Curso de Orientação a Objetos em C#	Aula05
Professora: Luana Fernandes	Turma: Quinta – 19h30

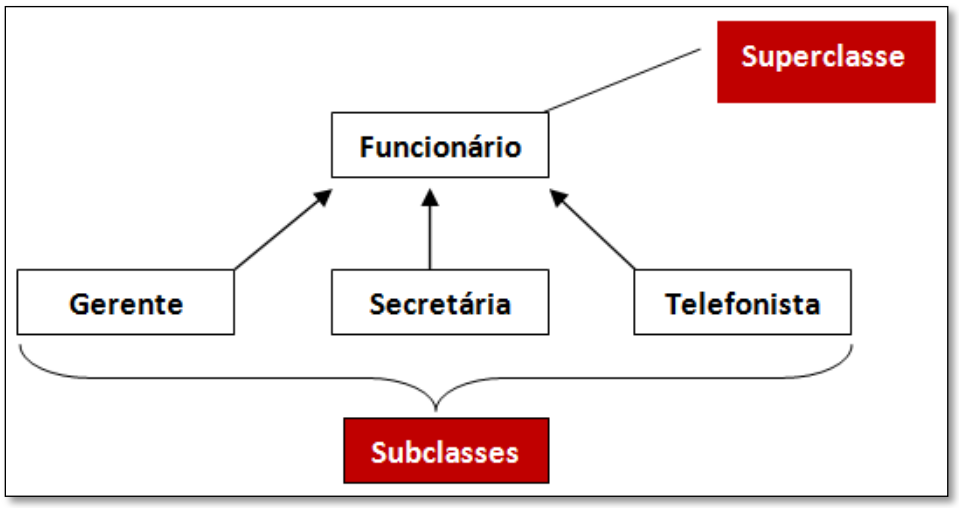
Conteúdo desta aula:

- Herança
- Associação
- Exercícios

HERANÇA

Na Programação Orientada a Objetos o significado de herança é o mesmo que no mundo real. Assim como um filho pode herdar alguma característica do pai, em Orientação a Objetos é permitido que uma classe herde atributos e métodos da outra. Para isso, os modificadores de acessos das classes, métodos e atributos que serão herdados devem estar com visibilidade **public** e **protected** para que as classes filhas tenham acesso a eles.

Uma das grandes vantagens de usar o recurso da herança é na reutilização do código. Esse reaproveitamento pode ser acionado quando se identifica que o atributo ou método de uma classe será igual para as outras. Para efetuar a herança de uma classe é utilizada a palavra reservada chamada **extends**.



Para saber se estamos aplicando a herança corretamente, realiza-se o teste "**É UM**". Esse teste simples ajuda a detectar se a subclasse pode herdar a superclasse.

Por exemplo, na imagem anterior, vemos que a classe Gerente herda da classe Funcionário. Se for aplicado o teste "É UM", nota-se que o teste é aprovado, pois o **Gerente "É UM" Funcionário**.

Classe Mãe		
public class Funcionario { private string nome; private double salario; }		
Classe filha	**Classe filha**	**Classe filha**
Gerente **extends** Funcionario { private string usuario; private string senha; }	Secretaria **extends** Funcionario { private int ramal; }	Telefonista **extends** Funcionario { private int estacaoDeTrabalho; }

**O public class foi omitido das classes filhas apenas para organizar melhor as informações nas tabelas.*

ASSOCIAÇÃO

A associação de classes indica um relacionamento do tipo "tem um", ou seja, quando um elemento de uma classe é outra classe. Por exemplo, uma pessoa tem um carro e isso indica que a classe Pessoa tem uma associação com a classe Carro. Esse tipo de relacionamento entre classes é importante, porque define como as classes interagem entre elas nas aplicações.

O exemplo a seguir mostra como implementar uma associação **um para um**. A associação é feita entre as classes Pessoa e Carro. A classe Carro tem os atributos modelo, placa, ano e valor; e a classe Pessoa tem os atributos nome, endereço, telefone e dataNascimento. Além disso, ela tem um relacionamento com a classe Carro.

Classe Carro	Classe Pessoa
public class Carro { private string modelo; private string placa; private int ano; private double valor; }	public class Pessoa { private string nome; private string endereco; private string telefone; private DateTime dataNascimento; **// Relacionamento com a classe Carro** **private Carro carro;** }

Além do relacionamento um para um, também existem o relacionamento **um para N** e **N para N**. Esses relacionamentos são modelados com uma lista de objetos de uma classe para outro como, por exemplo, se uma pessoa pode ter mais de um carro, é necessário mapear esse relacionamento com uma lista de carros na classe Pessoa. O exemplo a seguir mostra a alteração necessária na classe Pessoa, mas na classe Carro não é necessária nenhuma alteração.

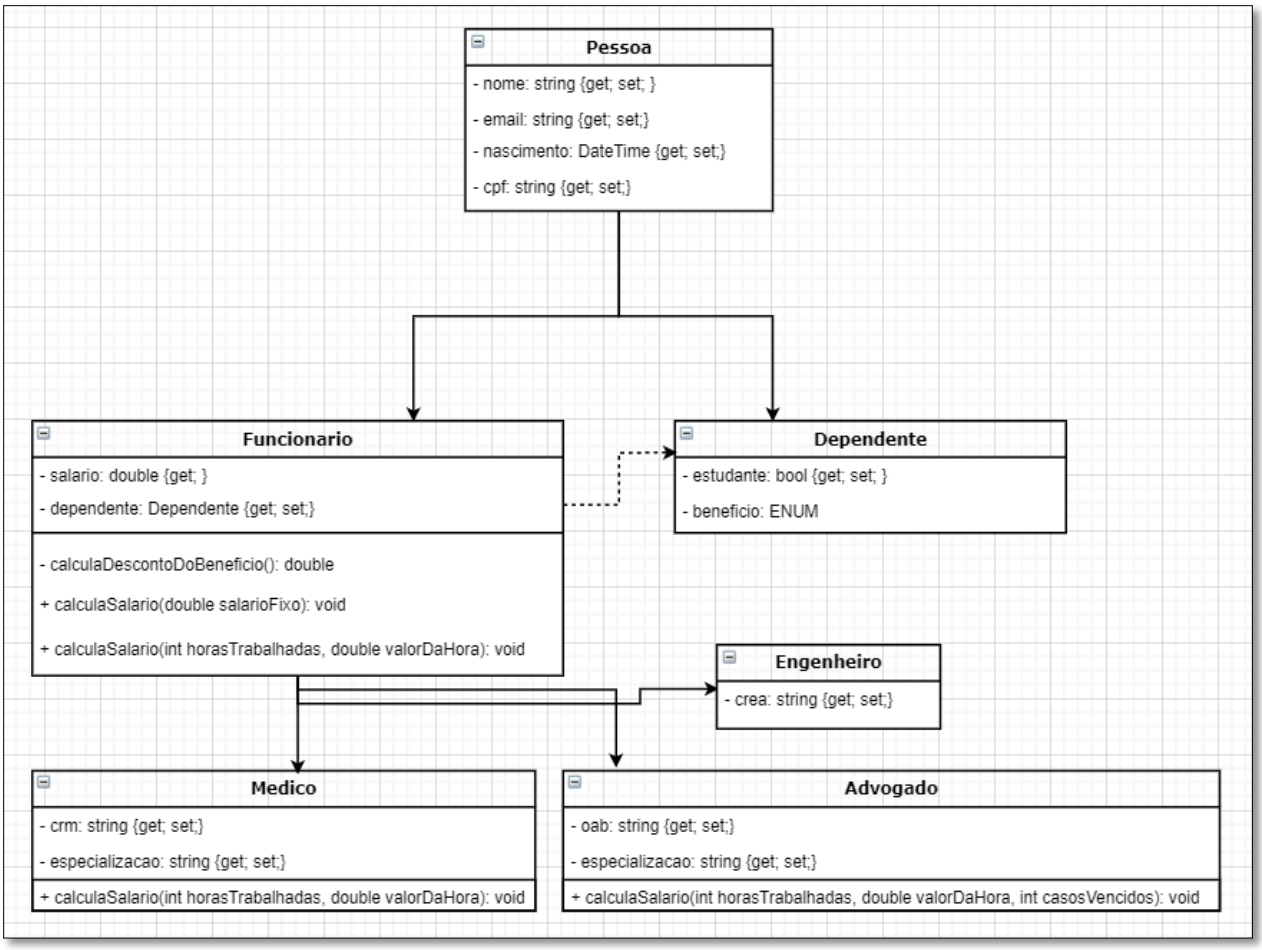
Classe Carro	Classe Pessoa
public class Carro { private string modelo; private string placa; private int ano; private double valor; }	public class Pessoa { private string nome; private string endereco; private string telefone; private DateTime dataNascimento; **// Relacionamento com a classe Carro** **private List<Carro> carros = new List<Carro>();** }

EXERCÍCIOS

Construa um pequeno sistema de folha de pagamento para uma empresa de extração de petróleo chamada Tamus. A seguir estão algumas instruções que você deve seguir para construir o sistema, além do diagrama de classe. **Atenção, nem todas as informações necessárias estão no diagrama. Por isso, leia toda a especificação com atenção e inclua as regras no seu código da melhor maneira possível**.

A Tamus possui vários tipos de funcionários. Alguns recebem por hora de trabalho, outros recebem um salário fixo mensal. Existem também funcionários que recebem valores adicionais.

Apenas algumas longas reuniões, sua equipe construiu um diagrama de classes para ajudar o desenvolvimento.



Regras para os funcionários:

- O nome e CPF são campos obrigatórios em para todos os funcionários.

Médico (a)

- Não é possível criar médicos sem CRM.
- Recebem o salário por hora.
- Recebem um adicional fixo de 8% referente ao risco de contágio por contato com doenças infecciosas. Além de um adicional de 15% se trabalharem mais de 120 horas no mês e 20% se trabalharem mais de 170 horas.

Advogado (a)

- Não é possível criar advogados sem OAB.
- Recebem o salário por hora
- Recebem um adicional de R$120 para cada caso vencido em tribunal.

Engenheiro (a)

- Não é possível criar engenheiros sem CREA.

- Recebem o salário por hora
- Não possuem adicional de serviço

Secretário (a)

- Recebem salário fixo mensal
- Não possuem adicional de serviço

B**enefícios aos dependentes**:

- Todos os funcionários têm direito a benefício para um dependente baseado na seguinte tabela de desconto:
    - Plano de Saúde → 4%
    - Auxílio Educação → 5%
    - Auxílio Nutricional e Esportivo → 6%
- Por dependente considera-se filho (a) biológico (a) ou adotivo (a).
- Para ter direito ao Auxílio Educação o dependente tem que obrigatoriamente ser estudante.
- O limite de idade para os dependentes terem direito ao benefício é de 24 anos, ou 30 se ainda forem estudantes.