Roteiro

1 – Herança em C#

Para criar uma herança em C#, basta adicionar na declaração da classe um dois pontos (:) seguido do nome da classe que será pai.

Exemplo: 02herança

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Teste
{
    public class Pessoa
    {
        public string nome { set; get; }
        public int idade { set; get; }

        public  void comer(string alimento)
        {
            Console.WriteLine("A Pessoa comendo {0}",alimento);
        }
    }

    public class Luciani : Pessoa
    {
        public int sapato { get; set; }
    }

     public static void Main()
    {
        string comida;
        Pessoa teste = new Pessoa();
        Console.Write("Qual comida você dará a Pessoa? ");
        comida = Console.ReadLine();
        teste.comer(comida);
        Luciani teste2 = new Luciani();
        teste2.sapato = 37;
        teste2.comer(comida);
        Console.WriteLine("Luciani calça {0}.", teste2.sapato);
        Console.ReadLine();
    }
}
```

2 – Override em C#

Como indicar que o método poderá ser reescrito? Utilizando o atributo *virtual* na hora de construir o método da classe pai, exemplo:

```
public virtual void comer(string alimento)
        {
            Console.WriteLine("A Pessoa comendo {0}",alimento);
        }
```

Para fazer a classe ser sobrescrita, é necessário utilizar a herança. Veja o exemplo 03override:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Teste
{
    public class Pessoa
    {
        public string nome { set; get; }
        public int idade { set; get; }

        public virtual void comer(string alimento)
        {
            Console.WriteLine("A Pessoa comendo {0}",alimento);
        }
    }

    public class Sergio : Pessoa
    {
        public override void comer(string alimento)
       {
           if (alimento == "peixe")
               Console.WriteLine("Sérgio não gosta de peixe. Não vai comer.");
           else
               base.comer(alimento); //base chama o método pai
       }
    }

     public static void Main()
    {
        string comida;
        Pessoa teste = new Pessoa();
        Console.Write("Qual comida você dará a Pessoa e Sergio? ");
        comida = Console.ReadLine();
        teste.comer(comida);
        Sergio teste3 = new Sergio();
        teste3.comer(comida);
        Console.ReadLine();
    }
}
```

3 – Overload em C#

Para fazer uma sobrecarga de métodos, basta utilizar a palavra virtual e criar os métodos no mesmo objeto, lembrando que deverão ter argumentos diferentes.

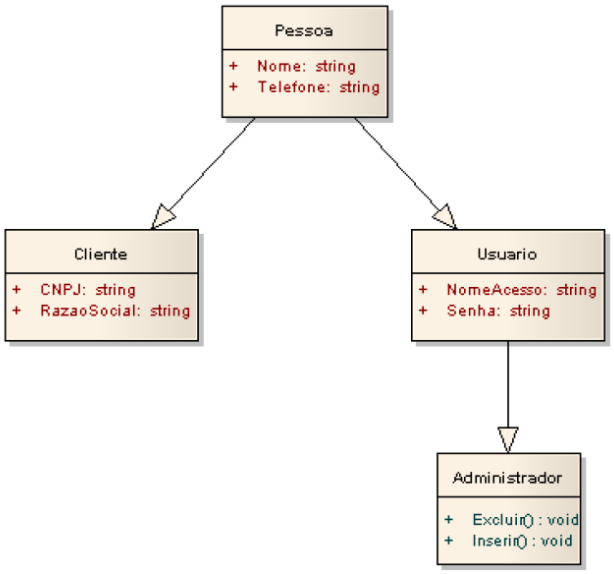
O método chamado será o que tiver os argumentos correspondentes.

Veja o exemplo 04Overload

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

public class Teste
{
    public class Pessoa
    {
        public string nome { set; get; }
        public int idade { set; get; }

        public virtual void comer()
        {
            Console.WriteLine("A Pessoa comendo.");
        }

        public  virtual void comer(string alimento)
        {
            Console.WriteLine("A Pessoa comendo {0}",alimento);
        }
    }



     public static void Main()
    {
        string comida;
        Pessoa teste = new Pessoa();
        Console.Write("Qual comida você dará a Pessoa? ");
        comida = Console.ReadLine();
        teste.comer();
        teste.comer(comida);

        Console.ReadLine();
    }
}
```

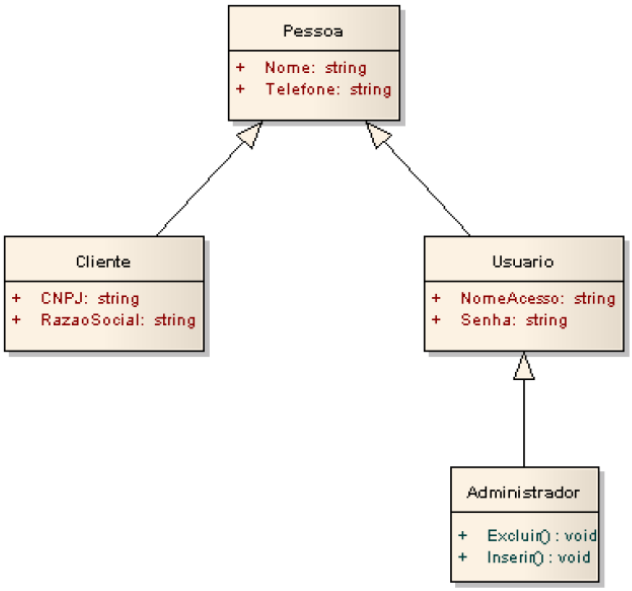
4 – Generalisação

Generalização Generalização é o ato de tornar um objeto geral, agrupar características comuns para objetos dentro de um mesmo contexto, abstrair. Eu tenho, por hábito e não por regra, a minha classe “Pai de Todas”(superclass) ser sempre abstrata.Veja o diagrama:
Neste caso a minha classe “Pai de Todas”(superclass) é a classe “Pessoa”




Se olharmos de cima para baixo podemos ver todos os objetos gerais, no topo temos a Pessoa, que pode ser um Cliente, um Usuário. O Usuário por sua vez pode ser um Administrador, que tem poderes especiais, o mesmo pode Excluir ou Inserir novos usuários. E se olharmos de baixo para cima? Aí teremos o próximo tópico Especialização.

**5 - Especialização**
A especialização nada mais é do que a parte que “especializa” o objeto vindo de uma Generalização, trazer características próprias para o objeto. Seria o mesmo que olhar o diagrama acima de baixo para cima.
Veja: O objeto Administrador especializa o objeto Usuario que por sua vez especializa o objeto Pessoa.




**6 - Delegates**

Quando há a necessidade de usarmos métodos, e ainda não sabemos qual método chamar, pois dependemos de algumas condições, podemos deixar o nosso código mais bonito usando um delegate. O delegate irá chamar o método definido de acordo com a nossa necessidade ou condição.
Veja exemplo: Abaixo temos um exemplo de uma aplicação Console em C#.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;


static class Program
{
//declara a assinatura do delegate.
//os métodos deverão ter a mesma assinatura
//para que possam ser utilizados pelo delegate
delegate int Calular(int a, int b);

static void Main()
{

//declara a variável do tipo delegate
Calular calc;

//uma condição qualquer, neste caso se o segundo é par ou ímpar
if (DateTime.Now.Second % 2 == 0)
//define o método dividir para ser usado no delegate se for par
calc = new Calular(Dividir);
else
//define o método somar para ser usado no delegate se for ímpar
calc = new Calular(Somar);

//chama o método
Console.WriteLine(calc.Invoke(4, 2));

Console.ReadKey();
}

//declara o método somar
private static int Somar(int a, int b)
```

```
{
return a + b;
}

private static int Dividir(int x, int y)
{
return x / y;
}
}
```

Exercícios

1 – Criar um classe chamado conta. Esta classe terá as propriedades (atributos) nro_conta e saldo e os métodos depósito (para acrescentar valor a saldo), saque (para retirar valor de saldo, desde que ele não fique negativo) e versaldo (para mostrar o valor de saldo). Depois, crie uma classe chamado conta_especial, que herda a classe conta e acrescenta a propriedade limite, que é um saldo extra acrescido ao saldo. Na classe conta_especial o método saque deve ser reescrito para permitir que haja saque, mesmo que o saldo fique negativo, até o valor de limite.

2 –Instancie um objeto do tipo conta especial e atribua os valores iniciais a esta conta para testar os métodos escritos anteriormente no exercício 1, criando um menu que permita escolher as opções de saque, depósito ou sair. Mostre os dados lidos e permita fazer saques e depósitos.