FIAP GRADUAÇÃO

DOMAIN DRIVEN DESIGN

Prof. Me. Thiago T. I. Yamamoto

#09 – HERANÇA

 *profthiagoy@fiap.com.br*  *thiagoyama*

TRAJETÓRIA

- Orientação a Objetos
- Introdução ao Java
- IDE e Tipos de Dados
- Classes, atributos e métodos
- Encapsulamento
- Construtores
- Conversões e Tomada de decisões
- Manipulação de Strings
- Herança

#09 - AGENDA

- Conceito de Herança
- Herança no Java
- Palavra reservada super

HERANÇA

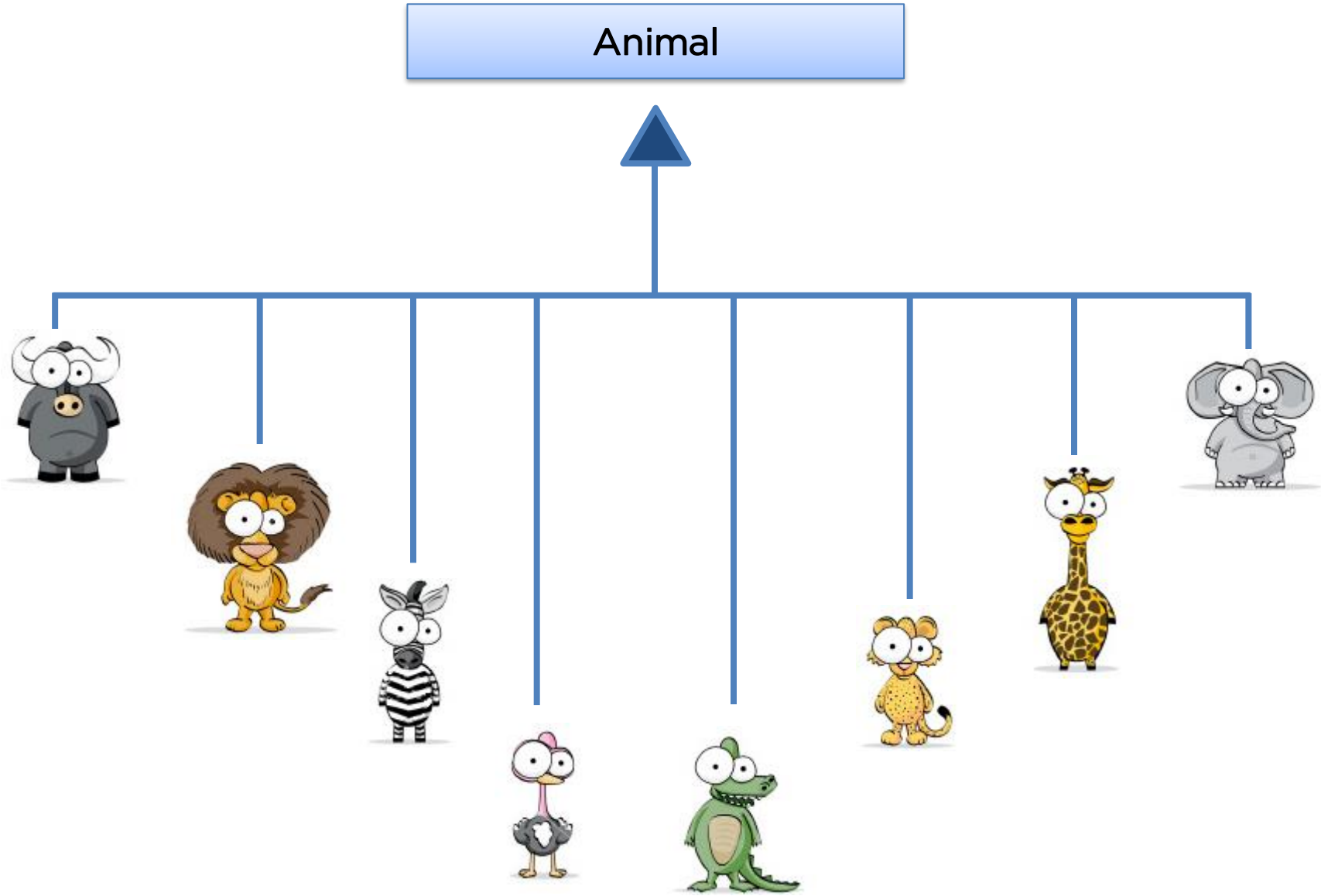
HERANÇA

- **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

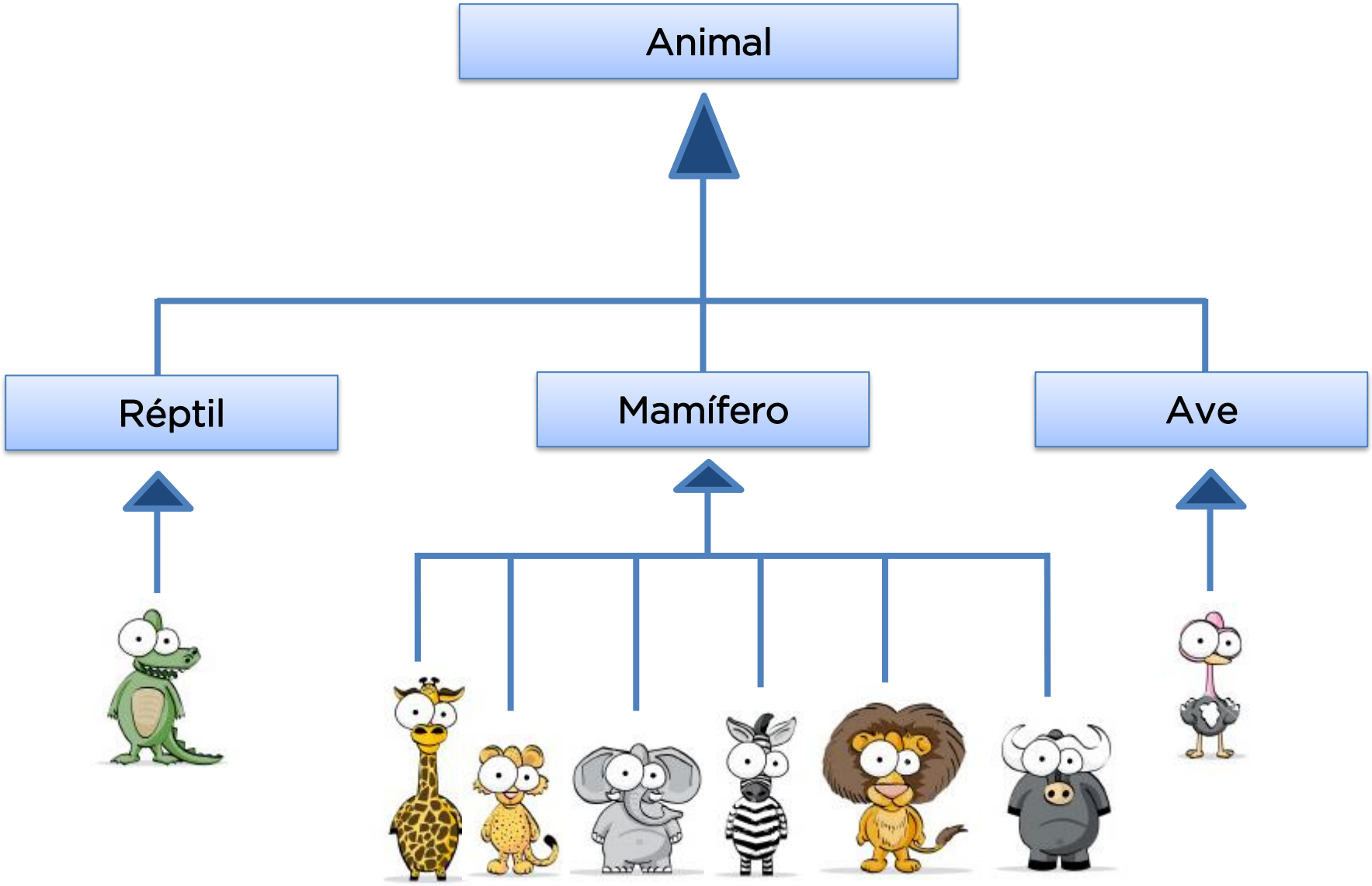
HERANÇA

- 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).

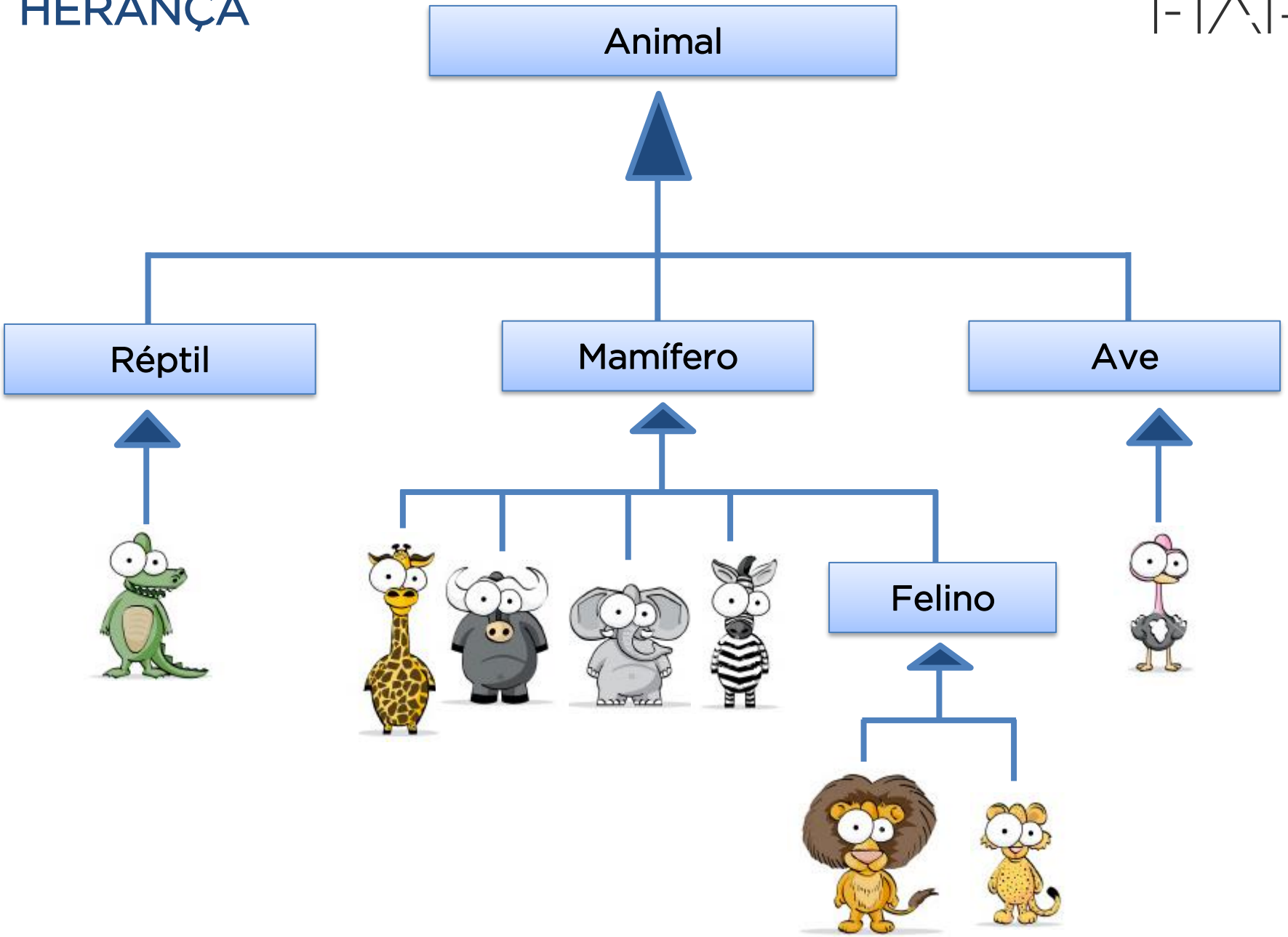
HERANÇA



HERANÇA

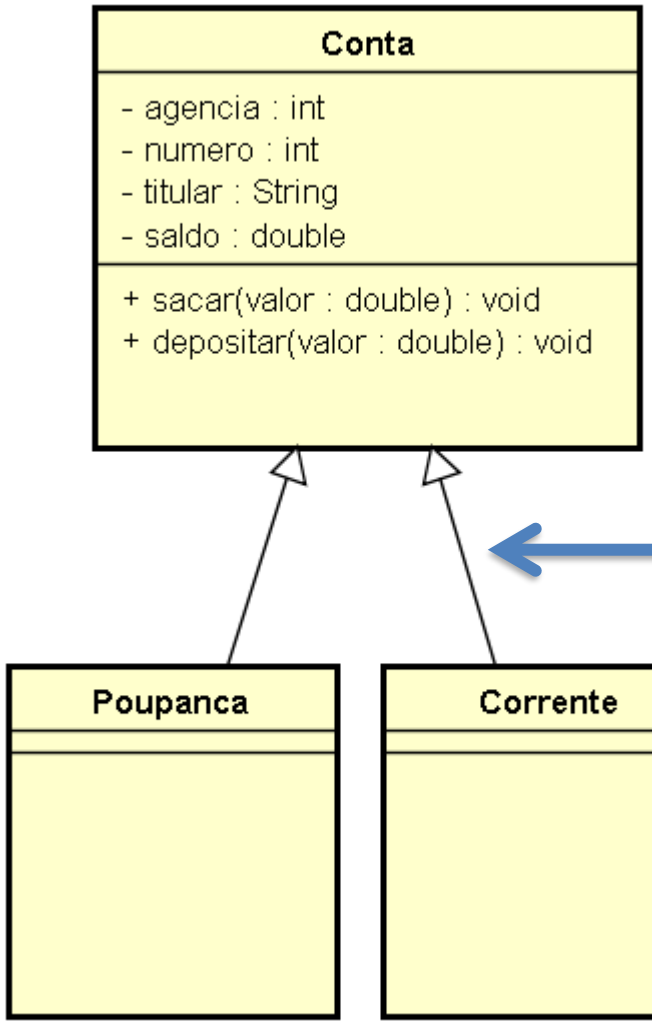


HERANÇA



HERANÇA

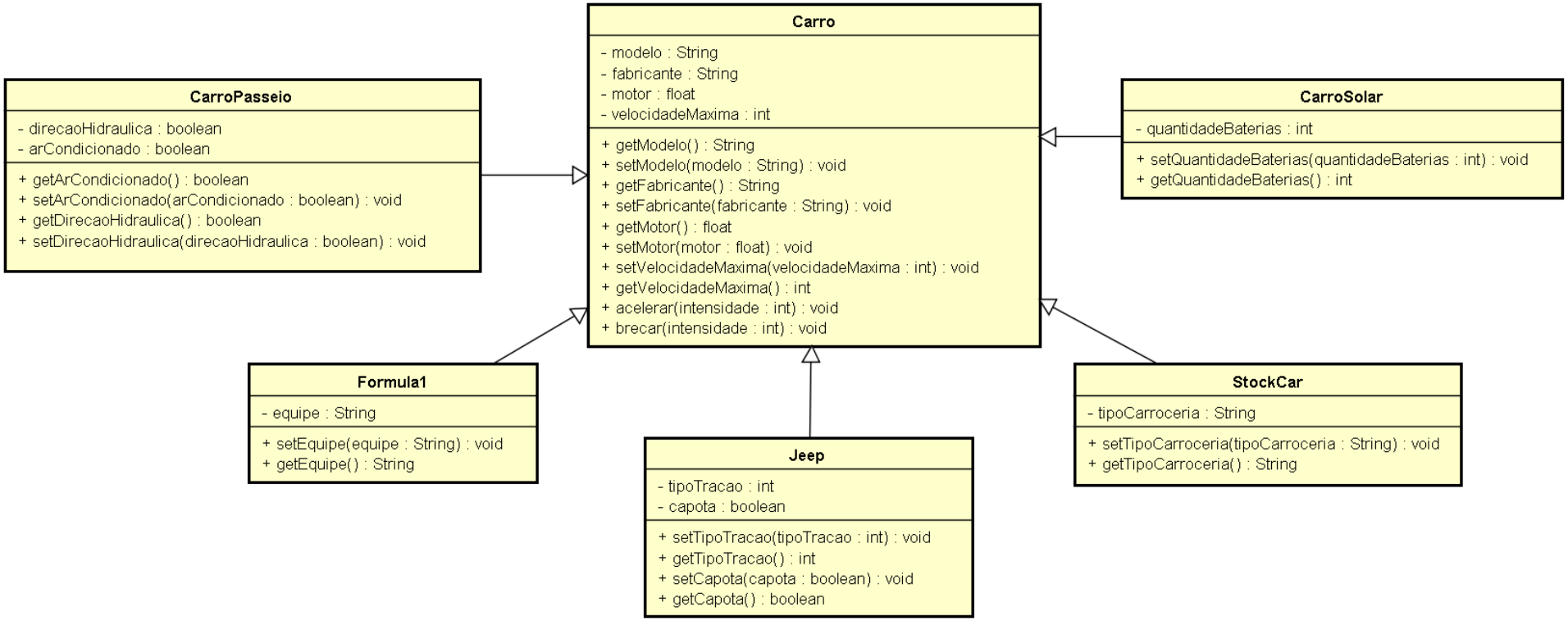
Superclasse
(classe pai)



Relacionamento de
Generalização
(Herança)

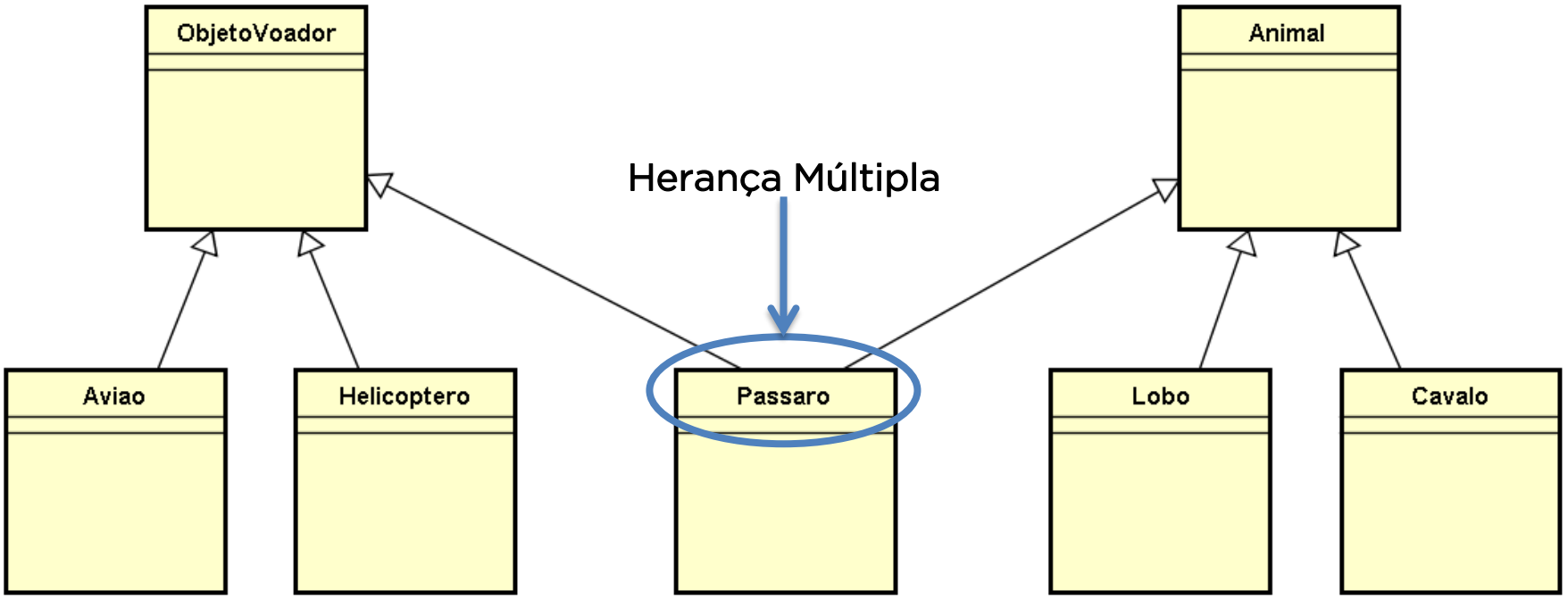
Subclasse
(classe filha)

HERANÇA


Carro
- modelo : String
- fabricante : String
- motor : float
- velocidadeMaxima : int
+ getModelo( ) : String
+ setModelo(modelo : String) : void
+ getFabricante( ) : String
+ setFabricante(fabricante : String) : void
+ getMotor( ) : float
+ setMotor(motor : float) : void
+ setVelocidadeMaxima(velocidadeMaxima : int) : void
+ getVelocidadeMaxima( ) : int
+ acelerar(intensidade : int) : void
+ brecar(intensidade : int) : void
CarroPasseio
- direcaoHidraulica : boolean
- arCondicionado : boolean
+ getArCondicionado( ) : boolean
+ setArCondicionado(arCondicionado : boolean) : void
+ getDirecaoHidraulica( ) : boolean
+ setDirecaoHidraulica(direcaoHidraulica : boolean) : void
CarroSolar
- quantidadeBaterias : int
+ setQuantidadeBaterias(quantidadeBaterias : int) : void
+ getQuantidadeBaterias( ) : int
Formula1
- equipe : String
+ setEquipe(equipe : String) : void
+ getEquipe( ) : String
Jeep
- tipoTracao : int
- capota : boolean
+ setTipoTracao(tipoTracao : int) : void
+ getTipoTracao( ) : int
+ setCapota(capota : boolean) : void
+ getCapota( ) : boolean
StockCar
- tipoCarroceria : String
+ setTipoCarroceria(tipoCarroceria : String) : void
+ getTipoCarroceria( ) : String

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**:

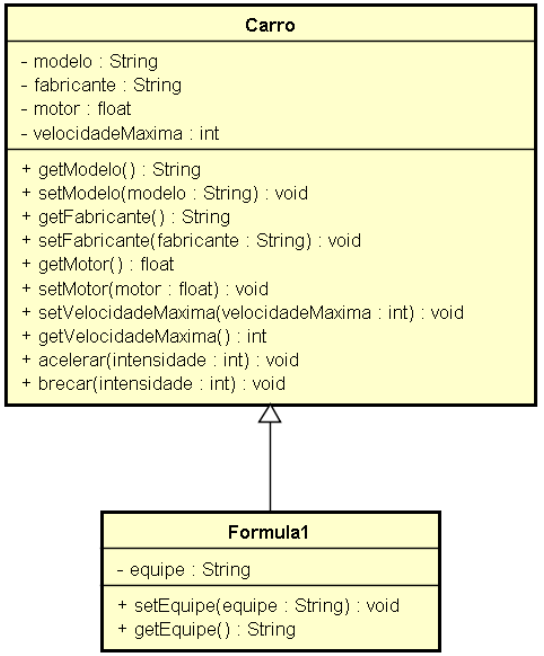
```
public class Formula1 extends Carro {

    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 – SUPER (EXEMPLO)

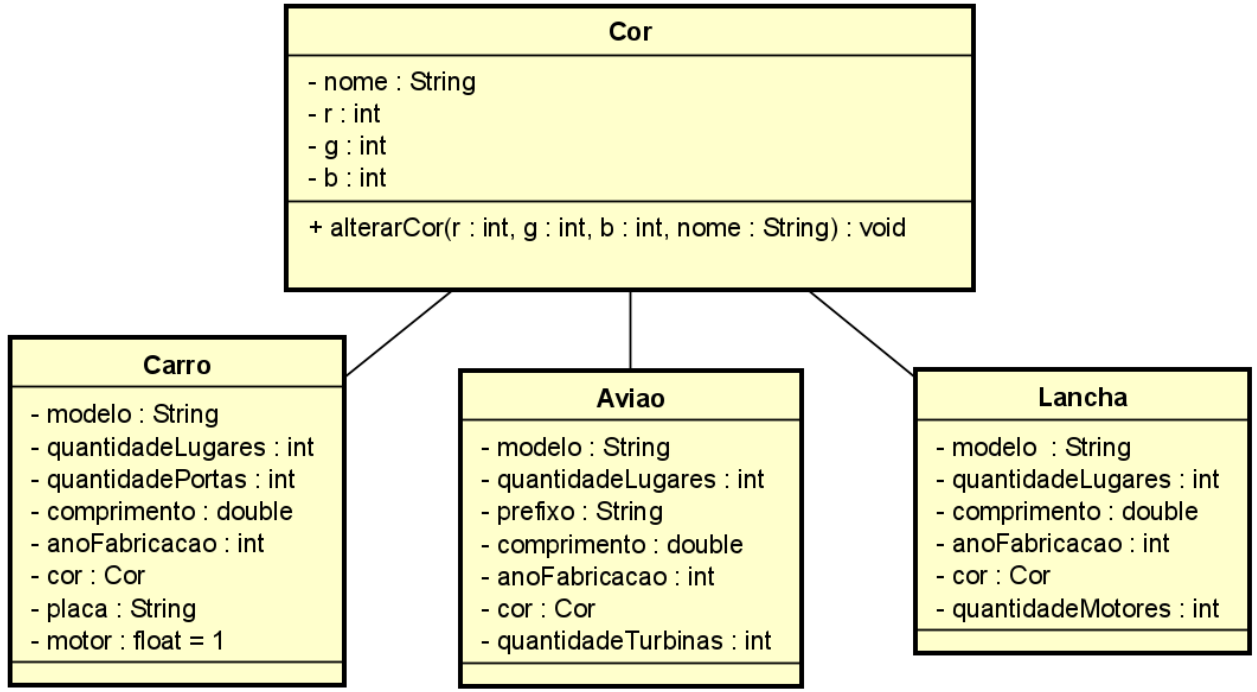
```
public class ClasseA{
      protected int atributo3 = 3;
}

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[]){
    CasseB cla = new ClasseB();
       cla.testeSuper();
  }
}
```

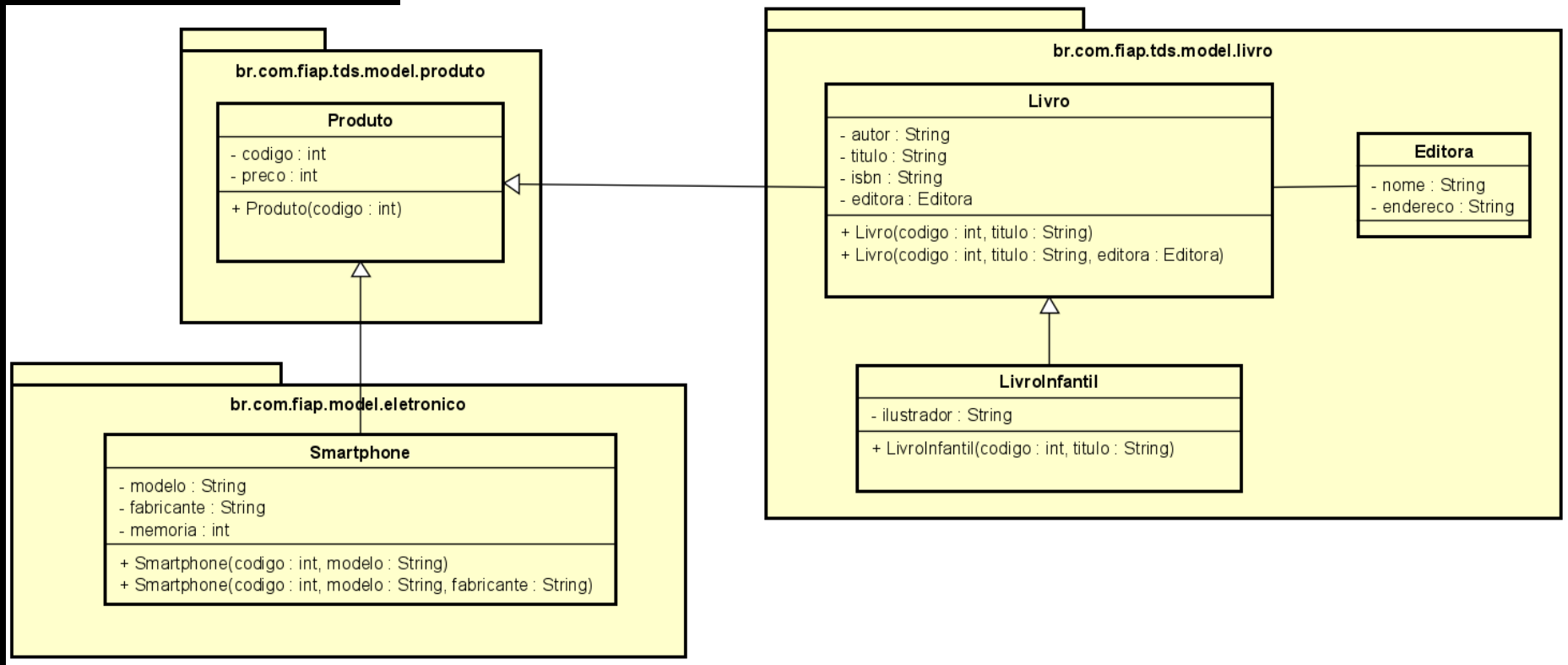
PRÁTICA 1

- Utilizando seus conhecimentos sobre herança, analise as classes exibidas abaixo e em seguida implemente uma nova hierarquia de classes de forma que o conceito de herança seja aplicado e evite as duplicações expostas.



PRÁTICA 2

- Implemente o **diagrama de classes** abaixo:



- **Crie uma classe de Teste** para instanciar um Livro com a Editora;

FIAP

"A lógica pode levar de um ponto A a um ponto B.
A imaginação pode levar a qualquer lugar"
Albert Einstein