ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

DESENVOLVIMENTO DE SISTEMAS – 2022/2

Luiz Fernando Souza / Quitéria Danno

Aula 02 – Classes e Enumerações – Introdução ao Padrão MVC (Model-View-Controller)

Nesta aula faremos a interpretação de diagramas de classe UML para construção de Classes e Enumeradores, programando as características de uma classe (atributos/métodos) e as operações (Métodos). Junto a isso, ao criar as classes, separaremos as mesmas por tipo, buscando uma melhor organização do projeto de acordo com as responsabilidades de cada classe. Esse modelo é conhecido como MVC – Model-View-Controller.

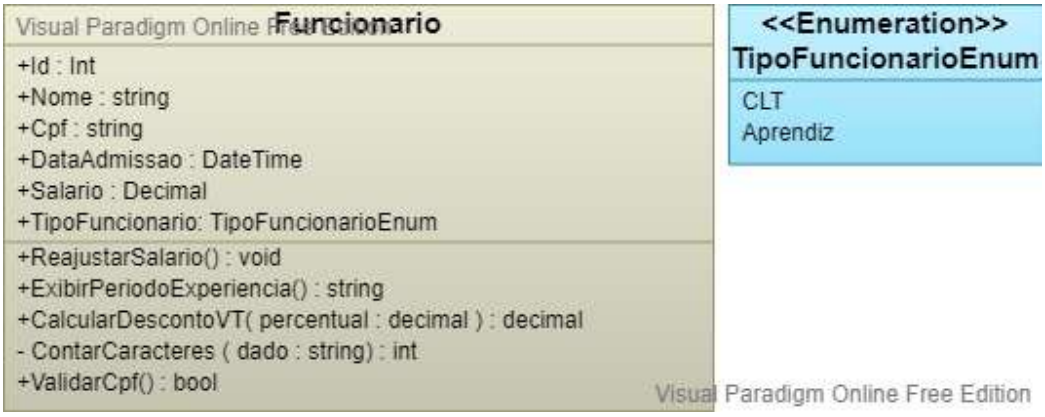
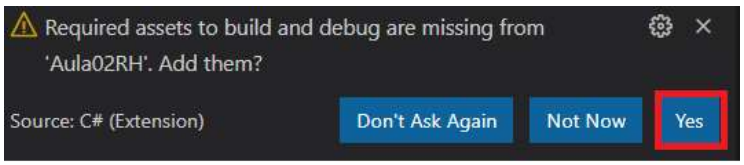


Diagrama de classe do padrão UML

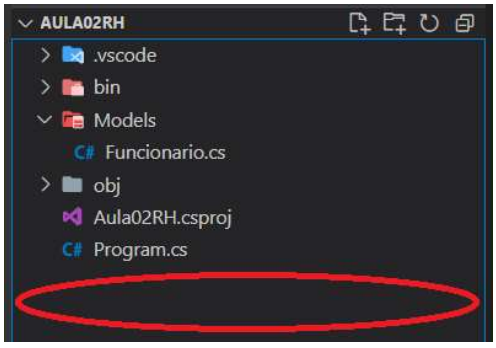
Uma classe é a representação de uma entidade do mundo real que viabilizará a construção de projetos

1. Crie a pasta Aula02RH na pasta em que costuma criar os projetos, abra a mesma no VS Code, ative o terminal e execute o comando para criação de um projeto do tipo Console Application.

2. Clique na classe Program, aguarde até aparecer a notificação para ativação da extensão do C# e clique em “Yes”



3. Clique com o direito do mouse dentro do retângulo de borda azul fina, abaixo da classe Program e escolha a opção New Folder, nomeando a pasta como **Models**.



4. Clique com o direito do mouse na pasta Models e crie uma classe chamada **Funcionario**, realizando a programação das propriedades a seguir.

   Dica: Digite “Prop” e pressione TAB para o atalho de criação de propriedades. O Curso automaticamente selecionará o tipo de variável para que você mantenha ou digite outra, após isso clique em TAB mais uma vez para o cursor selecionar o nome da propriedade para que você insira conforme o diagrama.

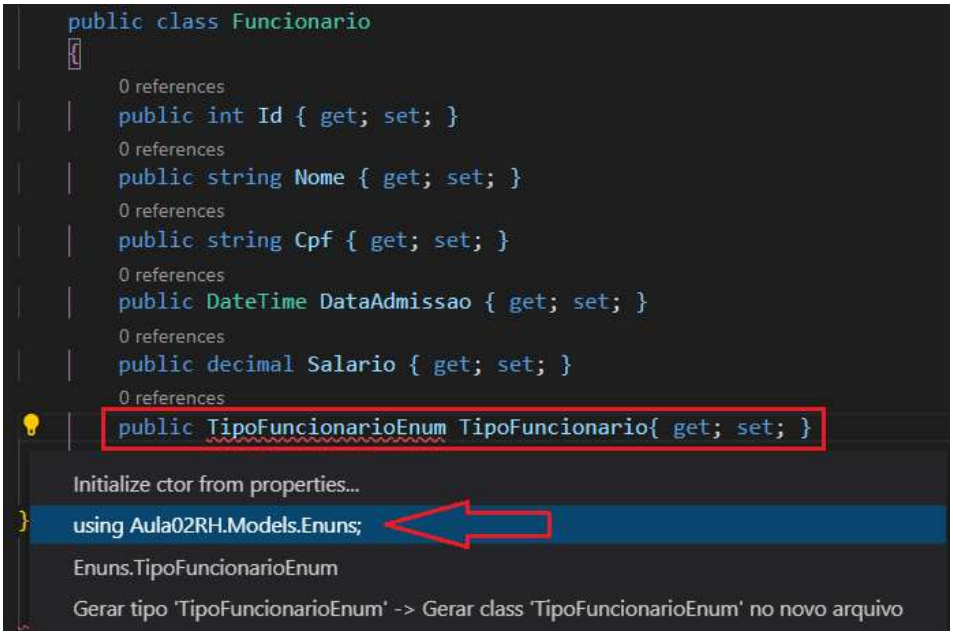
```
namespace Aula02RH.Models
{
    0 references
    public class Funcionario
    {
        0 references
        public int Id { get; set; }
        0 references
        public string Nome { get; set; }
        0 references
        public string Cpf { get; set; }
        0 references
        public DateTime DataAdmissao { get; set; }
        0 references
        public decimal Salario { get; set; }


    }
}
```

5. Clique com o direito na pasta Models e crie uma pasta chamada **Enuns**. Clique com o direito na pasta Enuns e crie uma classe chamada **TipoFuncionarioEnum**, fazendo as modificações abaixo

```
namespace Aula02RH.Models.Enuns
{
    1 reference
    public enum TipoFuncionarioEnum
    {
        0 references
        CLT=1,
        0 references
        Aprendiz=2
    }
}
```

- O tipo de arquivo foi modificado para ser um enum (enumeração). Usamos esse tipo quando temos dados que não se alteram facilmente e pode ser identificado por um número. Mais a frente isso fará muito sentido na hora de fazer comparações.

6. Volte à classe Funcionario e declare a enumeração criada na forma de propriedade. Perceba que se você digitou o tipo de maneira correta (TipoFuncionarioEnum) ele não vai reconhecer, pois os arquivos estão em pastas diferentes, então será necessário indicar qual o caminho em que a enumeração está. Você pode (1) clicar na lâmpada que aparece ao lado ou (2) usar o atalho CTRL + . (ponto) para fazer a janela de resolução de erros aparecer. Escolha o using da seta e verá que no topo da classe será adicionada uma referência ao endereço da enumeração.



7. Programe os métodos da classe antes do fechamento do corpo dela, conforme abaixo

```
0 references
public TipoFuncionarioEnum TipoFuncionario{ get; set; }
0 references
public void ReajustarSalario()
{
    Salario = Salario + (Salario * 10 /100);
}
0 references
public string ExibirPeriodoExperiencia()
{
    string periodoExperiencia =
        string.Format("Períodos de Experiência: {0} até {1}", DataAdmissao, DataAdmissao.AddMonths(3));
    return periodoExperiencia;
}

0 references
public decimal CalcularDescontoVT(decimal percentual)
{
    decimal desconto = this.Salario * percentual/100;
    return desconto;
}
```

```
private int ContarCaracteres(string dado)
{
    return dado.Length;
}

0 references
public bool ValidarCpf()
{
    if(ContarCaracteres(Cpf) == 11)
        return true;
    else
        return false;
}
```

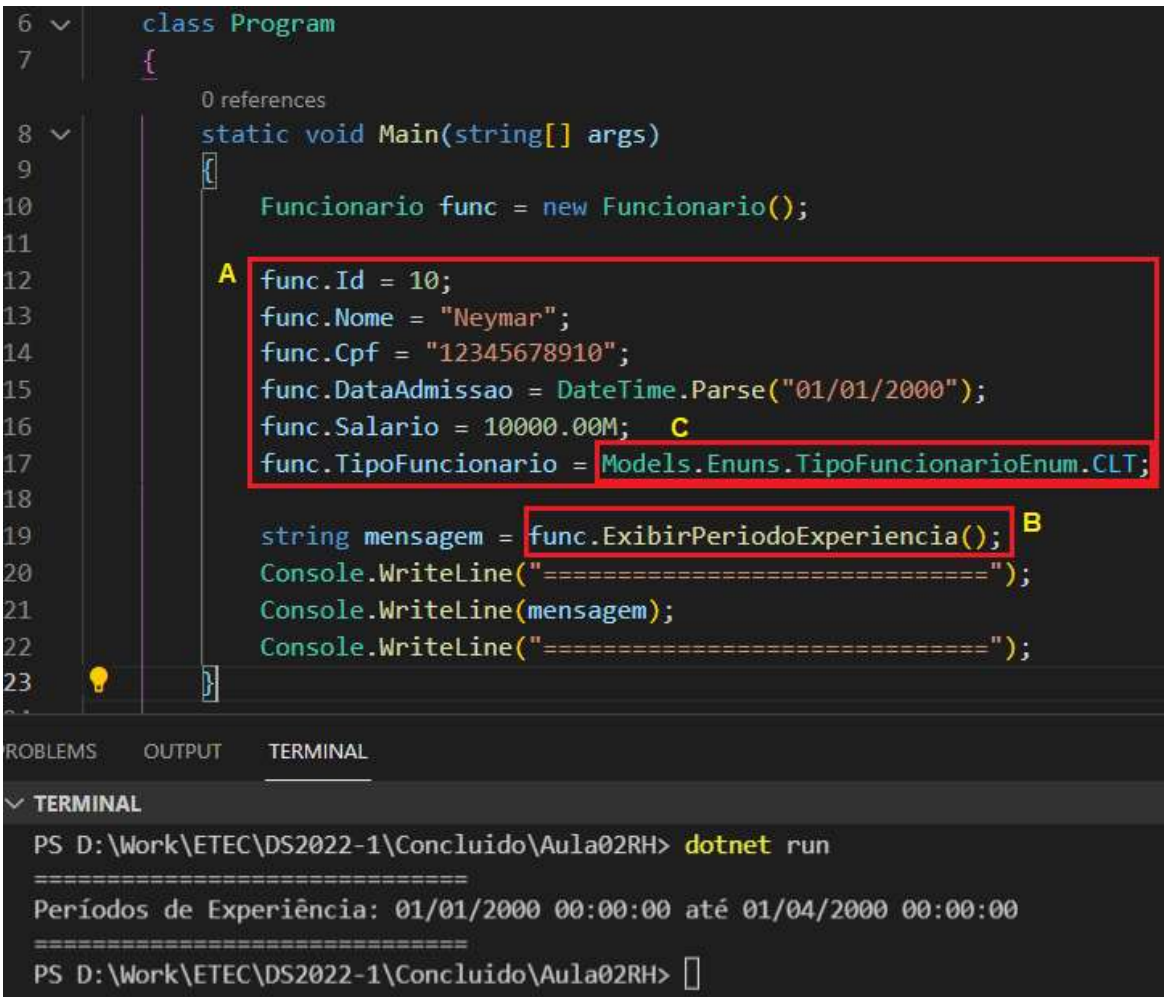
- Salve e execute o comando build no terminal para confirmar que não existe erros no código até aqui

8. No método principal da classe program iremos fazer a criação de uma cópia da classe em memória, o que chamamos de instância, para pode alimentar ela com dados próprios. Será necessário fazer o using para reconhecer a namespace Models clicando em cima do erro e usando o atalho CTRL + . ou clicando na lâmpada.

```
class Program
{
    0 references
    static void Main(string[] args)
    {   A                                   B
        Funcionario func = new Funcionario();
    }
}
```

(A) Declaração de uma variável do tipo **Funcionario** com o nome de **func**, representada pela classe Funcionario.
(B) Atribuição para func de uma cópia criada em memória, através do operador new, desta maneira, func passa a ser um objeto, do tipo Funcionario como mencionado anteriormente.

9. Agora será possível fazer a alimentação de cada propriedade presente na classe Funcionario através do objeto func e fazer a chamada para os métodos contidos na classe



(A) Inserção de dados nas propriedades do objeto func que é do tipo Funcionario
(B) Chamada do método para exibir o período de experiência.
(C) Perceba que a possibilita escolher apenas itens contidos dentro do arquivo dela.

Mas se quisermos fazer com que as propriedades sejam alimentadas com itens que venham da tela, como podemos fazer?

10. Siga a programação abaixo para alimentar as propriedades através da digitação no console

```
Funcionario func = new Funcionario();

Console.WriteLine("Digite o Id do funcionário: ");
func.Id = int.Parse(Console.ReadLine());

Console.WriteLine("Digite o nome do funcionário: ");
func.Nome = Console.ReadLine();

Console.WriteLine("Digite o CPF: ");
func.Cpf = Console.ReadLine();

Console.WriteLine("Digite a data de Admissão: ");
func.DataAdmissao = DateTime.Parse(Console.ReadLine());

Console.WriteLine("Digite o Salário: ");
func.Salario = decimal.Parse(Console.ReadLine());

Console.WriteLine("Escolha o tipo de Funcionário (1 - CLT / 2 - Aprendiz): ");
int opcao = int.Parse(Console.ReadLine());

//Operador Ternário - Interpretação: Se a condição do parenteses for verdadeira,
//escolhe o que está depois da "?", Caso contrário, escolhe o que está de pois dos ":"
func.TipoFuncionario = (opcao == 1) ? TipoFuncionarioEnum.CLT : TipoFuncionarioEnum.Aprendiz;

func.ReajustarSalario();
decimal valorDescontoVT = func.CalcularDescontoVT(6);

Console.WriteLine("==============================");
Console.WriteLine($"O salário reajusta do é {func.Salario}.\n");
Console.WriteLine($"O Desconto do VT  é {valorDescontoVT}.\n");
Console.WriteLine("==============================");
```

- Execute o programa para testar as condições:

Atividades:

(1) Utilize o programa desenvolvido até aqui para apresentar os valores/informações da execução dos demais métodos. Dica: programar abaixo do operador ternário.

(2) Crie uma forma do usuário escolher qual método ele quer que tenha a informação exibida em tela, após a digitação dos dados ter sido feita.