ROTEIRO DE AULA

Tema: Aula 2 – POO (Programação Orientada a Objetos)

Data: 26/9/2019

Semana:

Tempo estimado: 4 horas.

Requisitos e ferramentas para aula:

VsCode.

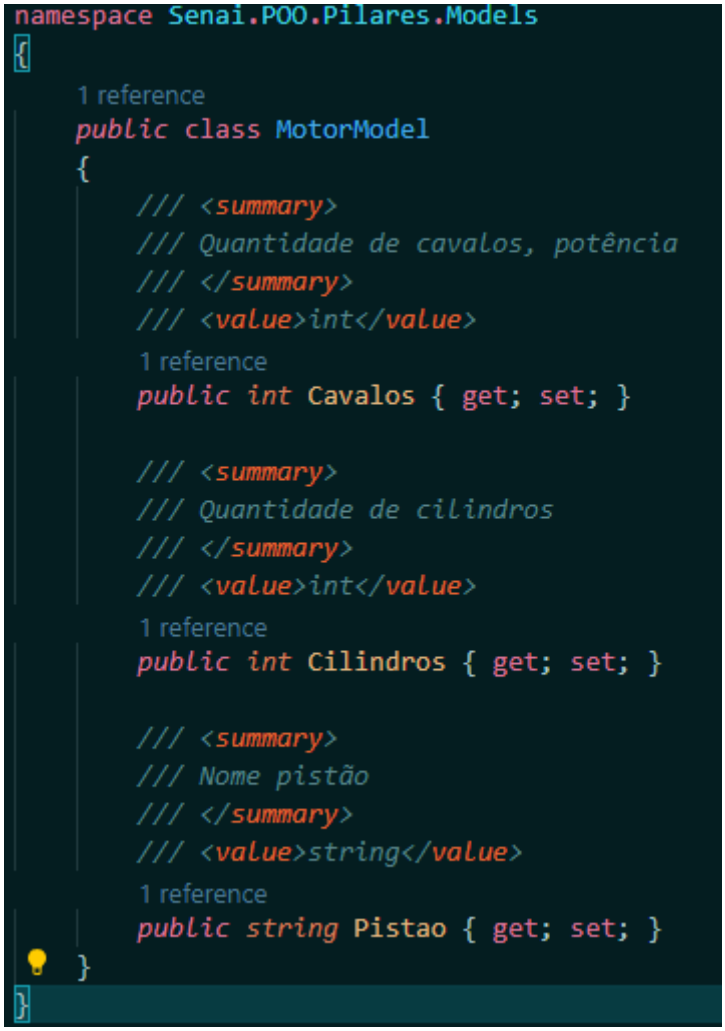
Descrever roteiro de aula:

1. Iniciar explicando o que veremos durante a aula, aprofundando-se nos pilares de POO, serão explicados composição, herança e encapsulamento, junto com Controllers. **Anotar na lousa para lembrar.
2. Criar novo projeto em console com dotnet.
3. Criar Pasta de Models.

4. Criar classe CarroModel.

```
namespace Senai.POO.Pilares.Models
{
    3 references
    public class CarroModel
    {
        /// <summary>
        /// Marca do carro
        /// </summary>
        /// <value>string</value>
        0 references
        public string Marca { get; set; }

        /// <summary>
        /// Modelo do carro
        /// </summary>
        /// <value>string</value>
        0 references
        public string Modelo { get; set; }

        /// <summary>
        /// Cor do carro
        /// </summary>
        /// <value>string</value>
        0 references
        public string Cor { get; set; }

        /// <summary>
        /// Placa do carro
        /// </summary>
        /// <value>string</value>
        0 references
        public string Placa { get; set; }

        /// <summary>
        /// Objeto do motor do carro
        /// </summary>
        /// <value>MotorModel</value>
        3 references
        public MotorModel Motor { get; set; }

        /// <summary>
        /// Propriedade que diz se o carro está ligado ou não
        /// </summary>
        /// <value>Se for 'true', o carro está ligado. Se for 'false', o carro está desligado</value>
        4 references
        public bool Ligado {get; set;}
    }
}
```

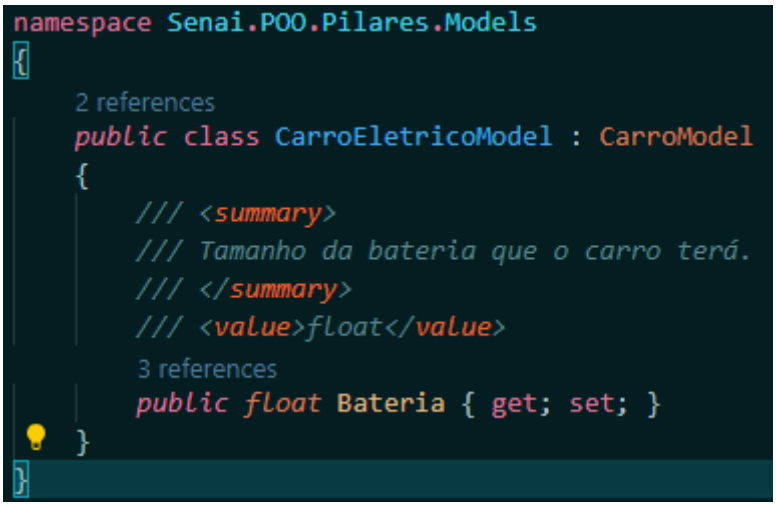
5. Criar a classe do Motor do carro, para explicar composição, um objeto dentro de um objeto.



```
namespace Senai.POO.Pilares.Models
{
    1 reference
    public class MotorModel
    {
        /// <summary>
        /// Quantidade de cavalos, potência
        /// </summary>
        /// <value>int</value>
        1 reference
        public int Cavalos { get; set; }

        /// <summary>
        /// Quantidade de cilindros
        /// </summary>
        /// <value>int</value>
        1 reference
        public int Cilindros { get; set; }

        /// <summary>
        /// Nome pistão
        /// </summary>
        /// <value>string</value>
        1 reference
        public string Pistao { get; set; }
    }
}
```

6. Incorporar MotorModel no CarroModel.
7. Criar CarroController com métodos: INSTÂNCIAR CARROMODEL NO CONTROLLER
    a. Ligar;
    b. Desligar;
    c. Acelerar; se o carro estiver ligado.
    d. Freiar; se o carro estiver ligado.
    e. Cadastrar Motor; Mostrar utilização do objeto Motor dentro do objeto Carro.
8. Instanciar CarroController e demontrar código no Program.cs.

9. Criar CarroEletricoModel herdando CarroModel para demonstrar herança.



```
namespace Senai.POO.Pilares.Models
{
    2 references
    public class CarroEletricoModel : CarroModel
    {
        /// <summary>
        /// Tamanho da bateria que o carro terá.
        /// </summary>
        /// <value>float</value>
        3 references
        public float Bateria { get; set; }
    }
}
```

10. Criar CarroEletricoController herdando CarroController para demonstrar herança.

```
using Senai.POO.Pilares.Models;

namespace Senai.POO.Pilares.Controllers
{
    2 references
    public class CarroEletricoController : CarroController
    {
        /// Instanciando Objeto CarroEletricoModel
        3 references
        CarroEletricoModel carro = new CarroEletricoModel();

        /// <summary>
        /// Declarando método de carregar bateria
        /// </summary>
        /// <param name="carga"></param>
        2 references
        public void CarregarBateria(float carga)
        {
            if (carro.Bateria < 100){
                carro.Bateria += carga;
            }
        }

        /// <summary>
        /// Retorna o Status da bateria.
        /// </summary>
        /// <returns>float</returns>
        1 reference
        public float StatusBateria()
        {
            return carro.Bateria;
        }
    }
}
```

11. Instanciar CarroEletricoController no Program.cs e demonstrar o código, explicar o Encapsulamento, já que as informações estão sendo puxadas do Controller e não do Model, por isso, temos acesso somente às informações que o Controller retorna, privando propriedades do Model.

Descrever atividades:

Os alunos irão desenvolver o código junto com o professor, o que deverá durar o dia todo. Porém, caso sobre tempo deverão começar os exercícios que seriam aplicados no dia seguinte.

Tempo atividades: 3hrs

Descrever dinâmica:

Tempo de dinâmica:

Descrever atividade extra: