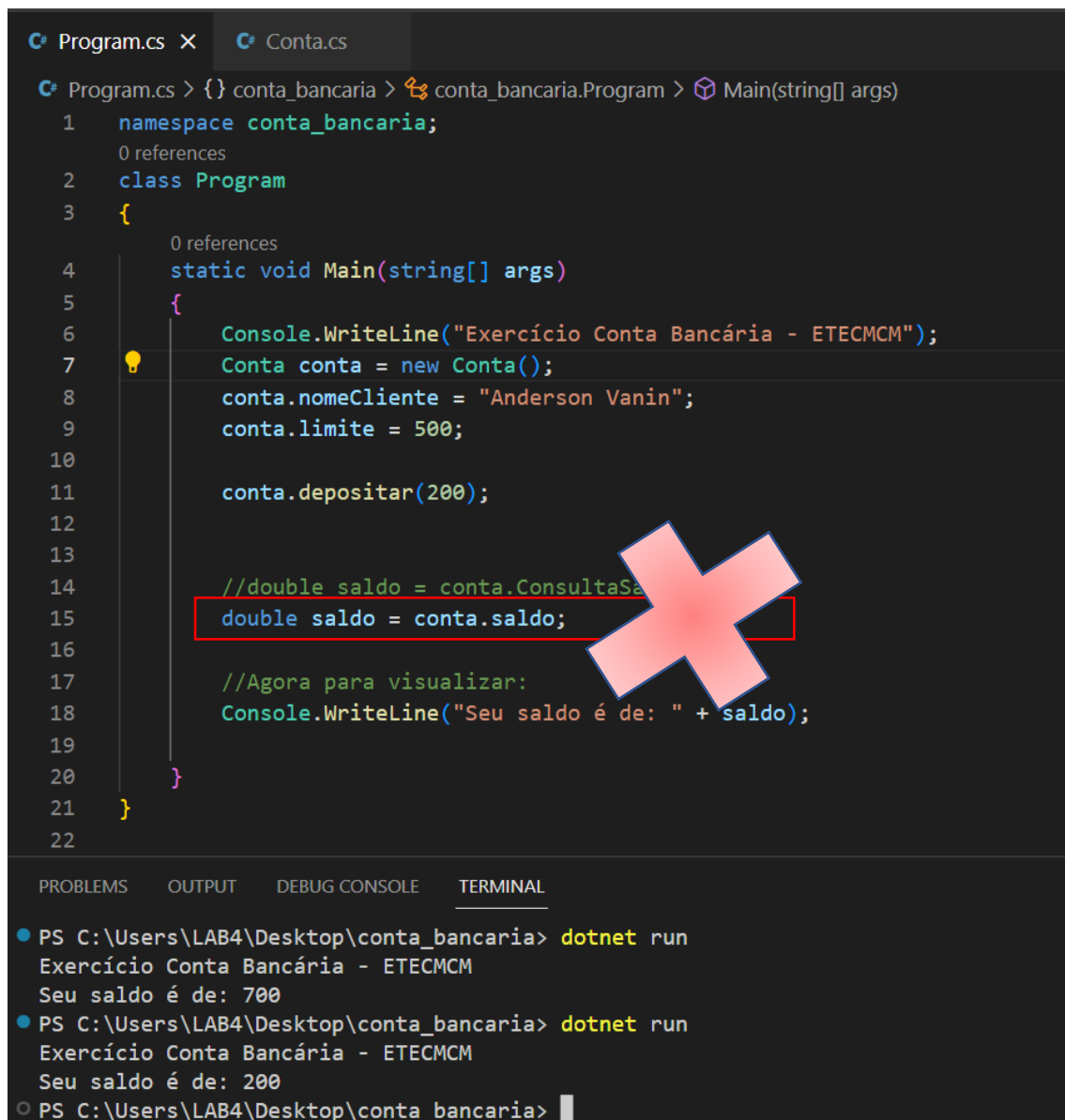
PS C:\Users\LAB4\Desktop\conta\_bancaria> dotnet run

Exercício Conta Bancária - ETECMCM

Seu saldo é de: 700

PS C:\Users\LAB4\Desktop\conta\_bancaria>

Quando consultamos o saldo pelo método `consultaSaldo`, o valor retornado já é adicionado de seu limite. Veja o que acontece se tentarmos acessar diretamente pelo atributo `saldo`.



```
Program.cs X Conta.cs
Program.cs > {} conta_bancaria > conta_bancaria.Program > Main(string[] args)
1 namespace conta_bancaria;
  0 references
2 class Program
3 {
  0 references
4     static void Main(string[] args)
5     {
6         Console.WriteLine("Exercício Conta Bancária - ETECMCM");
7         Conta conta = new Conta();
8         conta.nomeCliente = "Anderson Vanin";
9         conta.limite = 500;
10
11         conta.depositar(200);
12
13         //double saldo = conta.ConsultaSaldo();
14         double saldo = conta.saldo;
15
16         //Agora para visualizar:
17         Console.WriteLine("Seu saldo é de: " + saldo);
18
19     }
20 }
21
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

- PS C:\Users\LAB4\Desktop\conta\_bancaria> dotnet run  
Exercício Conta Bancária - ETECMCM  
Seu saldo é de: 700
- PS C:\Users\LAB4\Desktop\conta\_bancaria> dotnet run  
Exercício Conta Bancária - ETECMCM  
Seu saldo é de: 200
- PS C:\Users\LAB4\Desktop\conta\_bancaria>

**Repare que há um erro na informação exibida em relação ao saldo, pois o valor não tem acrescido o limite da conta!**

Da mesma forma, é um erro atribuir diretamente a variável `saldo` um valor sem passar pelo método de depositar.

Para evitar isso, vamos alterar a visibilidade do atributo `saldo` para que o mesmo seja acessível somente pelo método adequado.

```
Program.cs x Conta.cs x
Conta.cs > {} conta_bancaria > conta_bancaria.Conta > ConsultaSaldo()
1 namespace conta_bancaria;
  2 references
2 class Conta
3 {
  1 reference
4     public string? nomeCliente {get;set;}
  0 references
5     public int numeroConta {get; set;}
  2 references
6     private double saldo {get; set;}
  2 references
7     public double limite {get; set;}
8
9     //Método para realizar um depósito
  1 reference
10    public void depositar (double valor){
11        this.saldo += valor;
12    }
13
14    //Método ConsultaSaldo - retorna um saldo disponível + limite
  1 reference
15    public double ConsultaSaldo(){
16        return this.saldo + this.limite;
17    }
18 }
```

```
Program.cs x Conta.cs
Program.cs > ...
1 namespace conta_bancaria;
  0 references
2 class Program
3 {
  0 references
4     static void Main(string[] args)
5     {
6         Console.WriteLine("Exercício Conta Bancária - ETECMCM");
7         Conta conta = new Conta();
8         conta.nomeCliente = "Anderson Vanin";
9         conta.limite = 500;
10
11         conta.depositar(200);
12         double saldo = conta.ConsultaSaldo();
13
14         //Agora para visualizar:
15         Console.WriteLine("Seu saldo é de: " + saldo);
16     }
17 }
18
19
```

Agora adicione um método para sacar

```
Program.cs x Conta.cs x
Conta.cs > {} conta_bancaria > conta_bancaria.Conta > sacar(double valor)
1 namespace conta_bancaria;
  2 references
2 class Conta
3 {
  1 reference
4     public string? nomeCliente {get;set;}
  0 references
5     public int numeroConta {get; set;}
  3 references
6     private double saldo {get; set;}
  2 references
7     public double limite {get; set;}
8
9     //Método para realizar um depósito
  1 reference
10    public void depositar (double valor){
11        this.saldo += valor;
12    }
13
14    //Método para realizar um saque
  0 references
15    public void sacar (double valor){
16        this.saldo -= valor;
17    }
18
19    //Método ConsultaSaldo - retorna um saldo disponível + limite
  1 reference
20    public double ConsultaSaldo(){
21        return this.saldo + this.limite;
22    }
23 }
```

```
Program.cs x Conta.cs
Program.cs > {} conta_bancaria > conta_bancaria.Program > Main(string[] args)
1 namespace conta_bancaria;
  0 references
2 class Program
3 {
  0 references
4     static void Main(string[] args)
5     {
6         Console.WriteLine("Exercício Conta Bancária - ETECMCM");
7         Conta conta = new Conta();
8         conta.nomeCliente = "Anderson Vanin";
9         conta.limite = 500;
10
11         conta.depositar(200);
12         conta.sacar(50);
13
14         double saldo = conta.ConsultaSaldo();
15
16         //Agora para visualizar:
17         Console.WriteLine("Seu saldo é de: " + saldo);
18     }
19 }
20
21
22
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
PS C:\Users\LAB4\Desktop\conta_bancaria> dotnet run
● Exercício Conta Bancária - ETECMCM
Seu saldo é de: 650
○ PS C:\Users\LAB4\Desktop\conta_bancaria>
```

Mas agora imagine que o cliente não tenha um saldo maior do que o valor desejado. Aqui devemos modificar o método sacar para que o seja verificado antes de efetuar a operação de saque incluindo um condicional para que o saque só seja efetuado SE o valor disponível for maior ou igual ao total de saldo + limite.