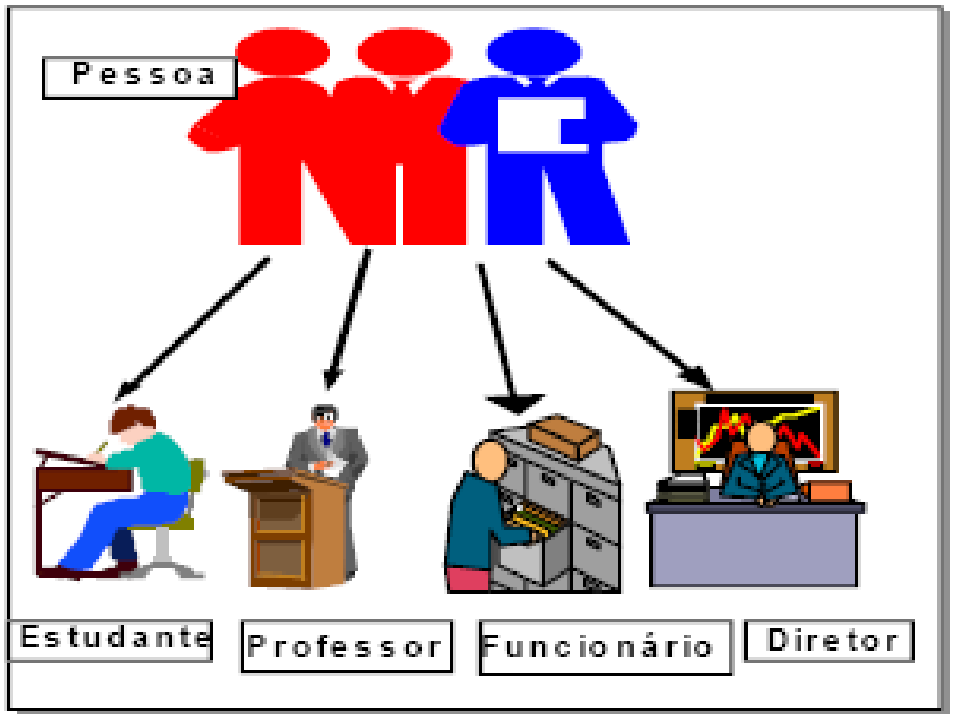
Programação Orientada à Objetos

Aula 11
Herança
Polimorfismo

Henrique Poyatos
henrique.poyatos@bandtec.com.br

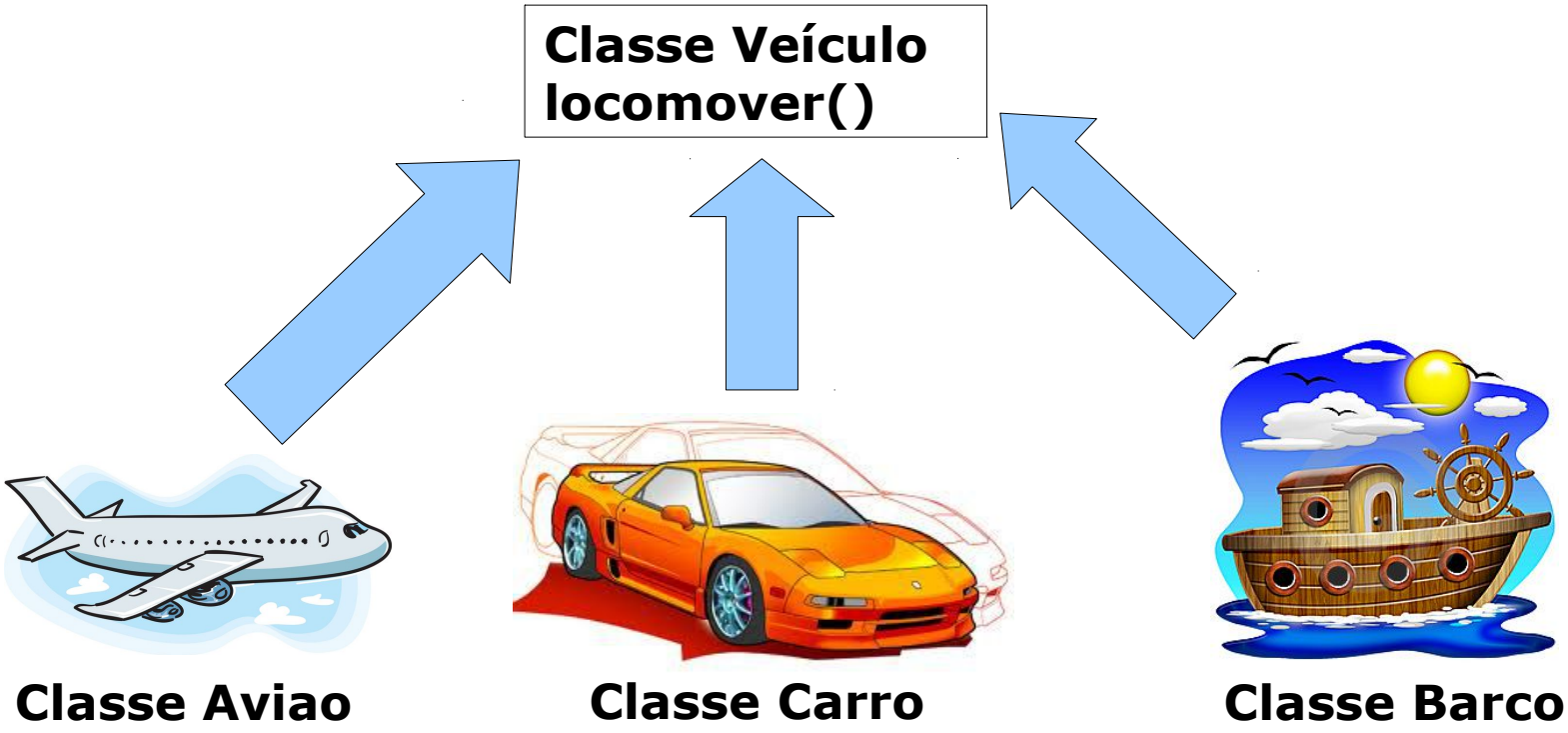
Herança

- É o mecanismo para expressar a similaridade entre Classes, simplificando a definição de classes iguais que já foram definidas.

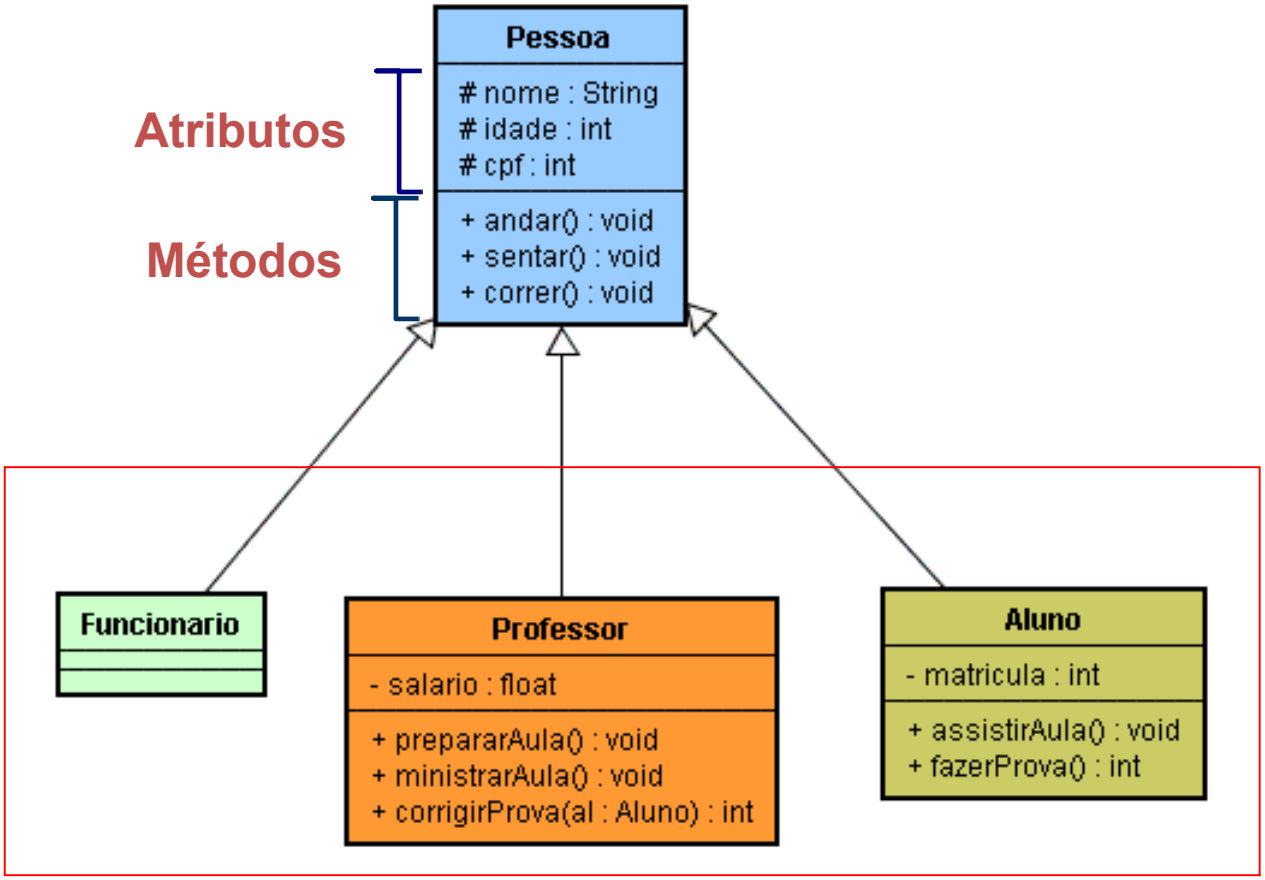


Generalização

- Generalizar significa pegar duas ou mais classes e pensar em características (atributos) e procedimentos (métodos) que estas classes têm em comum, concentrando estes itens comuns em um classe abstrata. (ABSTRAIR A CLASSE).



Herança
Exemplo Pessoa x Aluno x Professor



Possui os atributos e métodos da classe Herdada (Pessoa)

Herança
Exemplo Veiculo x Carro

```
public class Veiculo
{
    public string locomover();
}
public class Carro : Veiculo
{
}
public class Aviao : Veiculo
{
}
```

Classe Veiculo → A Classe "pai".

Declaração ":" → Acompanhada pelo nome da classe "pai", indica que esta classe está extendendo ou herdando todas as características (atributos e métodos) da classe pai "Veiculo".

Polimorfismo

- Polimorfismo é muitas vezes referida como o terceiro pilar da programação orientada a objetos, depois de encapsulamento e herança. Polimorfismo é uma palavra grega que significa "muitos em forma de" e tem dois aspectos distintos:
  - Em tempo de execução, os objetos de uma classe derivada podem ser tratados como objetos de uma classe base, em lugares como parâmetros do método e coleções ou arrays. Quando isso ocorre, tipo declarado do objeto não é mais idêntico ao seu tipo em tempo de execução.
  - Classes de base pode definir e implementar métodos virtuais, e as classes derivadas podem substituí-los, o que significa que eles fornecem sua própria definição e implementação. Em tempo de execução, quando o código do cliente chama o método, o CLR olha o tipo de tempo de execução do objeto, e invoca o override do método virtual. Assim, em seu código-fonte, você pode chamar um método em uma classe base, e fazer com que a versão de uma classe derivada do método a ser executado.

Polimorfismo
Exemplo Veiculo x Carro

```
public class Veiculo
{
      public virtual string locomover()
      {
            Console.WriteLine("Veículo em movimento");
      }
}
public class Carro : Veiculo
{
      public override string locomover()
      {
            Console.WriteLine("Carro em movimento");
      }
}
```

Método Redeclarado → O método original é substituído por um novo.

Polimorfismo – accessor base

Exemplo Veiculo x Carro

```
public class Veiculo
{
      public virtual string locomover()
      {
            Console.WriteLine("Veículo em movimento");
      }
}
public class Carro : Veiculo
{
      public override string locomover()
      {
            base.locomover();
            Console.WriteLine("Carro em movimento");
      }
}
```

Accessor base → Pode ser usado para chamar o método original (pai).