

# SW-I HERANÇA E POLIMORFISMO

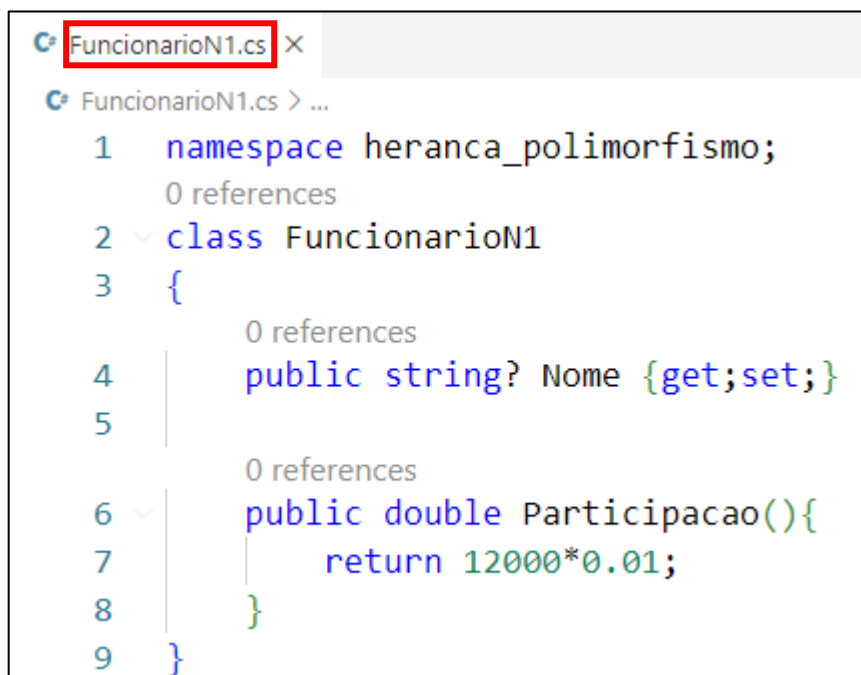
Prof. Anderson Vanin

1- Iniciar o Projeto:

Abra a pasta no VSCODE e digite o comando para criar um projeto C#

```
C:\Users\Anderson\Desktop\heranca_polimorfismo>dotnet new console --use-program-main  
O modelo "Aplicativo do Console" foi criado com êxito.
```

2- Crie três classes de Funcionários

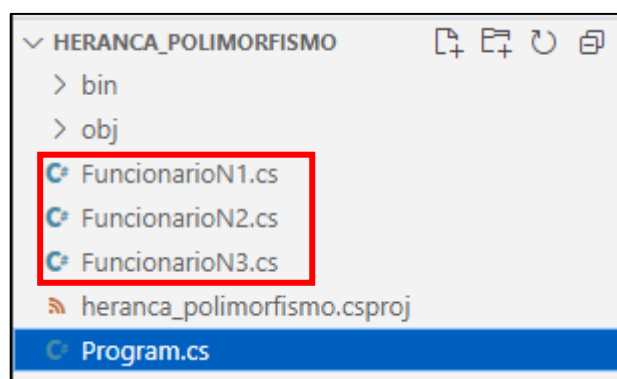


The screenshot shows the Visual Studio Code editor with a file named 'FuncionarioN1.cs' open. The code defines a namespace 'heranca\_polimorfismo' and a class 'FuncionarioN1'. The class has a public string property 'Nome' with get and set methods, and a public double method 'Participacao()' that returns the value 12000 multiplied by 0.01. The file explorer on the left shows the file 'FuncionarioN1.cs' selected.

```
1 namespace heranca_polimorfismo;  
   0 references  
2 class FuncionarioN1  
3 {  
   0 references  
4     public string? Nome {get;set;}  
5  
   0 references  
6     public double Participacao(){  
7         return 12000*0.01;  
8     }  
9 }
```

```
FuncionarioN2.cs X
FuncionarioN2.cs > ...
1 namespace heranca_polimorfismo;
  0 references
2 class FuncionarioN2
3 {
  0 references
4     public string? Nome {get;set;}
5
  0 references
6     public double Participacao(){
7         return 12000*0.02;
8     }
9 }
```

```
FuncionarioN3.cs X
FuncionarioN3.cs > ...
1 namespace heranca_polimorfismo;
  0 references
2 class FuncionarioN3
3 {
  0 references
4     public string? Nome {get;set;}
5
  0 references
6     public double Participacao(){
7         return 12000*0.03;
8     }
9 }
```



3- Instancie no programa principal um objeto de cada classe e mostre o nome e o valor de participação de cada funcionário criado a partir de sua classe.

```
Program.cs X
Program.cs > ...
1 namespace heranca_polimorfismo;
  0 references
2 class Program
3 {
  0 references
4     static void Main(string[] args)
5     {
6         FuncionarioN1 n1 = new FuncionarioN1();
7         FuncionarioN2 n2 = new FuncionarioN2();
8         FuncionarioN3 n3 = new FuncionarioN3();
9
10        n1.Nome = "Fulano";
11        n2.Nome = "Ciclano";
12        n3.Nome = "Beltrano";
13
14        Console.WriteLine("Funcionário: " + n1.Nome + " Participação: " + n1.Participacao());
15        Console.WriteLine("Funcionário: " + n2.Nome + " Participação: " + n2.Participacao());
16        Console.WriteLine("Funcionário: " + n3.Nome + " Participação: " + n3.Participacao());
17    }
18 }
```

```
C:\Users\Anderson\Desktop\heranca_polimorfismo>dotnet run
Funcionário: Fulano Participação: 120
Funcionário: Ciclano Participação: 240
Funcionário: Beltrano Participação: 360
```

4- Para aplicar o conceito de herança, precisamos criar uma classe mãe. Crie a classe Funcionario.

```
Funcionario.cs X
Funcionario.cs > ...
1 namespace heranca_polimorfismo;
  0 references
2 class Funcionario
3 {
  0 references
4     public string? Nome {get;set;}
  0 references
5     public int Idade {get;set;}
6
7 }
```

Altere as classes FuncionarioN1, FuncionarioN2 e FuncionarioN3, para informar que estas 3 classes irão herdar os atributos da classe Funcionario e remova seus atributos. Deixe somente o método criado anteriormente.

```
FuncionarioN1.cs > ...  
1 namespace heranca_polimorfismo;  
  2 references  
2 class FuncionarioN1:Funcionario  
3 {  
4 |  
  1 reference  
5 public double Participacao(){  
6     return 12000*0.01;  
7 }  
8 }
```

```
FuncionarioN2.cs > ...  
1 namespace heranca_polimorfismo;  
  2 references  
2 class FuncionarioN2:Funcionario  
3 {  
4 |  
  1 reference  
5 public double Participacao(){  
6     return 12000*0.02;  
7 }  
8 }
```

```
FuncionarioN3.cs > ...  
1 namespace heranca_polimorfismo;  
  2 references  
2 class FuncionarioN3:Funcionario  
3 {  
4 |  
  1 reference  
5 public double Participacao(){  
6     return 12000*0.03;  
7 }  
8 }
```

5- Rode a aplicação e veja que o atributo Nome, é herdado da classe Mãe Funcionário e utilizado pelas classes filhas FuncionarioN1, FuncionarioN2 e FuncionarioN3.

6- Definimos Polimorfismo como um princípio a partir do qual as classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de maneira diferente para cada uma das classes derivadas. Com o Polimorfismo, os mesmos atributos e objetos podem ser utilizados em objetos distintos, porém, com implementações lógicas diferentes.

Altere a classe Funcionario

```
Funcionario.cs > ...
1 namespace heranca_polimorfismo;
  3 references
2 class Funcionario
3 {
  6 references
4     public string? Nome {get;set;}
  0 references
5     public int Idade {get;set;}
6
  0 references
7     public virtual double Participacao(){
8         return 12000;
9     }
10 }
```

Altere as classes FuncionarioN1, FuncionarioN2 e FuncionarioN3

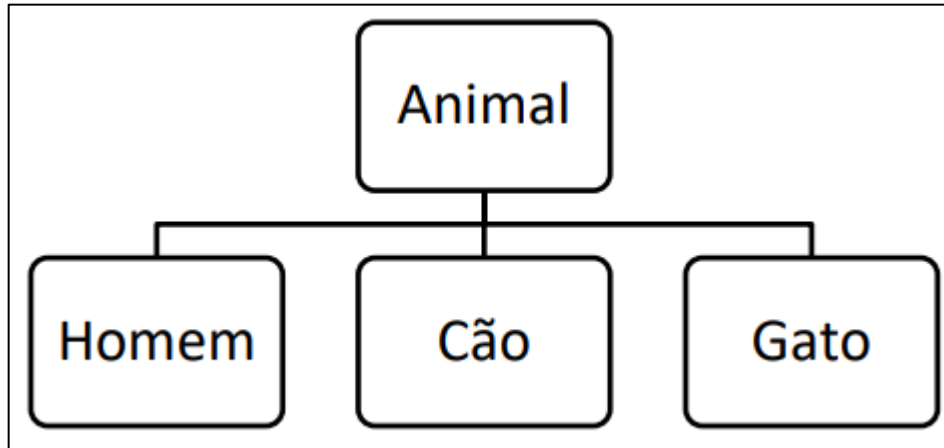
```
FuncionarioN1.cs > ...
1 namespace heranca_polimorfismo;
  2 references
2 class FuncionarioN1:Funcionario
3 {
4 |
5 |     2 references
6 |     public override double Participacao()
7 |     {
8 |         return base.Participacao()*0.01;
9 |     }
9 }
```

```
FuncionarioN2.cs > ...
1 namespace heranca_polimorfismo;
  2 references
2 class FuncionarioN2:Funcionario
3 {
4 |
5 |     4 references
6 |     public override double Participacao()
7 |     {
8 |         return base.Participacao()*0.02;
9 |     }
9 }
```

```
FuncionarioN3.cs > ...
1 namespace heranca_polimorfismo;
  2 references
2 class FuncionarioN3:Funcionario
3 {
4 |
5 |     6 references
6 |     public override double Participacao()
7 |     {
8 |         return base.Participacao()*0.03;
9 |     }
9 }
```

## Exercícios

1- Faça uma classe Animal com um método “fala”. Faça as classes Homem, Cão e Gato, herdando de animal, redefinindo o método “fala” para retornar “Oi”, “Au au” e “Miau”, respectivamente.



2- Os contratos podem ser contratos de pessoa física e contrato de pessoa jurídica. Os contratos de pessoa física também têm o CPF e a idade do contratante. Os contratos de pessoa jurídica possuem o CNPJ e a inscrição estadual da empresa contratante. Usando o conceito de herança, crie a classe ContratoPessoaFisica herdando da classe Contrato e com os atributos adicionais do Contrato Pessoa Física. Em seguida, crie a classe ContratoPessoaJuridica herdando da classe Contrato e com os atributos adicionais do Contrato Pessoa Jurídica.

3- O valor da prestação de um contrato é calculado por um método calcularPrestação(), como sendo o valor do contrato dividido pelo prazo. Este método calcularPrestação() existe para todos os Contratos. Entretanto, para os contratos de pessoa jurídica existe um adicional de 3 reais no valor de cada prestação e para os contratos de pessoa física o valor da prestação também tem um adicional no valor da prestação que deve ser calculado de acordo com a idade do contratante:

- idade <= 30 tem adicional de 1,00
- idade <= 40 tem adicional de 2,00
- idade <= 50 tem adicional de 3,00
- idade > 50 tem adicional de 4,00