

UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL

AULA 05

LINGUAGEM DE PROGRAMAÇÃO 2
JAVA



PROF. JANIHERYSON FELIPE

CONTEÚDO DESSA AULA

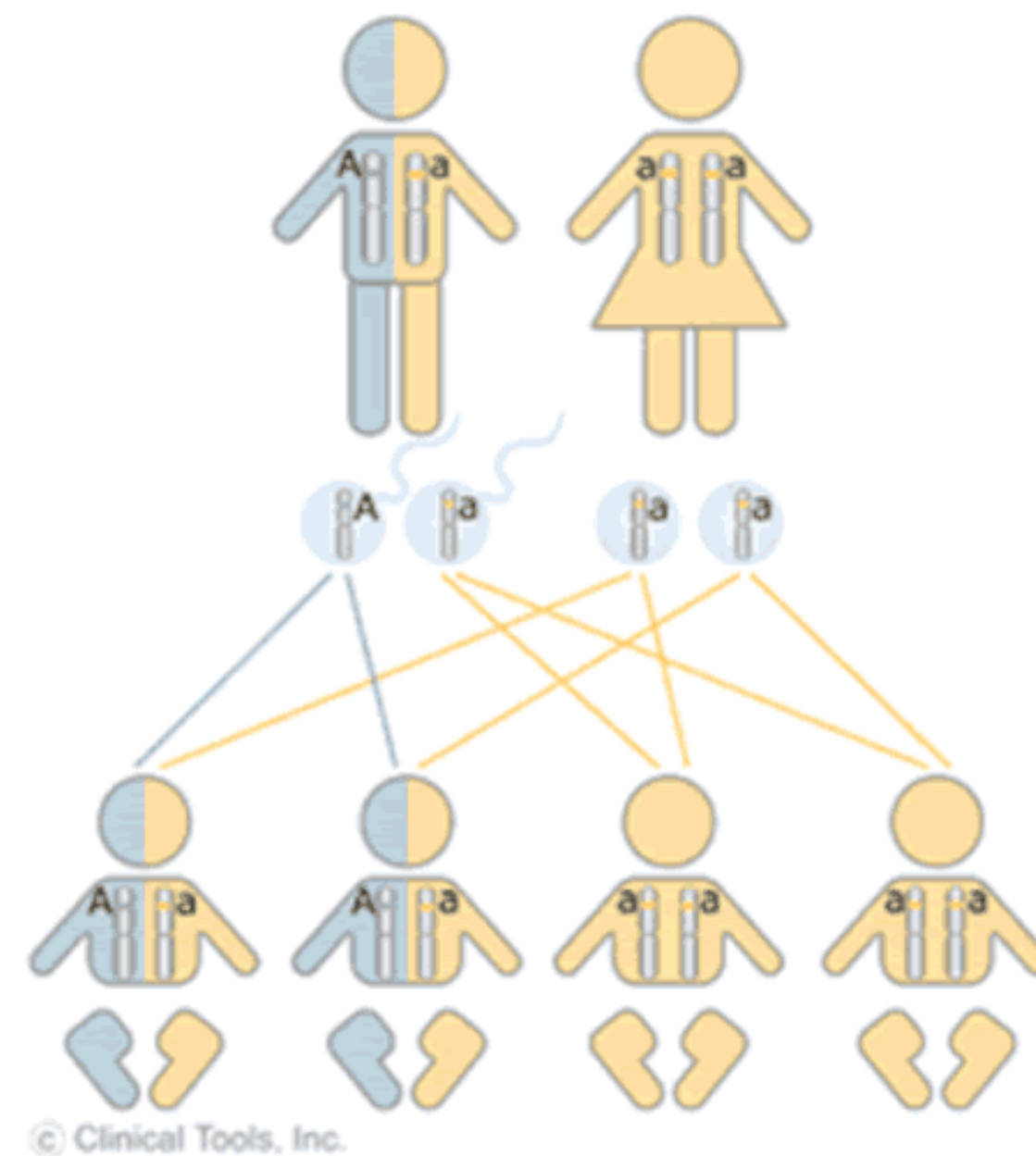
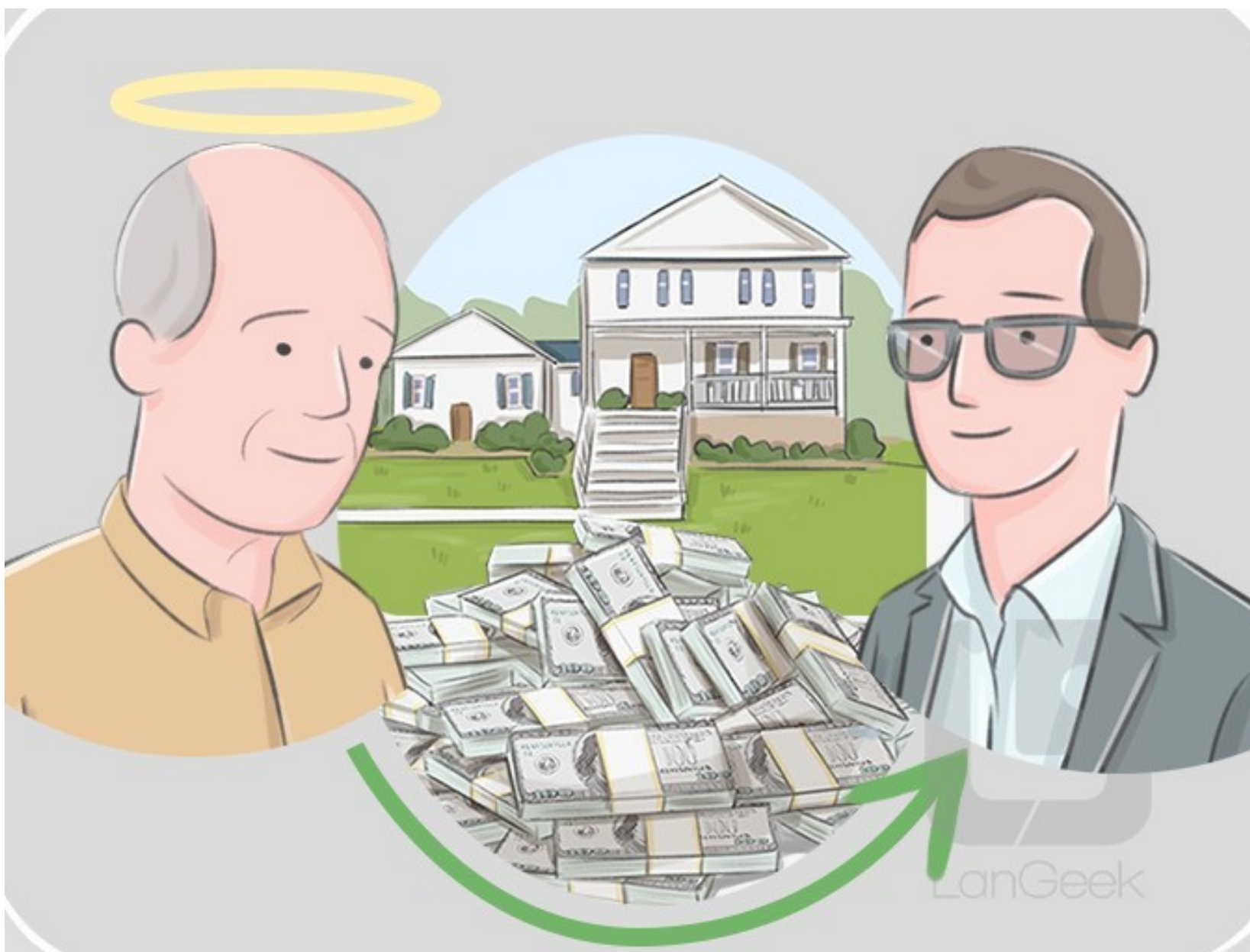
- **ENTENDER O CONCEITO DE HERANÇA;**
- **ENTENDER A HIERARQUIA ENTRE CLASSES MÃE E FILHAS;**
- **CONHECER O MODIFICADOR DE ACESSO PROTECTED;**
- **DISCUSSÕES E DÚVIDAS GERAIS.**

HERANÇA



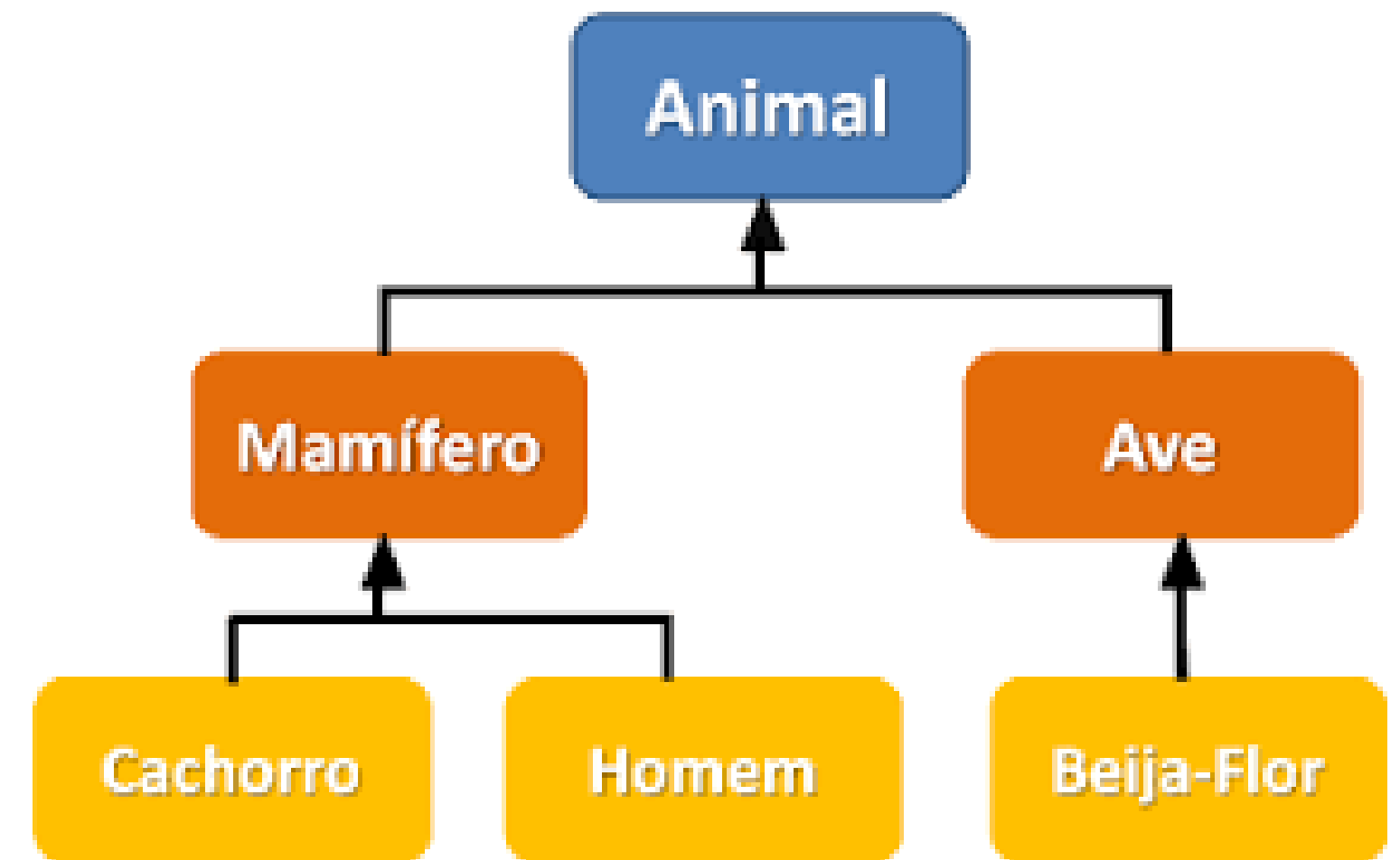
DEFINIÇÃO DE HERANÇA

O que pensamos quando falamos em herança?



HERANÇA EM POO

Em POO, herança é um conceito fundamental que permite que uma classe (conhecida como a classe **filha** ou **subclasse**) herde características e comportamentos de outra classe (conhecida como a classe **mãe** ou **superclasse**). A classe filha é capaz de herdar atributos e métodos da classe pai e também pode adicionar novos atributos e métodos ou modificar os existentes.



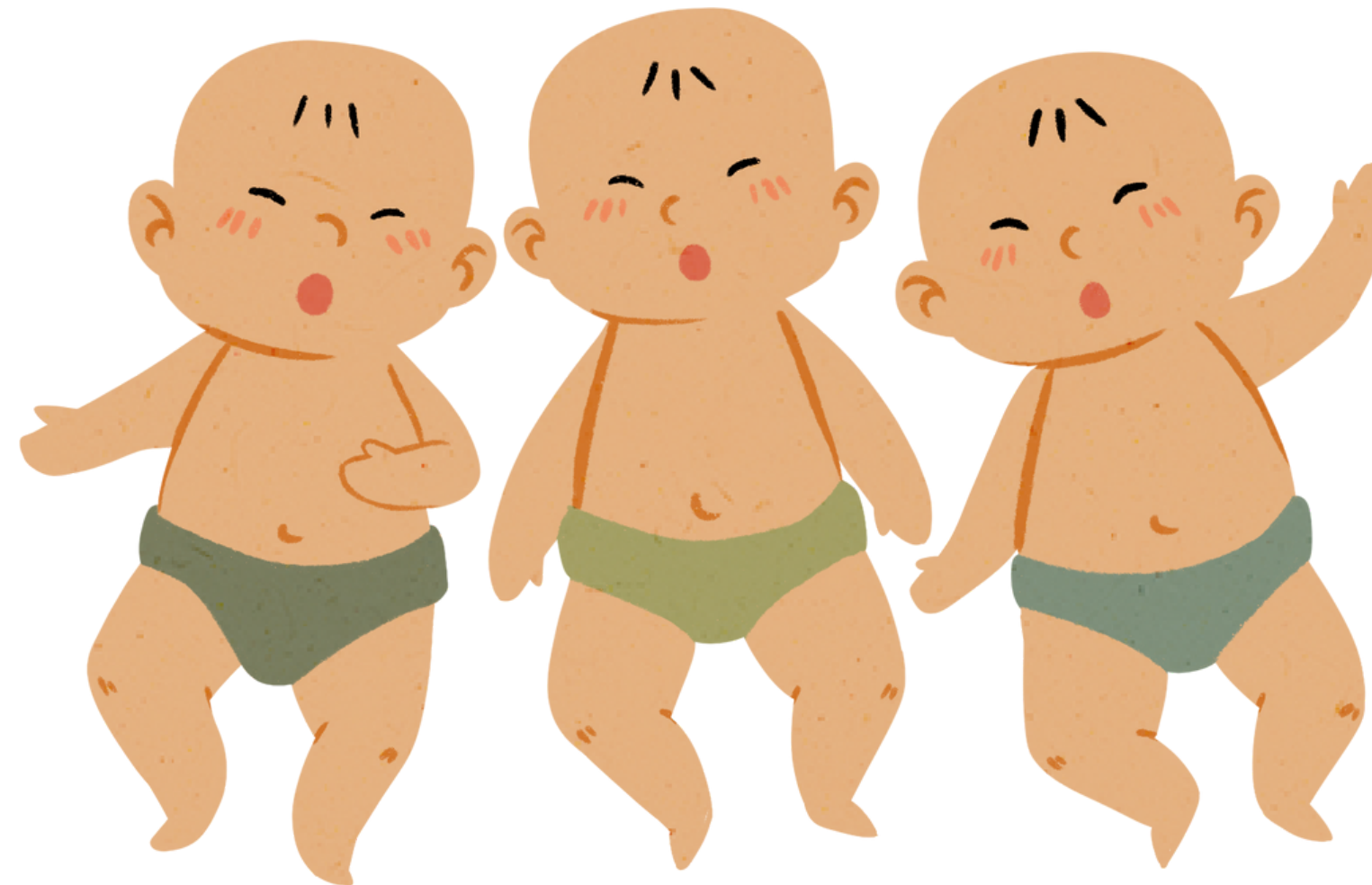
VANTAGENS DA HERANÇA

- A vantagem da herança é agrupar coisas comuns para poder reaproveitar o código;
- Legibilidade e manutenção;
- Abstração e generalização;
- Extensibilidade



O NÃO USO DE HERANÇA IMPLICA...

DUPLICAÇÃO DE CÓDIGO



HERANÇA EM JAVA

- Em Java criamos a **superclasse** como qualquer outra classe em java, sem adição de nenhum termo.
- Já as classes filhas são criadas utilizando a palavra chave **extends** após o nome da classe;

```
public class Funcionario extends Pessoa{  
    private int matricula;  
    private String funcao;  
    private double salario;
```


MODIFICADOR DE ACESSO PROTECTED

É um modificador utilizando na herança de classes e permite que as classes filhas tenha acessos a todos os atributos e metodos declarados na classe mãe (superclasse)

```
public class Pessoa {  
    protected String nome;  
    protected String cpf;  
  
    public Pessoa(){  
        this.cpf = "000.000.000-00";  
        this.nome = "Fulano de Souza";  
    }  
  
    public Pessoa(String nome, String cpf){  
        this.cpf = cpf;  
        this.nome = nome;  
    }  
}
```

PALAVRA RESERVADA SUPER

Permite acessar e inicializar o construtor da superclasse ou ainda chamar os métodos da classe mãe em funções override (subscritas)

```
public class Funcionario extends Pessoa{  
    private int matricula;  
    private String funcao;  
    private double salario;  
  
    public Funcionario() {  
        super(nome: "Francisca Maria", cpf: "074.874.541-45");  
        this.matricula = 0;  
        this.funcao = "SEM";  
        this.salario = 0;  
    }  
}
```

PALAVRA RESERVADA SUPER

- ❑ Sobrescrita de métodos:
 - ❖ Técnica conhecida como **override**;
 - ❖ Quando uma classe herda de outra, ela pode **redefinir métodos** da superclasse, isto é, sobrescrever métodos.
 - Os métodos sobrescritos **substituem** os métodos da superclasse;
 - A **assinatura do método** sobrescrito deve ser a mesma do método original.

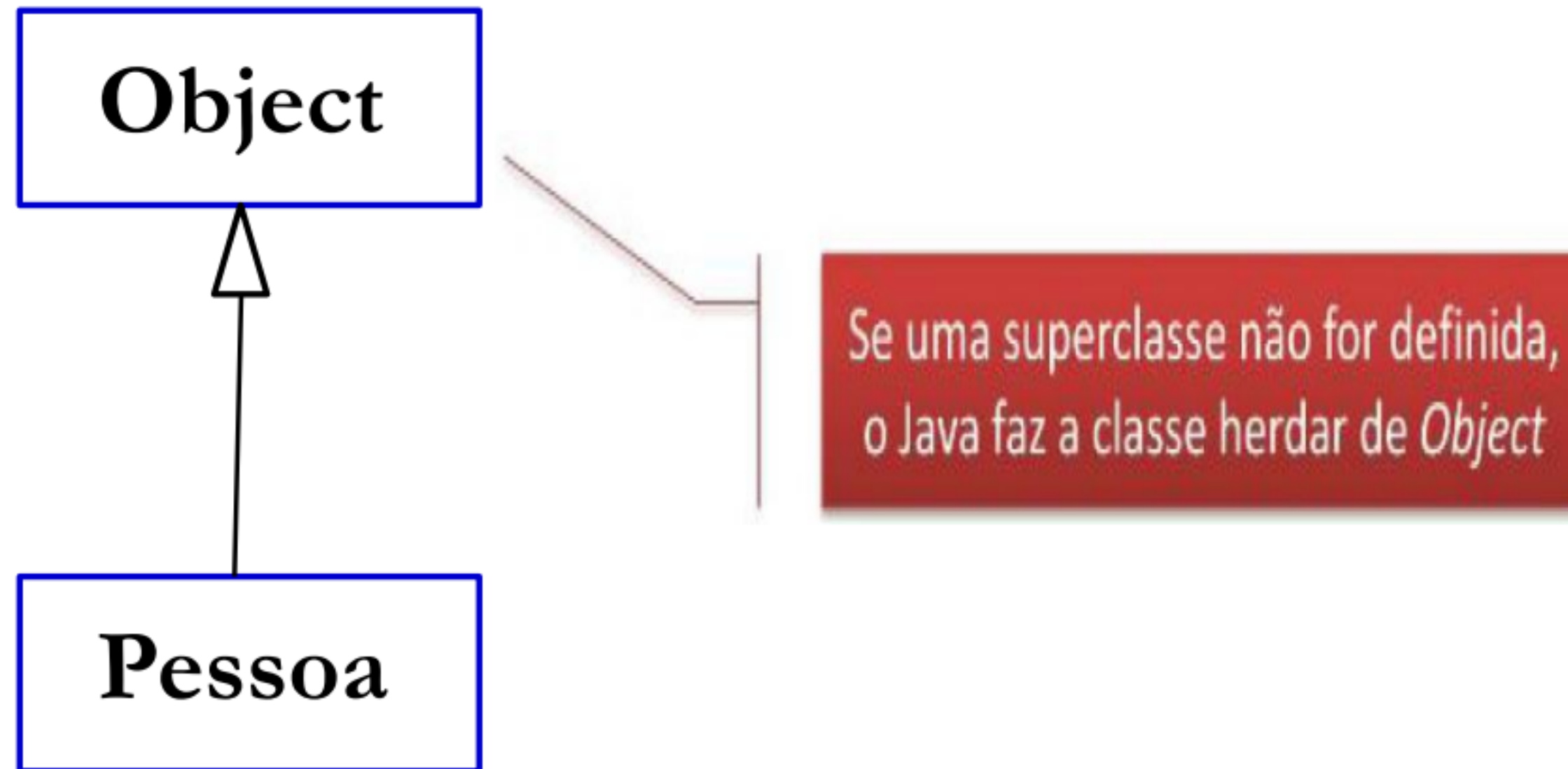
PALAVRA RESERVADA SUPER

```
public class Pessoa {  
    protected String nome;  
    protected String cpf;  
  
    public Pessoa(){  
        this.cpf = "000.000.000-00";  
        this.nome = "Fulano de Souza";  
    }  
  
    public Pessoa(String nome, String cpf){  
        this.cpf = cpf;  
        this.nome = nome;  
    }  
    1 override  
    protected void falar() {  
        System.out.println("Pessoa falando...");  
    }  
}
```

```
public class Funcionario extends Pessoa {  
    private int matricula;  
    private String funcao;  
    private double salario;  
  
    public Funcionario(){  
        super(nome: "Francisca Maria", cpf: "074.874.541-45");  
        this.matricula = 0;  
        this.funcao = "SEM";  
        this.salario = 0;  
    }  
    @Override  
    public void falar(){  
        System.out.println("Funcionario falando...");  
        super.falar();  
    }  
}
```

SUPERCLASSE OBJECT

- ❑ Todas as classes **herdam** apenas de uma superclasse:



CONCEITOS IMPORTANTES

- ❑ **Não confunda Overloading** ou com **Overriding**.
 - ❑ Overloading (Sobrecarga de Método):
 - ❖ Same method name but different parameters.
 - ❖ Overloading happens at compile-time;
 - ❖ **Static binding** is being used for overloaded methods.
 - ❑ Overriding (Sobrescrita de Método):
 - ❖ Same method name and parameters (i.e., method signature);
 - ❖ Overriding happens at runtime;
 - ❖ **Dynamic binding or Late binding** is being used for overriding methods.
-

