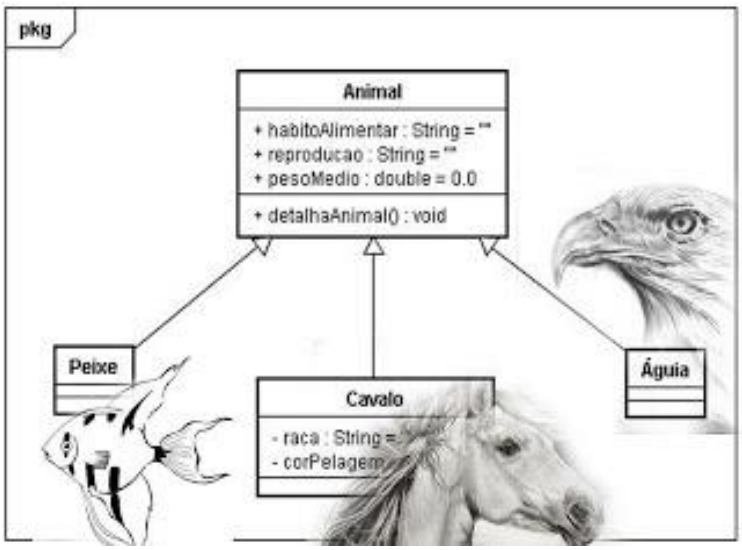

Herança de classes
em C#
Microsoft
Visual Studio

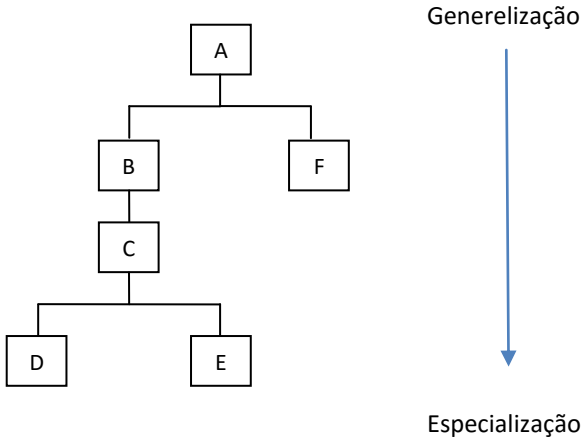

pkg
Animal
+ habitoAlimentar : String = ""
+ reproducao : String = ""
+ pesoMedio : double = 0.0
+ detalhaAnimal() : void
Peixe
Cavalo
- raca : String =
- corPelagem
Águia

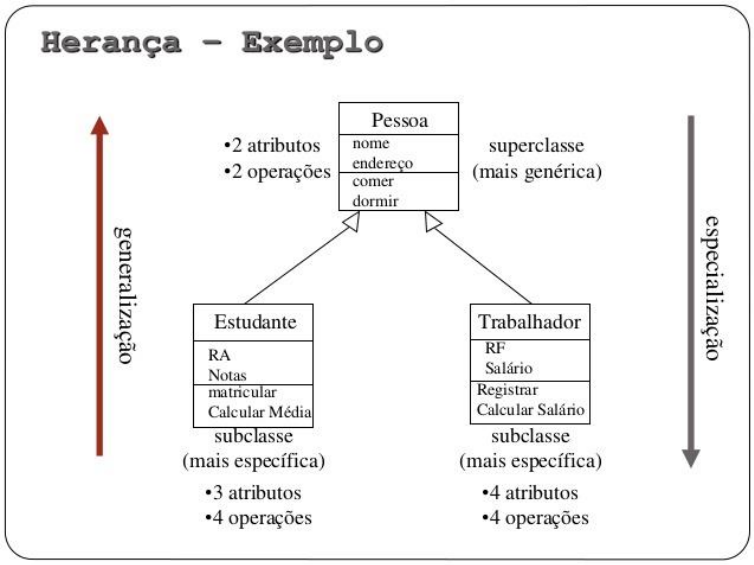
Classes - Herança

As Classes são **estruturadas hierarquicamente** em **superclasses**, **classes** e **subclasses** **por recurso a herança.**

As classes herdam as propriedades e os métodos das superclasses de que derivam, acrescentam mais propriedades e métodos e passam tudo as subclasses que delas derivam.

Esta hierarquia de classes é representada graficamente por árvores. Uma vez que uma subclasse é uma especialização da superclasse de que deriva, as árvores de classes mostram os diferentes níveis de generalização.



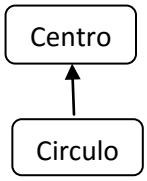


O C# não permite que uma subclasse, mas admite que as classes e subclasses partilhem métodos através da implementação de múltiplas interfaces (assunto a ver mais para frente).

A Herança como veículo transmissor da especialização de subclasses é reforçada pelo polimorfismo. O polimorfismo é a transposição para a programação de polimorfia, que é a propriedade das substâncias que se apresentam em várias formas. O polimorfismo permite ao programador definir métodos genéricos em classes ou interfaces, que podem ser implementados de diversas formas nas respectivas subclasses. O método pode, portanto, oferecer funcionalidades diferentes para cada classe que o implementa.

Derivação de uma subclasse

Uma subclasse deriva de uma classe, designada por **classe base**, herdando todas as propriedades e métodos desta.




```
public class Circulo: Centro
{
  // variáveis de instante
  // variáveis estáticas
  // construtores
  // acessores
  // métodos
}
```

A classe Circulo deriva da classe Centro.

Construtores de uma subclasse

Os construtores de uma subclasse constroem os objectos, colocando as suas propriedades – as especializadas definidas nessa subclasse e as genéricas que herdam da classe base – num estado inicial.

```
public Circulo(int x, int y, double R):base(x,y)
{
     raio = R;
}
// Instanciação de um objeto círculo com a variável de instante raio, definida
// pela subclasse Círculo e duas variáveis de instante herdadas da classe base.
```

Variáveis de instante das classes e subclasses

As variáveis de instante e as estáticas da classe base podem ser acedidas por métodos das subclasses se forem definidas com o atributo ***protected***

```
public class Pai
{…
    protected string NomePai;      // variável de instante NomePai é definida
                                   // como protected para poder ser manipuladas
                                   // pelas respetivas subclasses.
  …
}
```

Invocação de métodos da subclasse e da classe base

Os métodos de instante da classe base ou da subclasse são invocados para os respectivos objectos.