

DOMAIN DRIVEN DESIGN

Prof. Rafael Desiderio

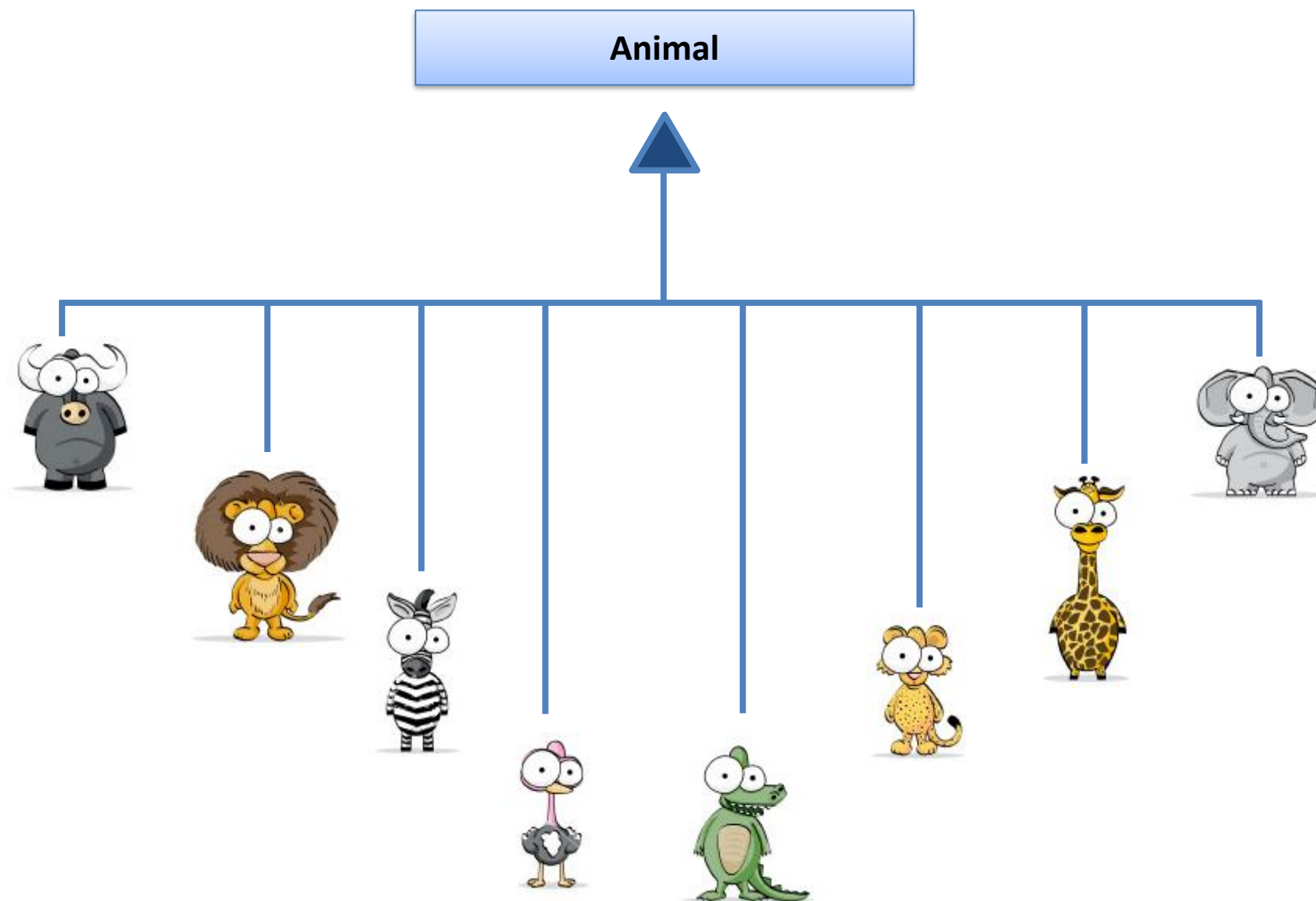
07 – HERANÇA

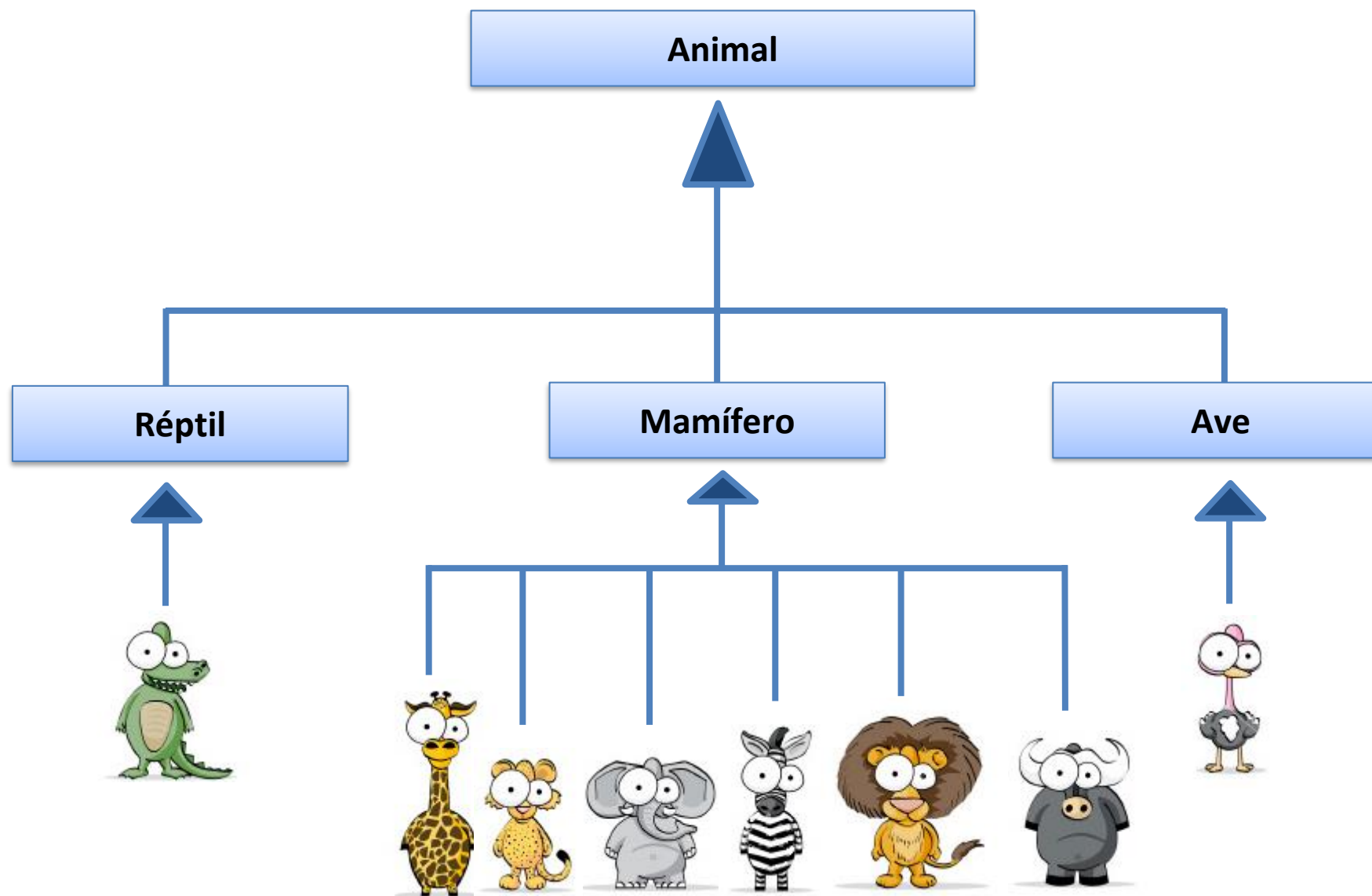
| Herança

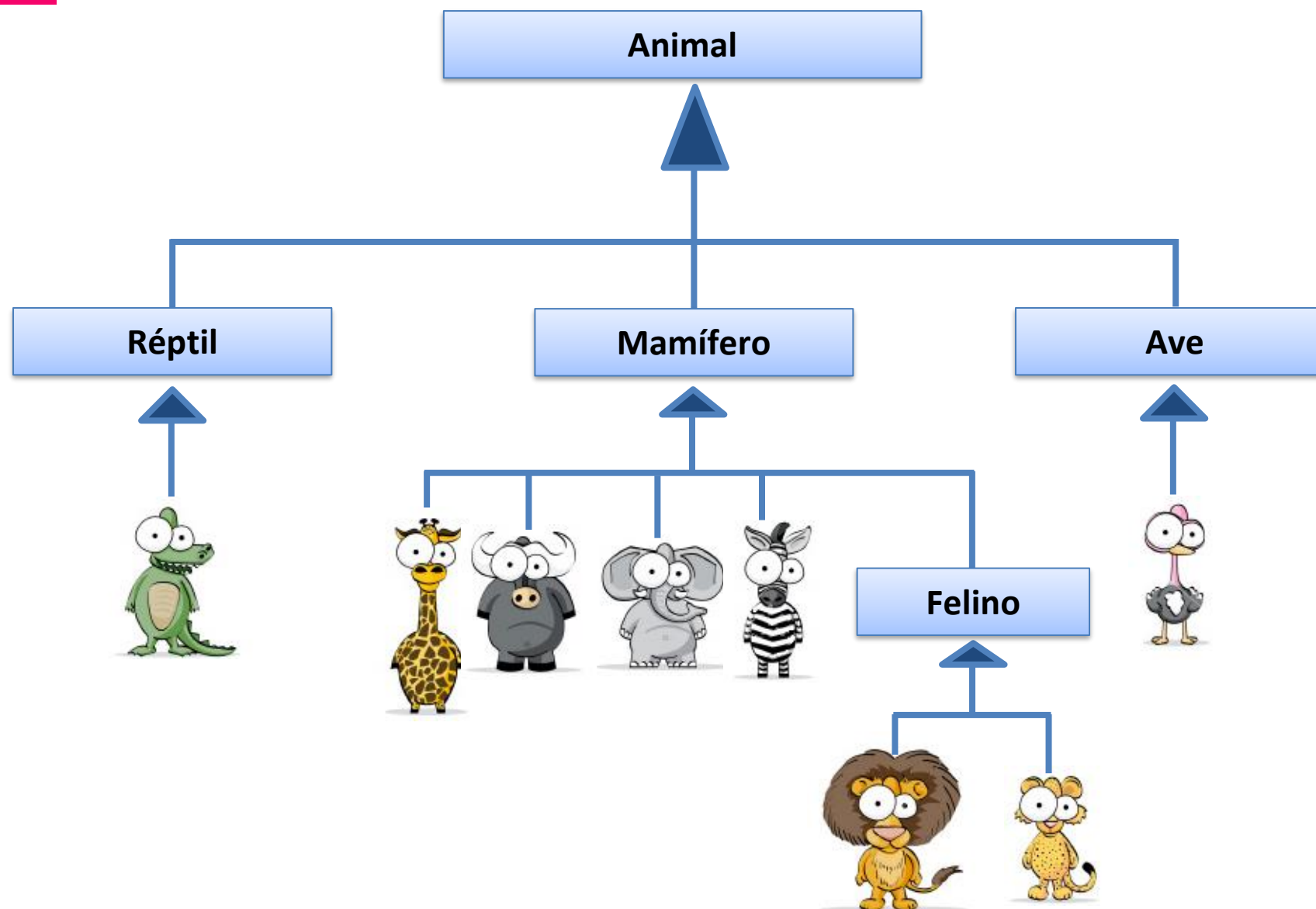
- **Herança** é um dos mecanismos **fundamentais** para as **linguagens** que suportam o **paradigma OO**;
- Este mecanismo possibilita a **criação de novas classes a partir de uma já existente**;
- A **herança** é utilizada como forma de **reutilizar os atributos e métodos** de classes já definidas, permitindo assim derivar uma **nova classe mais especializada** a partir de outra classe mais genérica existente;
- **Aplicar herança** sempre envolve basicamente dois elementos: uma **superclasse** (classe pai) e uma **subclasse** (classe filha);
- **Superclasse** é também conhecida como classe ancestral ou **classe pai**. Apresenta as características genéricas de um conjunto de objetos;
- **Subclasse** é também conhecida como classe descendente ou **classe filha**. Elas estende a superclasse para incluir suas características

A subclasse:

- **Herda os atributos;**
- Permite **adicionar novos atributos** (que será visível somente na subclasse);
- Em relação aos **métodos**, a subclasse poderá **utilizá-los/herdá-los** (superclasse), bem como **criar novos** métodos e alterá-los;
- Métodos **construtores não são herdados** (porém podemos chamá-los dentro do construtor da subclasse).

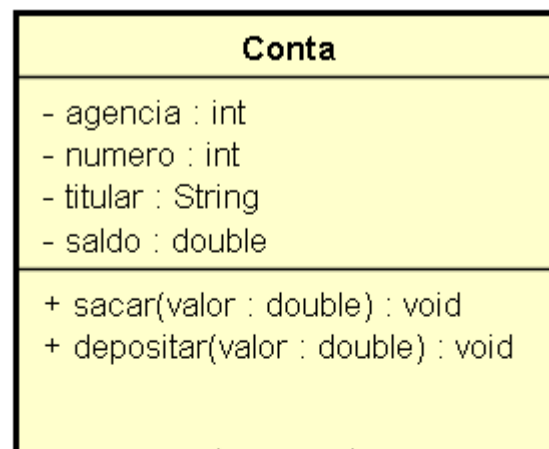






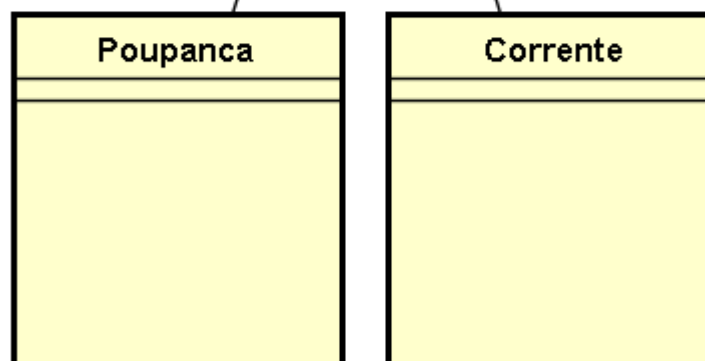
| Herança

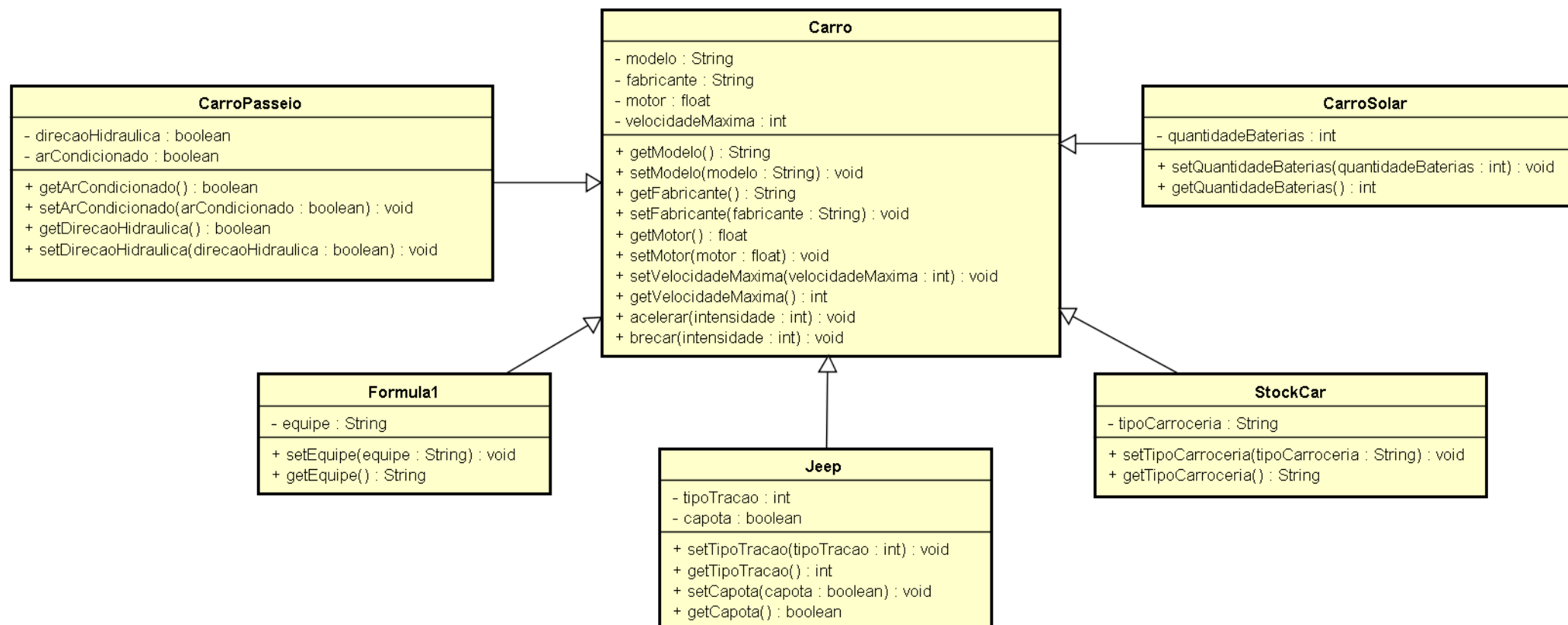
Superclasse
(classe pai)



Relacionamento de Generalização
(Herança)

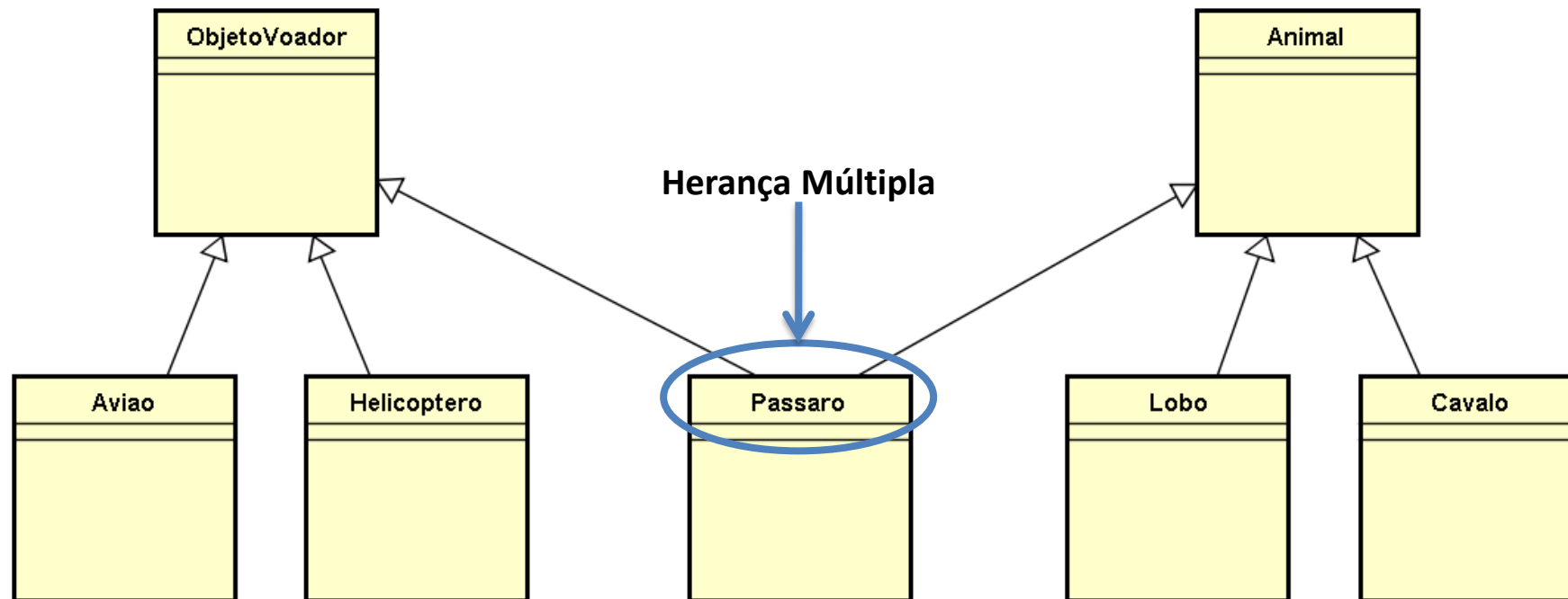
Subclasse
(classe filha)





| Herança Múltipla

- **Herança múltipla** significa que uma classe pode **herdar** de **várias outras classes**;
- O **Java não** permite a **herança múltipla**;



| Herança no Java – extends

- Toda **classe criada no Java** é estendida a partir da classe **Object**;
- A palavra-chave **extends** é utilizada na declaração de uma classe para especificar quem é sua superclasse;
- Caso a palavra-chave seja omitida, a classe **Object** será assumida como a **superclasse** da nova classe;

- Sintaxe:

```
[public] [abstract | final] class <subclasse> extends <superclasse>
```

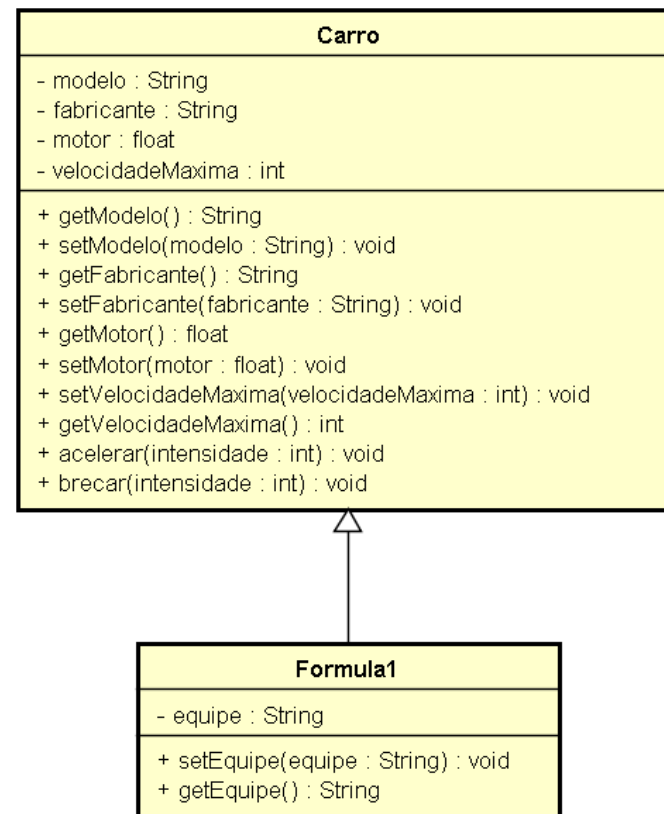
```
{
```

```
}
```

| Herança no Java – extends

- No exemplo abaixo, a classe **Formula1** estende a classe **Carro**:

```
public class Formula1 extends Carro {  
  
    private String equipe;  
  
    public void setEquipe(String equipe) {  
        this.equipe = equipe;  
    }  
  
    public String getEquipe() {  
        return equipe;  
    }  
  
}
```



| Herança no Java – super ()

- O **construtor** da **classe estendida** lida apenas com as **variáveis definidas na classe**, e o construtor da superclasse lida com as variáveis que são herdadas;
- Um **construtor** da classe estendida **pode invocar diretamente um dos construtores da superclasse**;
- **Construtores não são herdados** e precisam ser implementados na subclasse;
- Construtores da subclasse **“sempre”** utilizam algum construtor da superclasse;
- A referência **super(<parâmetros>)** é utilizada para **invocar o construtor da superclasse**;

| Herança no Java – super ()

```
public class Conta {  
    → public Conta(String numero, double saldo, Cliente cliente){  
        }  
    }  
  
    public class Poupanca extends Conta{  
  
        public Poupanca(String numero, double saldo, Cliente cliente){  
            → super(numero, saldo, cliente);  
            System.out.println("Classe: Poupanca - Construtor:  
                Poupanca(String numero, double saldo, Cliente cliente)");  
        }  
    }  
}
```

- **super** chama o **construtor** da superclasse
 - se **super** não for **chamado**, o compilador acrescenta uma chamada ao construtor *default*: **super();**
 - se não existir um construtor *default* na superclasse, haverá um **erro de compilação**;

| Herança no Java – super ()

- Permite que **atributos** e **métodos** da **superclasse** sejam **referenciados** pelos métodos da **subclasse**;
 - Sintaxe:
 - **super.<atributo>;**
 - **super.<método>;**
- Uso sucessivo de super **não** é permitido;
 - **super.super.nomeMetodo;**
- Caso queira referir-se a um **construtor da superclasse**, a sintaxe é diferente. Deve ser utilizada apenas a **referência seguida de um par de parênteses**;
 - **super();**

| Herança no Java, Exemplo – super ()

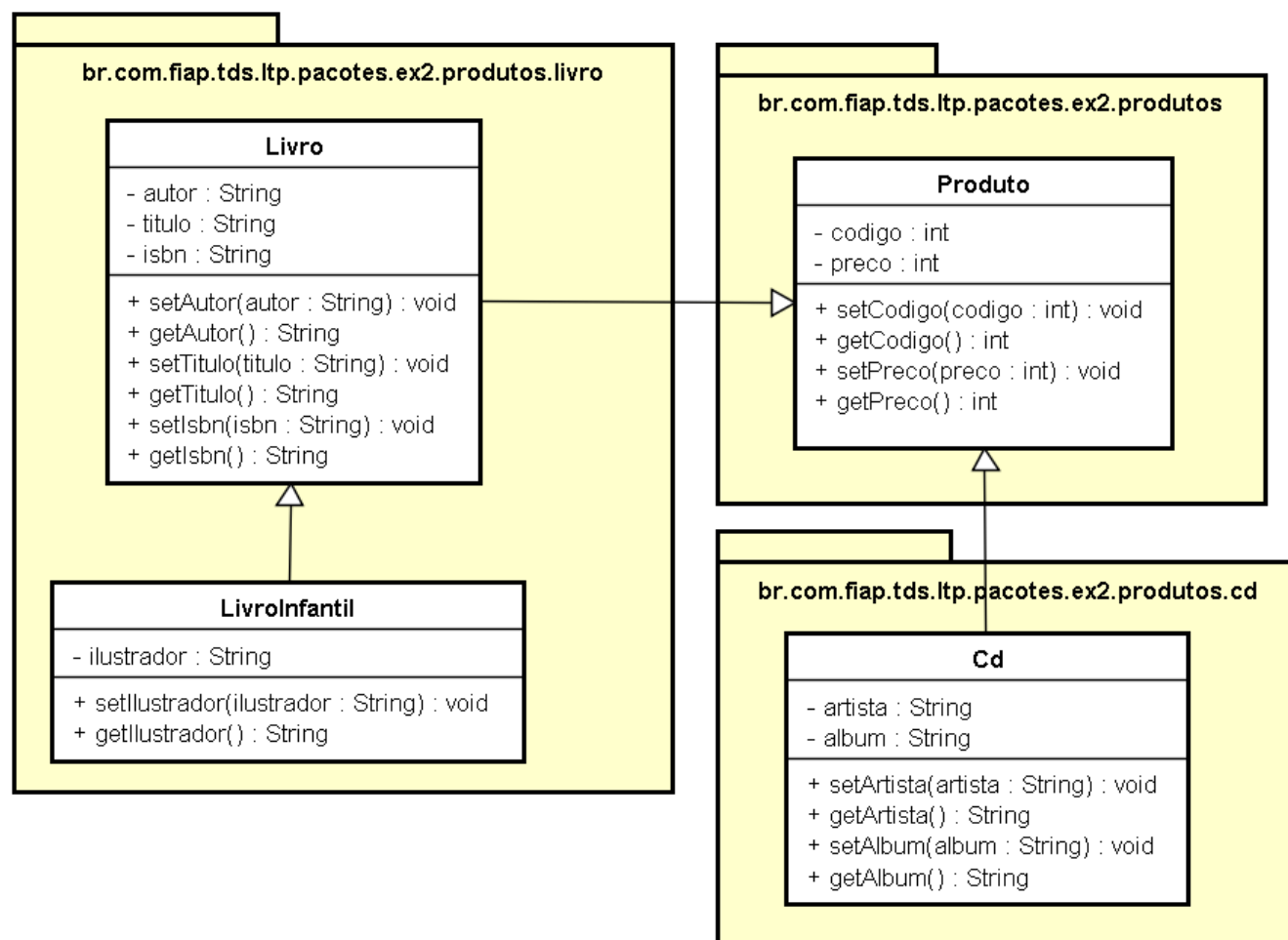
```
public class ClasseA{
    protected int atributo3 = 3;
}

public class ClasseB extends ClasseA{
    public void testeSuper(){
        super.atributo3 = 33;
        System.out.println("super.atributo3=" + this.atributo3);
    }
}

public class TesteSuper {
    public static void main(String args[]){
        ClasseB cla = new ClasseB();
        cla.testeSuper();
    }
}
```

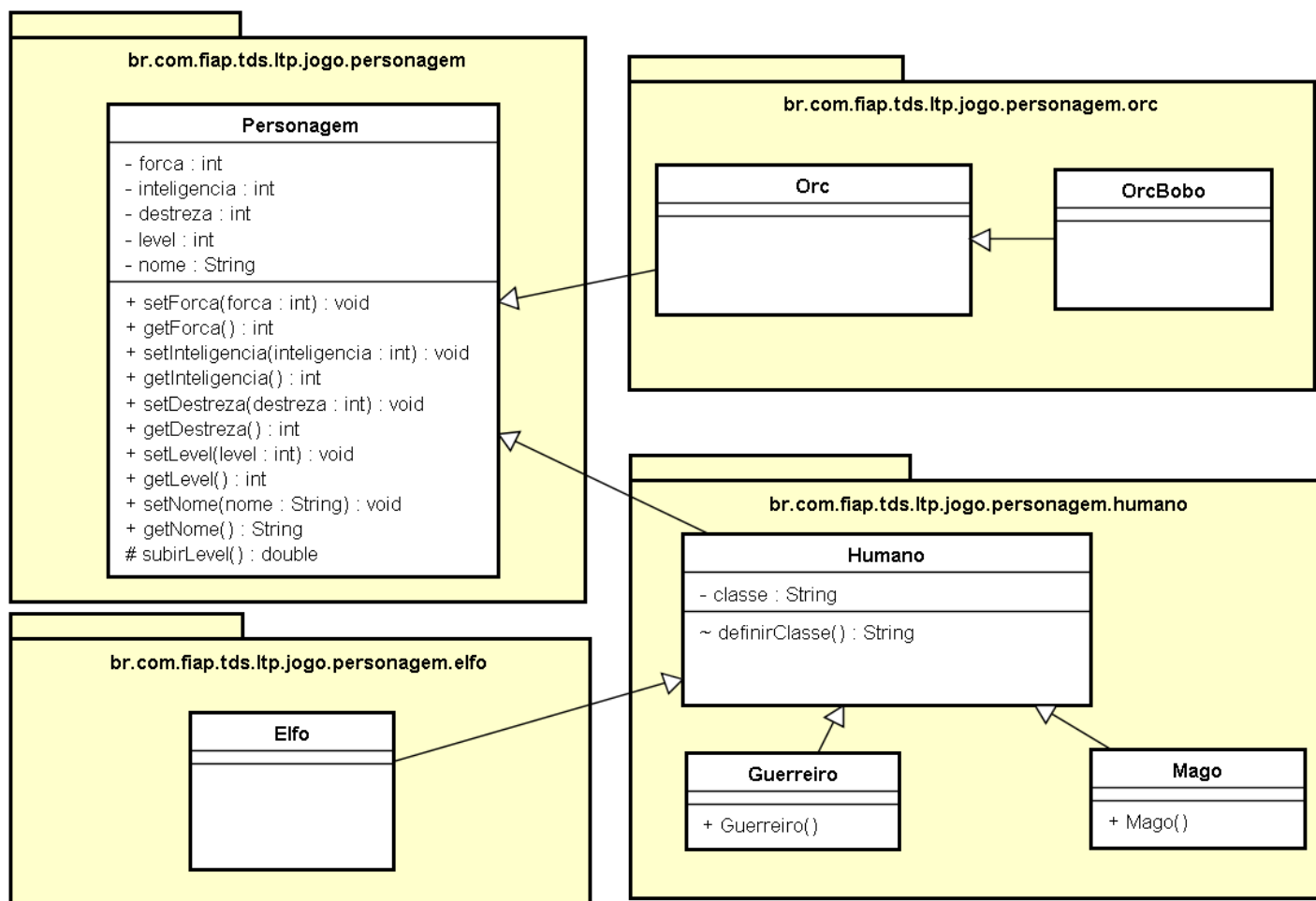
Vamos à prática!

Implemente o diagrama de classes abaixo:



Vamos à prática!

Implemente o diagrama de classes abaixo:



Copyright © 2024
Prof. Rafael Desiderio

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito do Professor (autor).