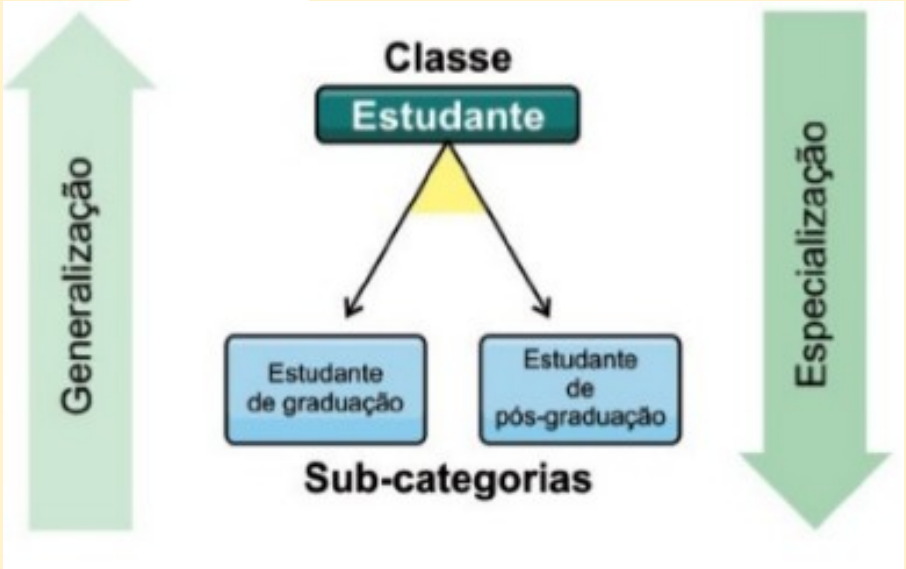
Herança: Especialização e Generalização



Curso: Sistemas de Informação
Disciplina: POO

Prof. Fábio L. R. Cordeiro

SUMÁRIO

- → Conceitos
- → Herança
- → Generalização
- → Especialização

Conceito

Herança

Herança é a capacidade de uma **subclasse** (filho) acessar as propriedades da **superclasse** (mãe) a ela relacionada propagando assim seus atributos e métodos de cima para baixo.

Neste caso, é dito que uma subclasse herda as propriedades e métodos da sua superclasse.

A relação de Herança entre duas classes configura-se da seguinte forma:

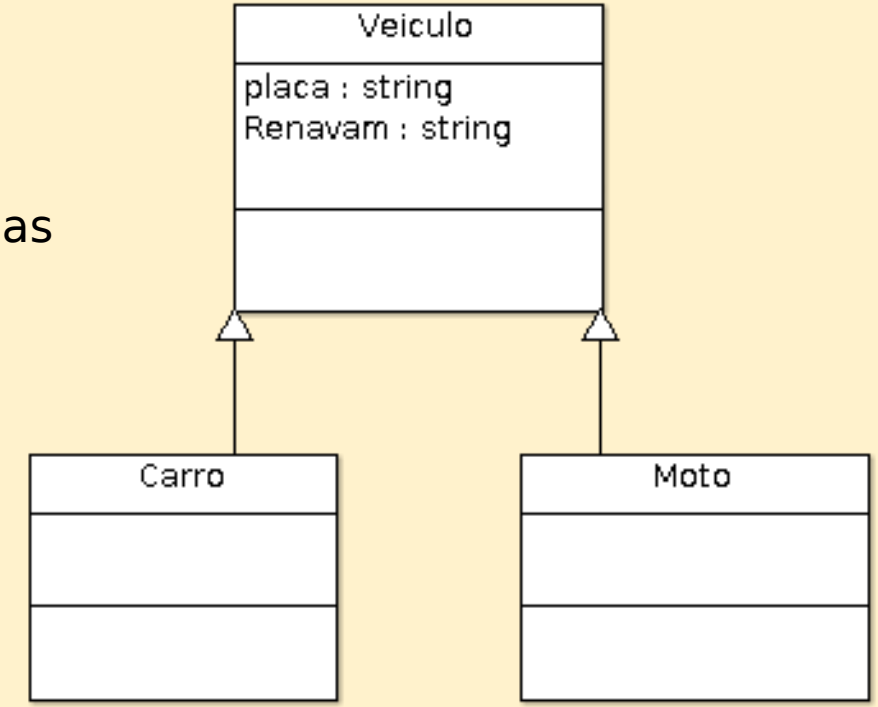
ClasseA “é um tipo de” ClasseB

Exemplo

Herança

As sub-classes Veículo e Moto **“são um tipo de”** da classe-base Veículo:

Os atributos Placa e Renavam
São herdados pelas classes filhas



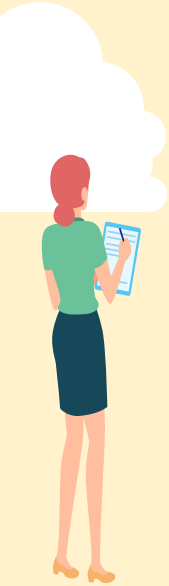
Herança

Construtor de Classes Derivadas

Ao instanciar um objeto de uma classe derivada inicia-se uma chamada a uma cadeia de construtores em uma execução Top-Down.

Antes de realizar suas tarefas o construtor da classe derivada chama o construtor da classe base.

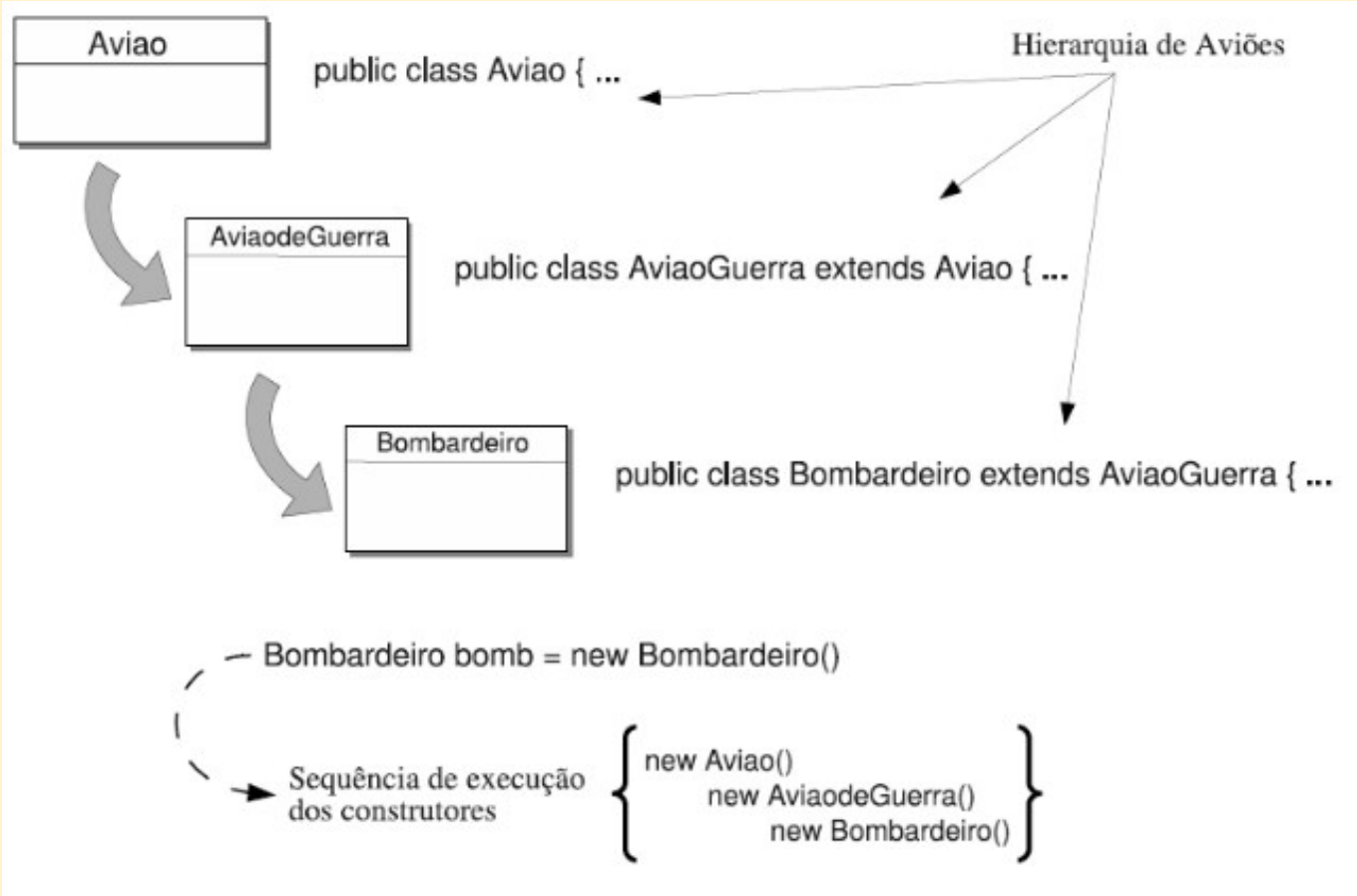
Sendo assim, quando criamos um objeto da classe derivada estamos criando também um objeto da classe base.



Exemplo

Herança

Exemplo de Derivação Top-Down de Construtores



Herança

Generalização/Especialização

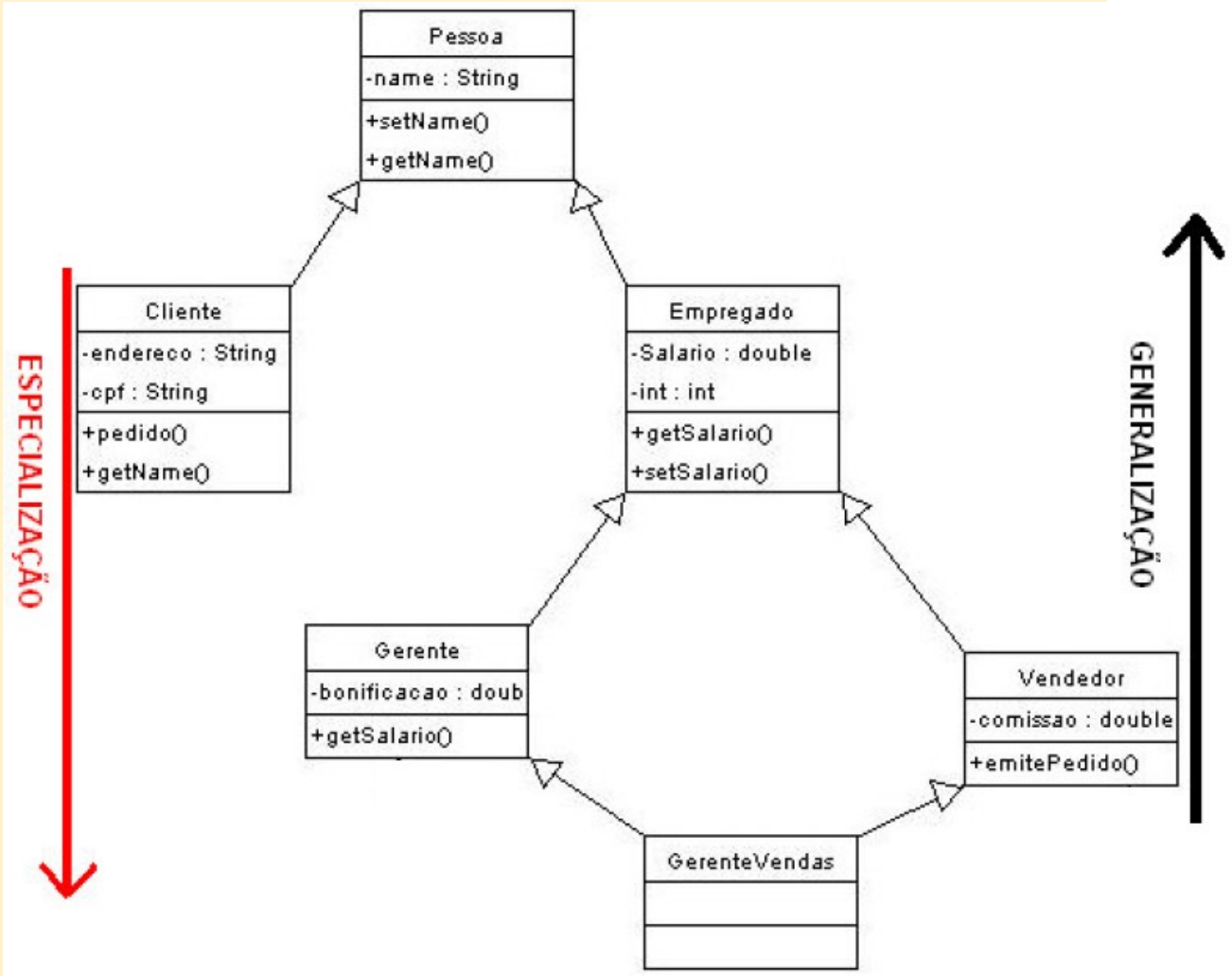
A **generalização** consiste em obter similaridade entre várias classes e a partir destas características em comum, novas classes são definidas, as **superclasses.**

A **especialização** por sua vez consiste em observar diferenças entre os objetos de uma mesma classe e dessa forma novas classes são criadas, as **subclasses**

Exemplo

Generalização/ Especialização

Exemplo Diagrama de Classe: Pessoa



Conceito

Modificadores de Acesso

+ public : um método definido como public pode ser acessado por qualquer classe de qualquer projeto

- private : é mais restritivo, somente a classe onde ele foi definido é que pode acessá-lo, nenhuma outra tem permissão, nem mesmo classes que herdam da classe onde o método foi definido

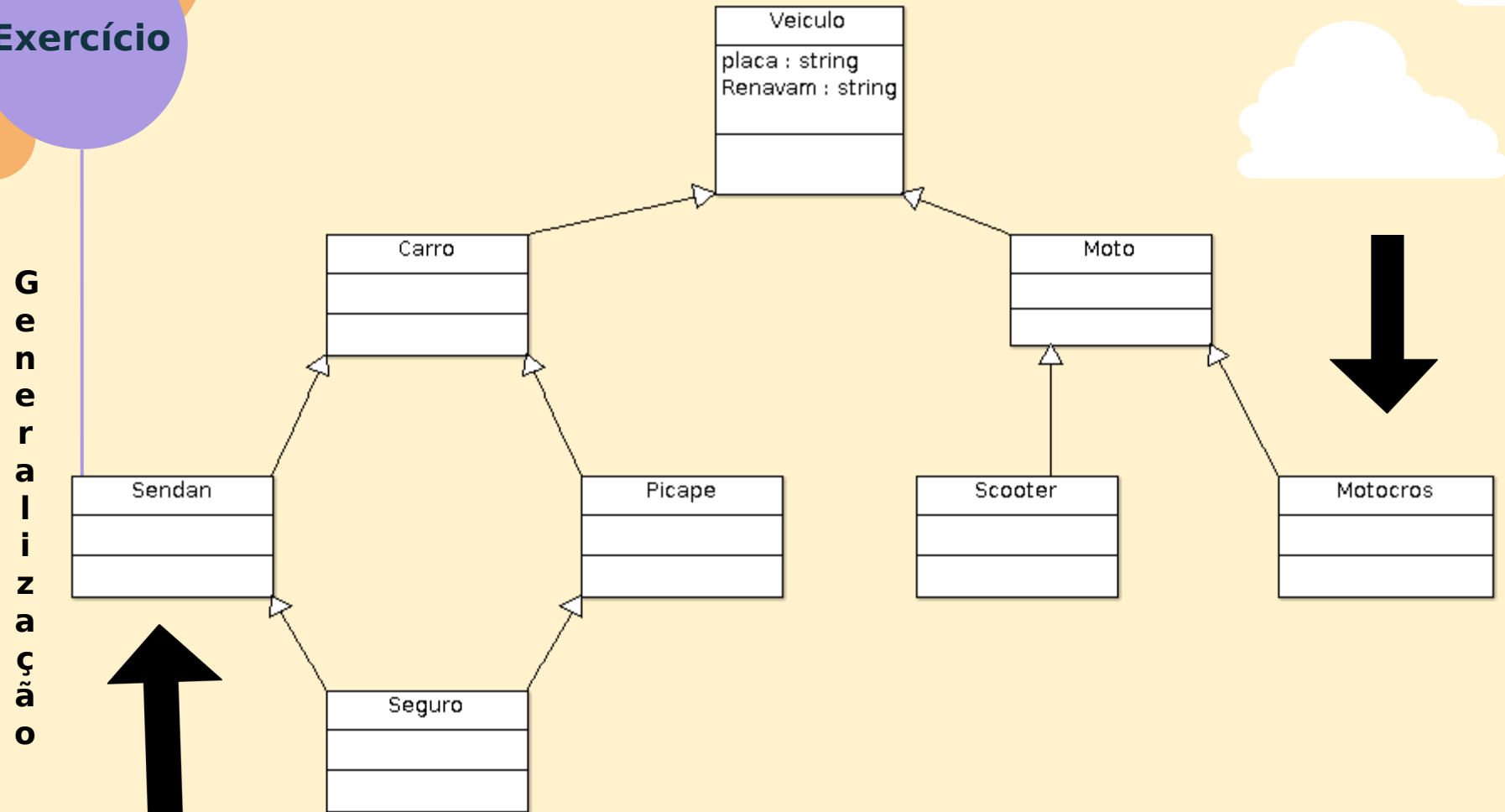
 # protected : somente as classes que herdam da classe que contem o método protegido tem tem permissão para acessá-lo e as classes que estão no mesmo pacote.

Existem outros modificadores de acesso em C#:

- Internal: acesso limitado ao assembly atual (*componente*)
- protected internal: ao assembly atual e aos seus componentes derivados

Exemplo Diagrama de Classe

Exercício



Exercício

Exercício: Veículo

1) Implemente o diagrama de Classes do Sistema de Veículos visto anteriormente:

a) Pacotes: crie um pacote para cada tipo de SubClasse:

Ex: **com.aluno.veiculos.entidades.carros**

b) Controle os modificadores permitindo acesso de maneira mais restritiva funcional possível:

ex: Atributos **-private** e métodos **#protected** e **+public** de acordo com necessidade.

c) Implemente o método mostrar( ) na Classe Veiculo que deverá ser sobrescrito nas classes filhas:

Veiculo (1 nível) : ex: **public virtual void mostrar( ) { ...};**

Carro (2 nível) : ex: **public override void mostrar( ) { ... };**

Sedan (3 nível): ex: **public override void mostrar( ) {...}**

d) Crie um pacote para a classes de serviços, dentro de entidades e crie uma “classe” **Seguro** que tenha um método apenas com assinatura, herde e sobrescreva este método nas classes Carros:

Ex: nome do pacote: **com.aluno.veiculos.entidades.servicos**

Ex Interface: **public interface Seguro { public void seguroParticular( ); }**

e) Faça um esquema de controle de cadastro de veículos na **Main**, testando cada um dos tipos como pelo menos 2 objetos instanciados de cada.