



### 3. HERANÇA

PROGRAMAÇÃO ORIENTADA AOS OBJETOS

Desenvolvido por:

Carlos Urbano  
Catarina Reis  
José Magno  
Marco Ferreira  
Ricardo Antunes

- 3.1. DEFINIÇÃO DE HERANÇA
- 3.2. O QUE HÁ EM COMUM?
- 3.3. O QUE PODE SER HERDADO?
- 3.4. EXTRAÇÃO DE UMA SUPERCLASSE
- 3.5. EXEMPLO
- 3.6. O QUE HÁ AINDA EM COMUM?
- 3.7. O QUE AINDA PODE SER HERDADO?
- 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE
- 3.9. CLASSES ABSTRATAS
- 3.10. EXTRAÇÃO DE OUTRA SUPERCLASSE – CORRIGIDO
- 3.11. O QUE HÁ AINDA EM COMUM?
- 3.12. CLASSES ABSTRATAS – IMPOSIÇÃO vs OPÇÃO
- 3.13. O QUE AINDA PODE SER HERDADO? – CORRIGIDO
- 3.14. EXEMPLO
- 3.15. REFERÊNCIAS this E super

## 3.1. DEFINIÇÃO DE HERANÇA

3

**É UM MECANISMO QUE PERMITE QUE UMA CLASSE HERDE ATRIBUTOS E MÉTODOS DE OUTRA**

**UMA DAS SUAS GRANDES VANTAGENS É A REUTILIZAÇÃO DO CÓDIGO**

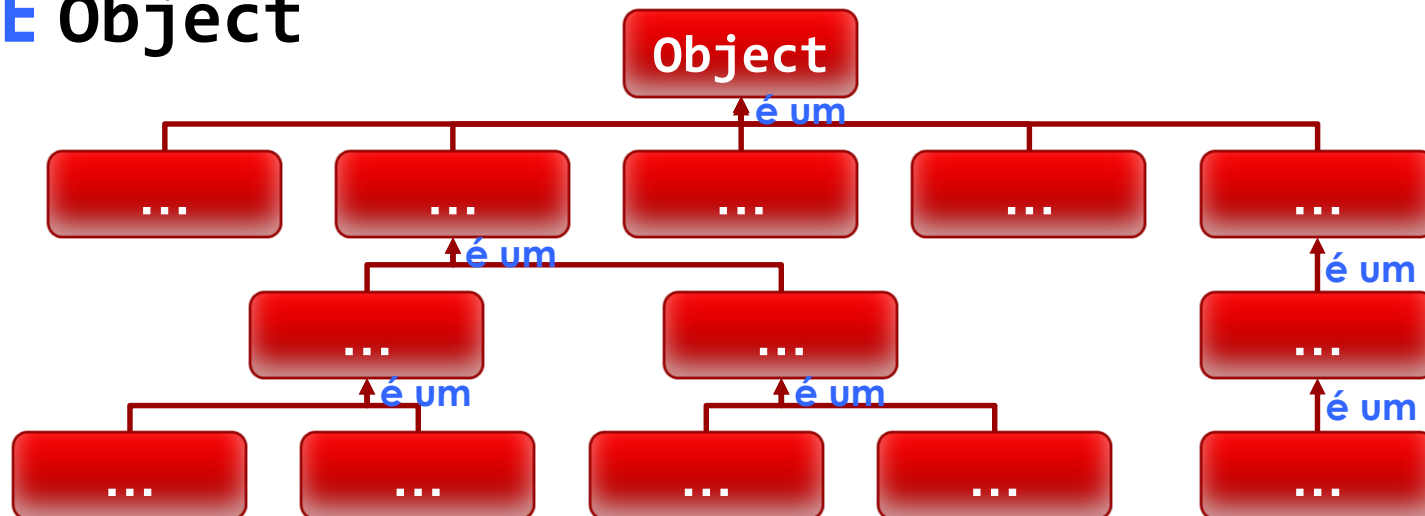
**ESSA REUTILIZAÇÃO ACONTECE QUANDO SE IDENTIFICA QUE ATRIBUTOS E/OU MÉTODOS SÃO COMUNS VÁRIAS CLASSES**

### 3.1. DEFINIÇÃO DE HERANÇA

COM A HERANÇA OBTÉM-SE A HIERARQUIZAÇÃO DAS CLASSES (DA MAIS GENÉRICA PARA A MAIS ESPECIALIZADA OU DETALHADA)

EM JAVA, APENAS EXISTE HERANÇA SIMPLES (CADA CLASSE APENAS PODE HERDAR DE OUTRA) SENDO A “RAIZ” (A MÃE DE TODAS) A CLASSE **Object**

EM JAVA, UMA CLASSE HERDA POR OMISSÃO DA CLASSE **Object**



## 3.2. O QUE HÁ EM COMUM?

5

**professor**

**aluno**

**aula**

### PROPRIEDADES

nome  
número  
aulas

### PROPRIEDADES

nome  
número  
aulas

### PROPRIEDADES

nome  
número  
sumário  
professor  
alunos

### FUNCIONALIDADES

adicionar(aula)  
preencherSumário(aula)  
número?  
nome?  
atribuir(número)  
remover(aula)

### FUNCIONALIDADES

adicionar(aula)  
preencherSumário(aula)  
número?  
nome?  
atribuir(número)  
remover(aula)

### FUNCIONALIDADES

atribuir(professor)  
adicionar(aluno)  
adicionarLinhaSumário(linha)  
número?  
nome?  
atribuir(número)  
sumário?  
professor?  
remover(aluno)  
desassociarProfessor()  
alunos?

CONSTRUTOR

CONSTRUTOR

CONSTRUTOR

## 3.2. O QUE HÁ EM COMUM?

6

**professor**

**aluno**

**aula**

### PROPRIEDADES

nome  
número  
aulas

### PROPRIEDADES

nome  
número  
aulas

### PROPRIEDADES

nome  
número  
sumário  
professor  
alunos

### FUNCIONALIDADES

adicionar(aula)  
preencherSumário(aula)  
número?  
nome?  
atribuir(número)  
remover(aula)

### FUNCIONALIDADES

adicionar(aula)  
preencherSumário(aula)  
número?  
nome?  
atribuir(número)  
remover(aula)

### FUNCIONALIDADES

atribuir(professor)  
adicionar(aluno)  
adicionarLinhaSumário(linha)  
número?  
nome?  
atribuir(número)  
sumário?  
professor?  
remover(aluno)  
desassociarProfessor()  
alunos?

CONSTRUTOR

CONSTRUTOR

CONSTRUTOR

## 3.2. O QUE HÁ EM COMUM?

7

professor

aluno

aula

### PROPRIEDADES

nome  
número  
aulas

### PROPRIEDADES

nome  
número  
aulas

### PROPRIEDADES

nome  
número  
sumário  
professor  
alunos

### FUNCCIONALIDADES

adicionar(aula)  
preencherSumário(aula)  
número?  
nome?  
atribuir(número)  
remover(aula)

### FUNCCIONALIDADES

adicionar(aula)  
preencherSumário(aula)  
número?  
nome?  
atribuir(número)  
remover(aula)

### FUNCCIONALIDADES

atribuir(professor)  
adicionar(aluno)  
adicionar(linha) Sumário(linha)  
número?  
nome?  
atribuir(número)  
sumário?  
professor?  
remover(aluno)  
desassociarProfessor()  
alunos?

COMUM AO MAIOR  
NÚMERO DE CLASSES  
A HERDAR DA MESMA  
SUPERCLASSE?

CONSTRUTOR

CONSTRUTOR

CONSTRUTOR

## 3.2. O QUE HÁ EM COMUM?

8

```
public class Professor {  
    ...  
    public String getNome() {  
        return nome;  
    }  
  
    public long getNumero() {  
        return numero;  
    }  
  
    public void setNumero(long numero) {  
        this.numero = numero;  
    }  
}
```

```
public class Aluno {  
    ...  
    public String getNome() {  
        return nome;  
    }  
  
    public long getNumero() {  
        return numero;  
    }  
  
    public void setNumero(long numero) {  
        this.numero = numero;  
    }  
}
```

```
public class Aula {  
    ...  
    public String getNome() {  
        return nome;  
    }  
  
    public long getNumero() {  
        return numero;  
    }  
  
    public void setNumero(long numero) {  
        this.numero = numero;  
    }  
}
```

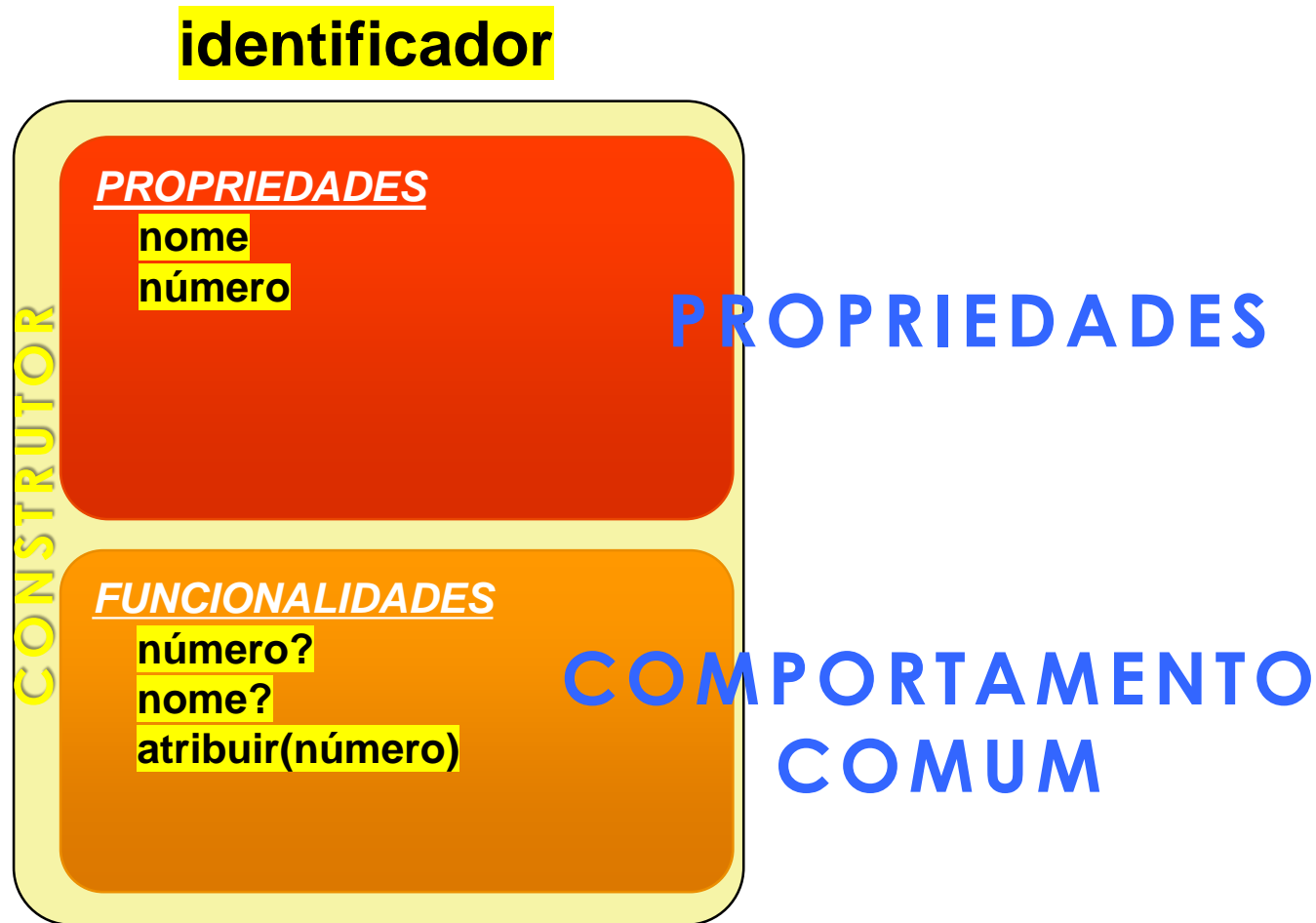
COMPORTAMENTO  
COMUM

CLASSE

Identificador



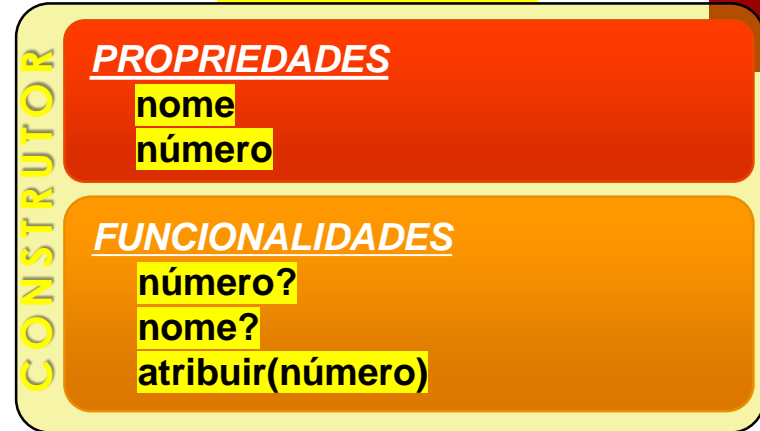
## 3.2. O QUE HÁ EM COMUM?



## 3.2. O QUE HÁ EM COMUM?

identificador

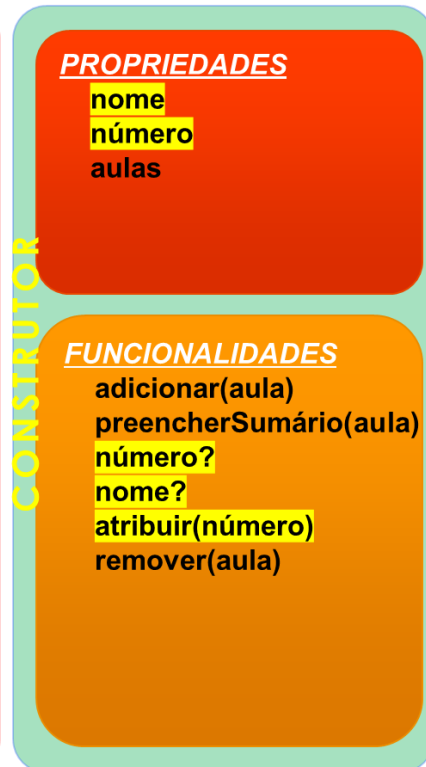
10



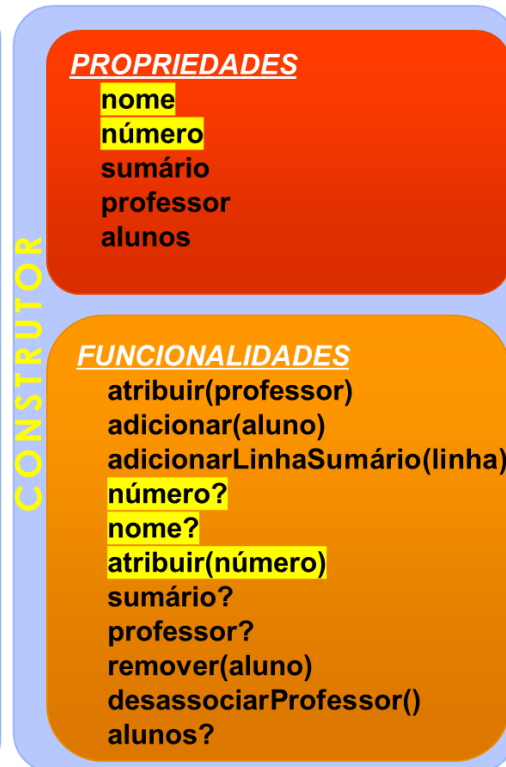
professor



aluno



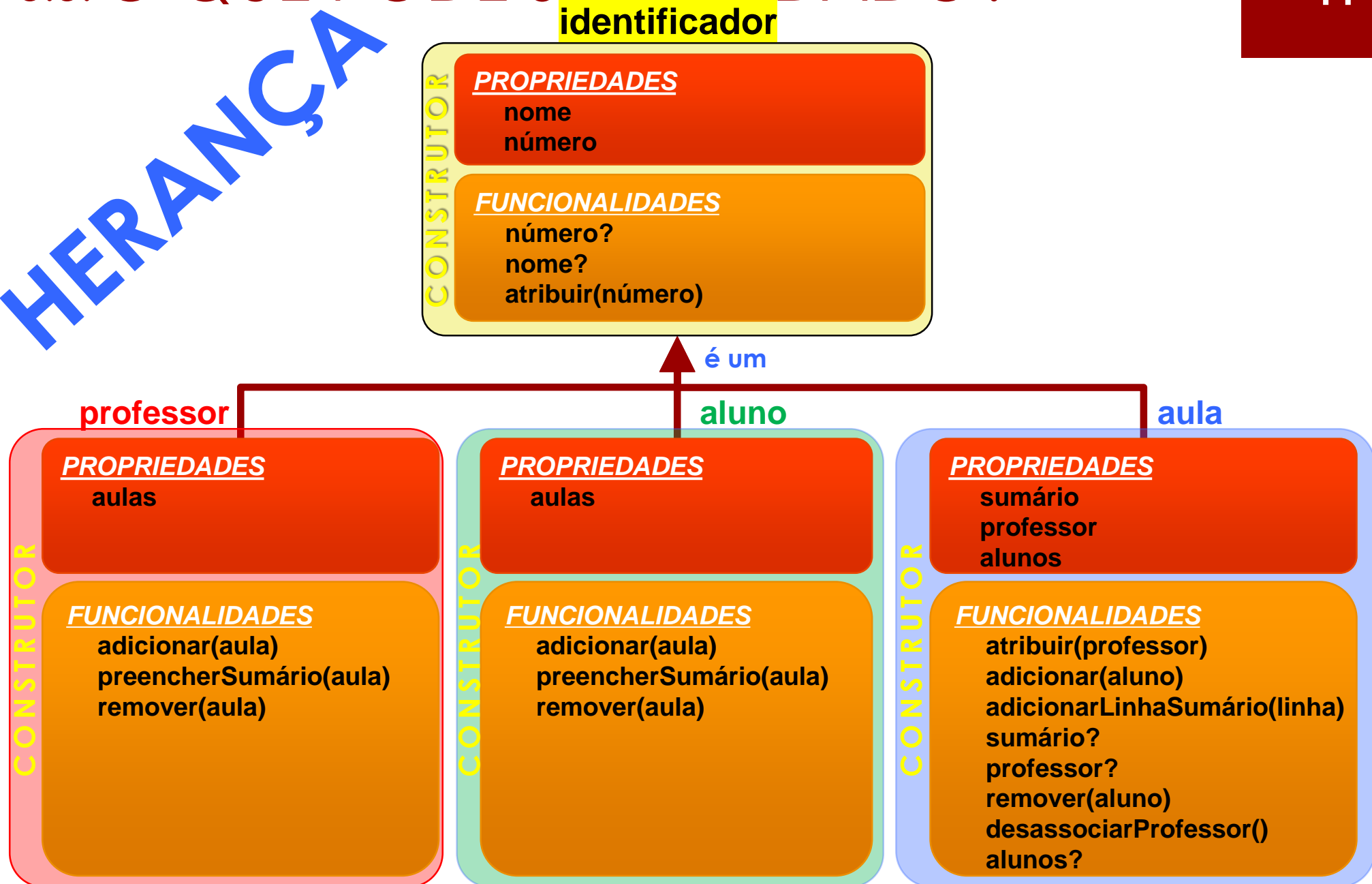
aula



### 3.3. O QUE PODE SER HERDADO?

11

HERANÇA



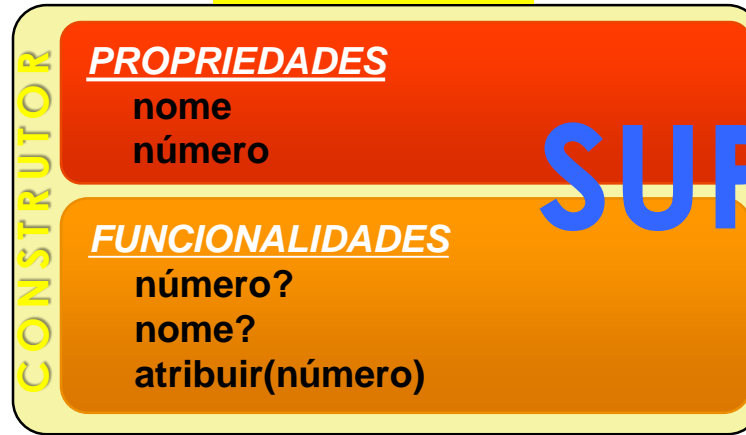
### 3.3. O QUE PODE SER HERDADO?

12

HERANÇA

identificador

SUPERCLASSE

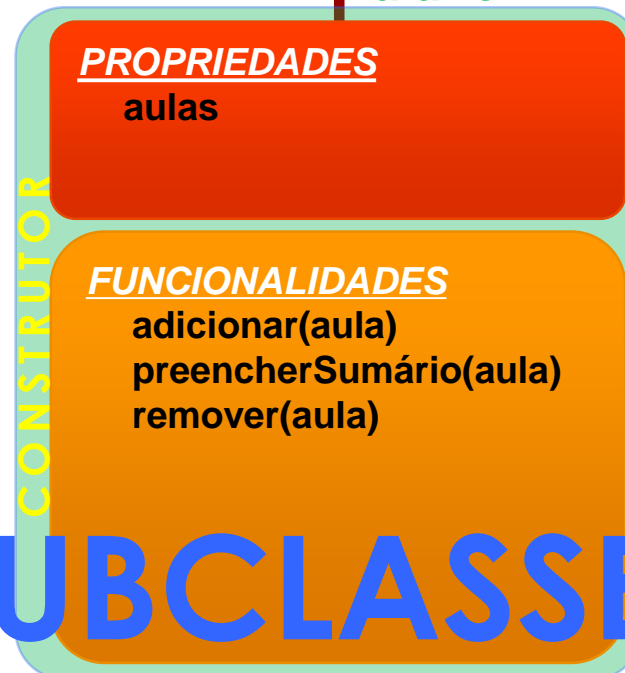


é um

professor

aluno

aula



SUBCLASSES

**OUTRA ABORDAGEM PARA  
IDENTIFICAR O QUE É  
COMUM EM DIAGRAMAS  
DE CLASSES COMPLEXOS**

### 3.3. O QUE PODE SER HERDADO?

14

|                   | superclasses/interfaces       |           |       |      |
|-------------------|-------------------------------|-----------|-------|------|
|                   |                               | Professor | Aluno | Aula |
|                   | classes                       |           |       |      |
| <b>tipo</b>       | <b>atributo</b>               |           |       |      |
| String            | nome                          | x         | x     | x    |
| long              | numero                        | x         | x     | x    |
| LinkedList<Aula>  | aulas                         | x         | x     |      |
| String            | sumario                       |           |       | x    |
| Professor         | professor                     |           |       | x    |
| LinkedList<Aluno> | alunos                        |           |       | x    |
| <b>return</b>     | <b>método</b>                 |           |       |      |
| void              | setProfessor(Professor)       |           |       | x    |
| void              | adicionar(Aula)               | x         | x     |      |
| void              | preencherSumario(Aula)        | x         | x     |      |
| void              | adicionar(Aluno)              |           |       | x    |
| void              | adicionarLinhaSumario(String) |           |       | x    |
| String            | getNome()                     | x         | x     | x    |
| long              | getNumero()                   | x         | x     | x    |
| void              | setNumero(long)               | x         | x     | x    |
| String            | getSumario()                  |           |       | x    |
| Professor         | getProfessor()                |           |       | x    |
| void              | desassociarProfessor()        |           |       | x    |
| void              | remover(Aula)                 | x         | x     |      |
| LinkedList<Aluno> | getAlunos()                   |           |       | x    |
| void              | remover(Aluno)                |           |       | x    |

COMUM AO  
MAIOR  
NÚMERO DE  
CLASSES A  
HERDAR DA  
MESMA  
SUPERCLASSE?

### 3.3. O QUE PODE SER HERDADO?

15

|                   | superclasses/interfaces       |           |       |      |               |
|-------------------|-------------------------------|-----------|-------|------|---------------|
|                   | classes                       | Professor | Aluno | Aula | Identificador |
| <b>tipo</b>       | <b>atributo</b>               |           |       |      |               |
| String            | nome                          | x         | x     | x    | x             |
| long              | numero                        | x         | x     | x    | x             |
| LinkedList<Aula>  | aulas                         | x         | x     |      |               |
| String            | sumario                       |           |       | x    |               |
| Professor         | professor                     |           |       | x    |               |
| LinkedList<Aluno> | alunos                        |           |       | x    |               |
| <b>return</b>     | <b>método</b>                 |           |       |      |               |
| void              | setProfessor(Professor)       |           |       | x    |               |
| void              | adicionar(Aula)               | x         | x     |      |               |
| void              | preencherSumario(Aula)        | x         | x     |      |               |
| void              | adicionar(Aluno)              |           |       | x    |               |
| void              | adicionarLinhaSumario(String) |           |       | x    |               |
| String            | getNome()                     | x         | x     | x    | x             |
| long              | getNumero()                   | x         | x     | x    | x             |
| void              | setNumero(long)               | x         | x     | x    | x             |
| String            | getSumario()                  |           |       | x    |               |
| Professor         | getProfessor()                |           |       | x    |               |
| void              | desassociarProfessor()        |           |       | x    |               |
| void              | remover(Aula)                 | x         | x     |      |               |
| LinkedList<Aluno> | getAlunos()                   |           |       | x    |               |
| void              | remover(Aluno)                |           |       | x    |               |

### 3.3. O QUE PODE SER HERDADO?

16

|                   | superclasses/interfaces       | Identificador | Identificador | Identificador |               |
|-------------------|-------------------------------|---------------|---------------|---------------|---------------|
|                   | classes                       | Professor     | Aluno         | Aula          | Identificador |
| <b>tipo</b>       | <b>atributo</b>               |               |               |               |               |
| String            | nome                          |               |               |               | x             |
| long              | numero                        |               |               |               | x             |
| LinkedList<Aula>  | aulas                         | x             | x             |               |               |
| String            | sumario                       |               |               | x             |               |
| Professor         | professor                     |               |               | x             |               |
| LinkedList<Aluno> | alunos                        |               |               | x             |               |
| <b>return</b>     | <b>método</b>                 |               |               |               |               |
| void              | setProfessor(Professor)       |               |               | x             |               |
| void              | adicionar(Aula)               | x             | x             |               |               |
| void              | preencherSumario(Aula)        | x             | x             |               |               |
| void              | adicionar(Aluno)              |               |               | x             |               |
| void              | adicionarLinhaSumario(String) |               |               | x             |               |
| String            | getNome()                     |               |               |               | x             |
| long              | getNumero()                   |               |               |               | x             |
| void              | setNumero(long)               |               |               |               | x             |
| String            | getSumario()                  |               |               | x             |               |
| Professor         | getProfessor()                |               |               | x             |               |
| void              | desassociarProfessor()        |               |               | x             |               |
| void              | remover(Aula)                 | x             | x             |               |               |
| LinkedList<Aluno> | getAlunos()                   |               |               | x             |               |
| void              | remover(Aluno)                |               |               | x             |               |

**CASO 1: TODO  
O CÓDIGO  
PASSA PARA A  
SUPERCLASSE**



**VAMOS AGORA  
EXTRAIR A SUPERCLASSE  
Identificador**

## 3.4. EXTRAÇÃO DE UMA SUPERCLASSE

18

```
public class Identificador {  
    protected String nome;  
    protected long numero;
```

**ATRIBUTOS COMUNS COM  
ACESSO PROTEGIDO**

```
    public Identificador(String nome, long numero) {  
        this.nome = nome;  
        this.numero = numero;  
    }
```

```
    public String getNome() {  
        return nome;  
    }
```

```
    public long getNumero() {  
        return numero;  
    }
```

```
    public void setNumero(long numero) {  
        this.numero = numero;  
    }
```

```
}
```

**COMPORTAMENTO  
COMUM**

## 3.4. EXTRAÇÃO DE UMA SUPERCLASSE

19

```
public class Professor extends Identificador {  
    private LinkedList<Aula> aulas;  
  
    public Professor(String nome, long numero) {  
        this(nome, numero, new LinkedList<>());  
    }  
  
    public Professor(String nome, long numero, LinkedList<Aula> aulas) {  
        this.nome = nome;  
        this.numero = numero;  
        this.aulas = new LinkedList<>();  
        for (Aula aula : aulas) {  
            adicionar(aula);  
        }  
    }  
  
    public void adicionar(Aula aula) { ... }  
  
    public void remover(Aula aula) { ... }  
  
    public void preencherSumario(Aula aula) { ... }  
}
```

HERDADO DA CLASSE Identificador

nome  
numero  
getNome()  
getNumero()  
setNumero(long)

## 3.4. EXTRAÇÃO DE UMA SUPERCLASSE

20

```
public class Professor extends Identificador {
    private LinkedList<Aula> aulas;

    public Professor(String nome, long numero) {
        this(nome, numero, new LinkedList<>());
    }

    public Professor(String nome, long numero, LinkedList<Aula> aulas) {
        this.nome = nome;
        this.numero = numero;
        this.aulas = new LinkedList<>();
        for (Aula aula : aulas) {
            adicionar(aula);
        }
    }



    public void adicionar(Aula aula) { ... }


    public void remover(Aula aula) { ... }

    public void preencherSumario(Aula aula) { ... }
}
```

## 3.4. EXTRAÇÃO DE UMA SUPERCLASSE

21

```
public class Professor extends Identificador {  
    private LinkedList<Aula> aulas;  
  
    public Professor(String nome, long numero) {  
        this(nome, numero, new LinkedList<>());  
    }  
  
    public Professor(String nome, long numero, LinkedList<Aula> aulas) {  
         this.nome = nome;  
         this.numero = numero;  
        this.aulas = new LinkedList<>();  
        for (Aula aula : aulas) {  
            adicionar(aula);  
        }  
    }  
  
    public void adicionar(Aula aula) { ... }  
  
    public void remover(Aula aula) { ... }  
  
    public void preencherSumario(Aula aula) { ... }  
}
```



```
super(nome, numero);
```

## 3.4. EXTRAÇÃO DE UMA SUPERCLASSE

22

```
public class Professor extends Identificador {  
    private LinkedList<Aula> aulas;  
  
    public Professor(String nome, long numero) {  
        this(nome, numero, new LinkedList<>());  
    }  
  
    public Professor(String nome, long numero, LinkedList<Aula> aulas) {  
        super(nome, numero);  
        this.aulas = new LinkedList<>();  
        for (Aula aula : aulas) {  
            adicionar(aula);  
        }  
    }  
  
    public void adicionar(Aula aula) { ... }  
  
    public void remover(Aula aula) { ... }  
  
    public void preencherSumario(Aula aula) { ... }  
}
```

INVOCAÇÃO DO  
CONSTRUTOR DA  
SUPERCLASSE

## 3.4. EXTRAÇÃO DE UMA SUPERCLASSE

23

```
public class Aluno extends Identificador {
    private LinkedList<Aula> aulas;

    public Aluno(String nome, long numero) {
        this(nome, numero, new LinkedList<>());
    }

    public Aluno(String nome, long numero, LinkedList<Aula> aulas) {
        super(nome, numero);
        this.aulas = new LinkedList<>();
        for (Aula aula : aulas) {
            adicionar(aula);
        }
    }

    public void adicionar(Aula aula) { ... }

    public void remover(Aula aula) { ... }

    public void preencherSumario(Aula aula) { ... }
}
```

INVOCAÇÃO DO  
CONSTRUTOR DA  
SUPERCLASSE

## 3.4. EXTRAÇÃO DE UMA SUPERCLASSE

24

```
public class Aula extends Identificador {  
    private String sumario;  
    private Professor professor;  
    private LinkedList<Aluno> alunos;  
  
    public Aula(String nome, long numero) {  
        this(nome, numero, null, new LinkedList<>());  
    }  
  
    public Aula(String nome, long numero, Professor professor,  
                LinkedList<Aluno> alunos) {  
        super(nome, numero);  
        this.sumario = "";  
        setProfessor(professor);  
        this.alunos = new LinkedList<>();  
        for (Aluno aluno : alunos) {  
            adicionar(aluno);  
        }  
    }  
}
```

HERDADO DA CLASSE Identificador

nome  
numero  
getNome()  
getNumero()  
setNumero(long)

INVOCAÇÃO DO  
CONSTRUTOR DA  
SUPERCLASSE

...



## 3.4. EXTRAÇÃO DE UMA SUPERCLASSE

25

...

```
public String getSumario() { ... }

public Professor getProfessor() { ... }

public void setProfessor(Professor professor) { ... }

public void adicionar(Aluno aluno) { ... }

public void desassociarProfessor() { ... }

public LinkedList<Aluno> getAlunos() { ... }

public void remover(Aluno aluno) { ... }

public void adicionarLinhaSumario(String linha) { ... }

}
```

## 3.4. EXTRAÇÃO DE UMA SUPERCLASSE

26

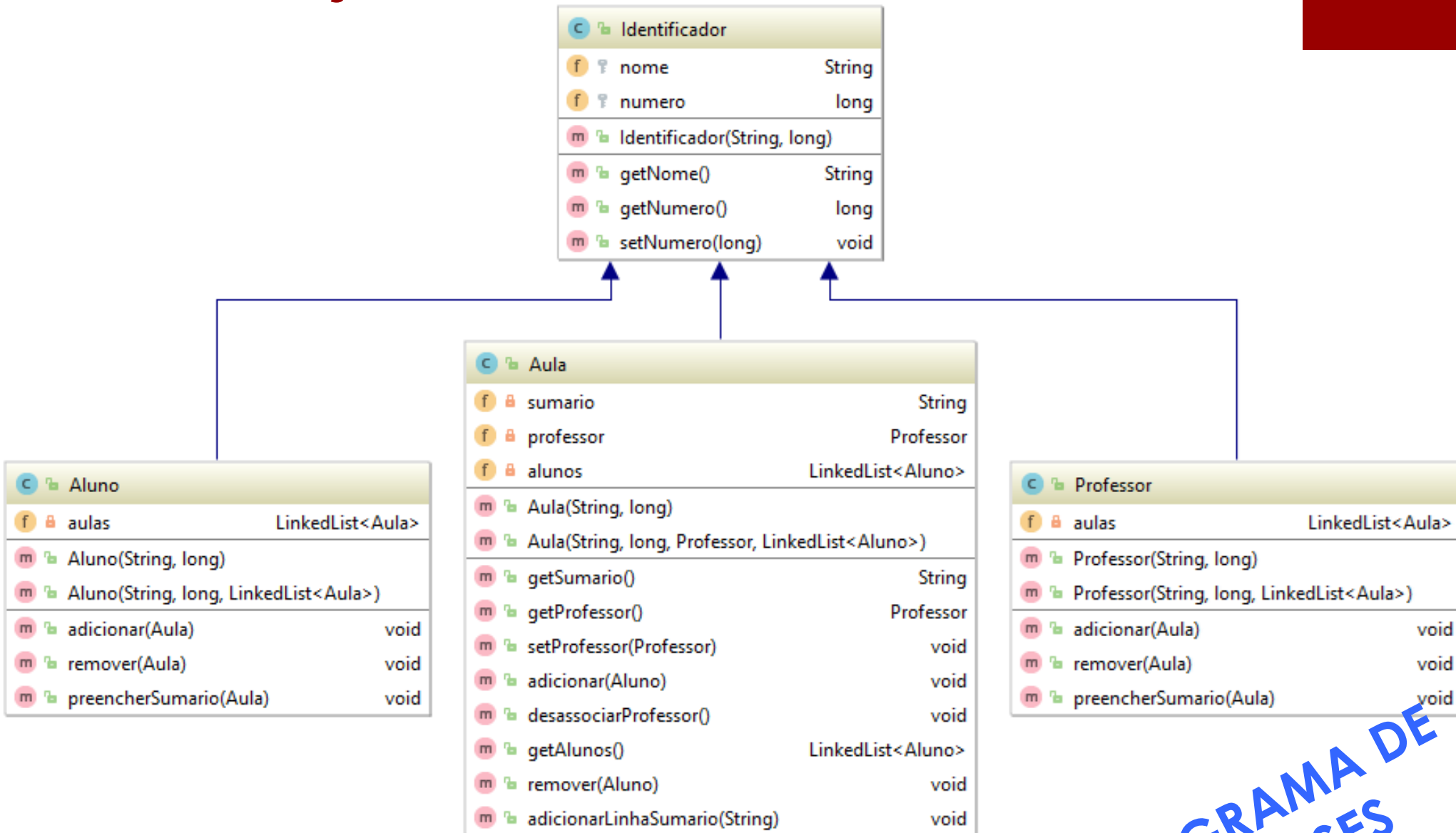


DIAGRAMA DE  
CLASSES

## 3.5. EXEMPLO

 Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```


## 3.5. EXEMPLO

28

 Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

o1

 Professor



## 3.5. EXEMPLO

29

 Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

o1

   
Professor

## 3.5. EXEMPLO

30

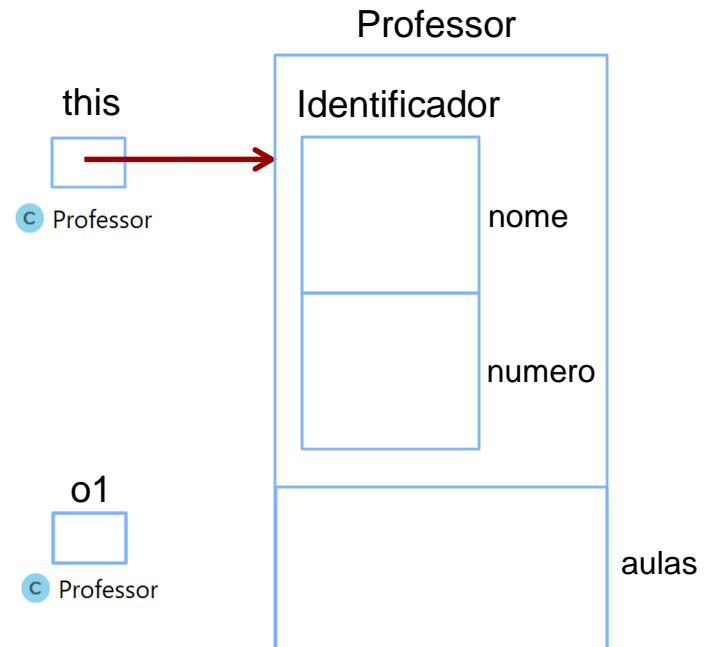
Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```



Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```



## 3.5. EXEMPLO

31

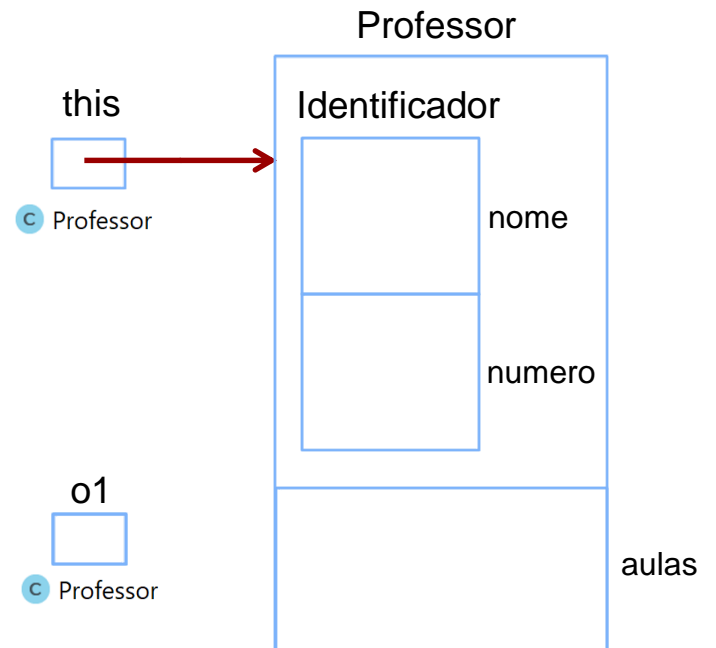
Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```



Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```



## 3.5. EXEMPLO

32

Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

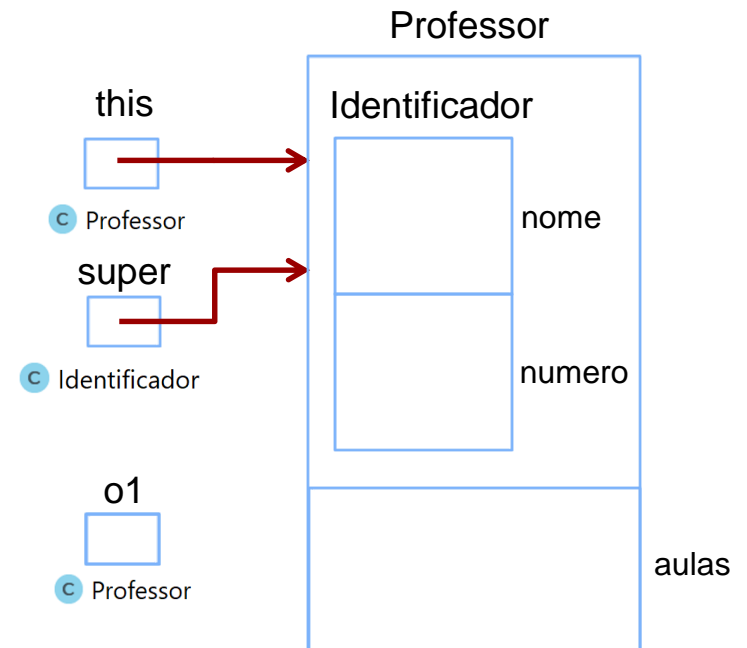
Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

Professor

```
public Professor(String nome, long numero,  
                  LinkedList<Aula> aulas) {  
    super(nome, numero);  
    this.aulas = new LinkedList<>();  
    for (Aula aula : aulas) {  
        adicionar(aula);  
    }  
}
```

NA INSTANCIÇÃO, O **super** É  
UMA REFERÊNCIA INTERNA  
PARA A INSTÂNCIA QUE ESTÁ A  
SER CONSTRUÍDA CONSIDERANDO  
APENAS A SUPERCLASSE





## 3.5. EXEMPLO

33

• Main

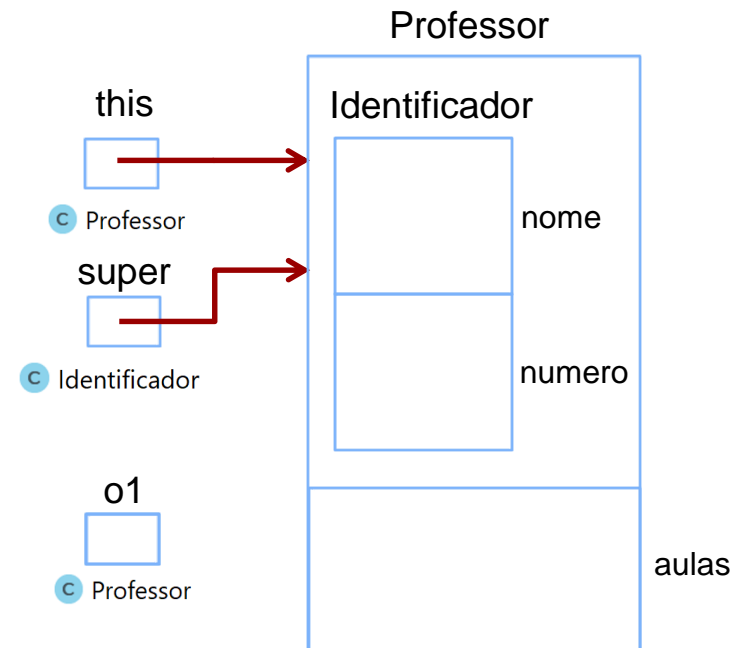
```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

• Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

• Professor

```
public Professor(String nome, long numero,  
                 LinkedList<Aula> aulas) {  
    super(nome, numero);  
    this.aulas = new LinkedList<>();  
    for (Aula aula : aulas) {  
        adicionar(aula);  
    }  
}
```



## 3.5. EXEMPLO

34

• Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

• Professor

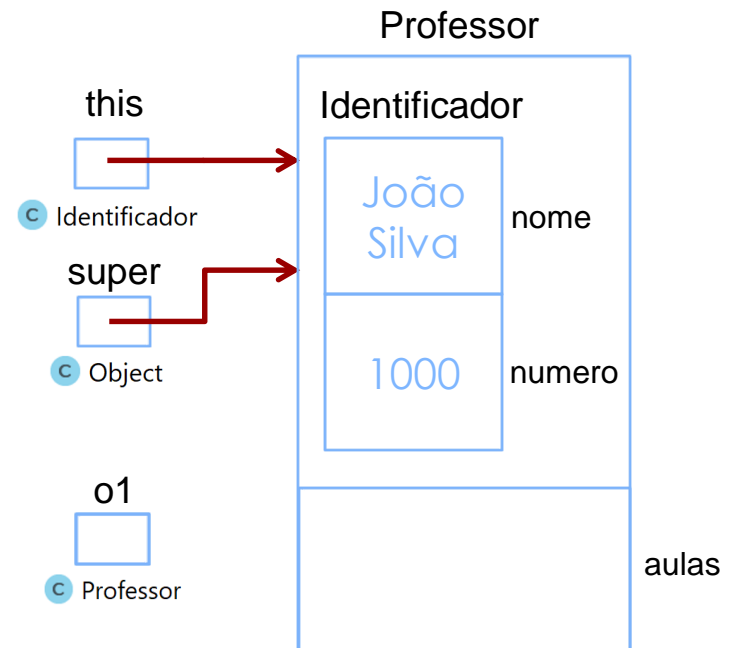
```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

• Professor

```
public Professor(String nome, long numero,  
                 LinkedList<Aula> aulas) {  
    super(nome, numero);  
    this.aulas = new LinkedList<>();  
    for (Aula aula : aulas) {  
        adiciona(aula);  
    }  
}
```

• Identificador

```
public Identificador(String nome, long numero) {  
    this.nome = nome;  
    this.numero = numero;  
}
```



## 3.5. EXEMPLO

35

• Main

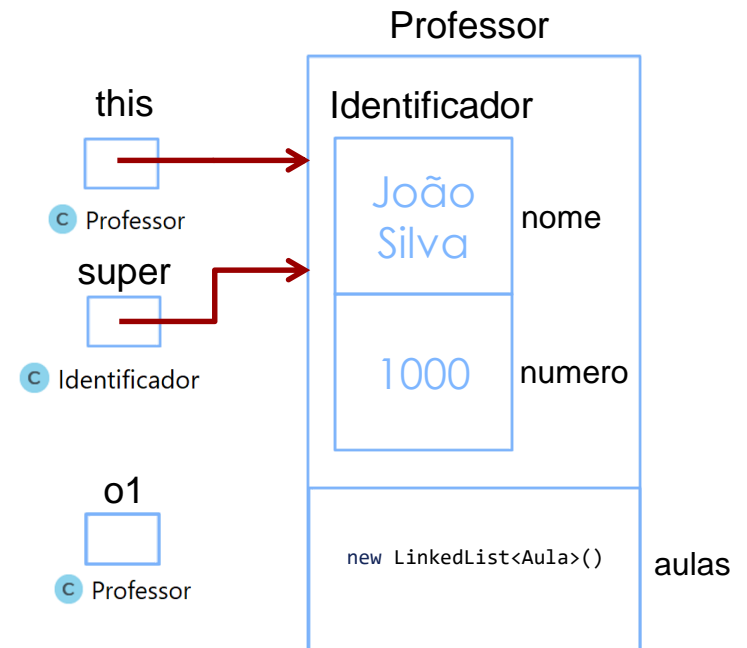
```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

• Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

• Professor

```
public Professor(String nome, long numero,  
                 LinkedList<Aula> aulas) {  
    super(nome, numero);  
    this.aulas = new LinkedList<>();  
    for (Aula aula : aulas) {  
        adicionar(aula);  
    }  
}
```



## 3.5. EXEMPLO

36

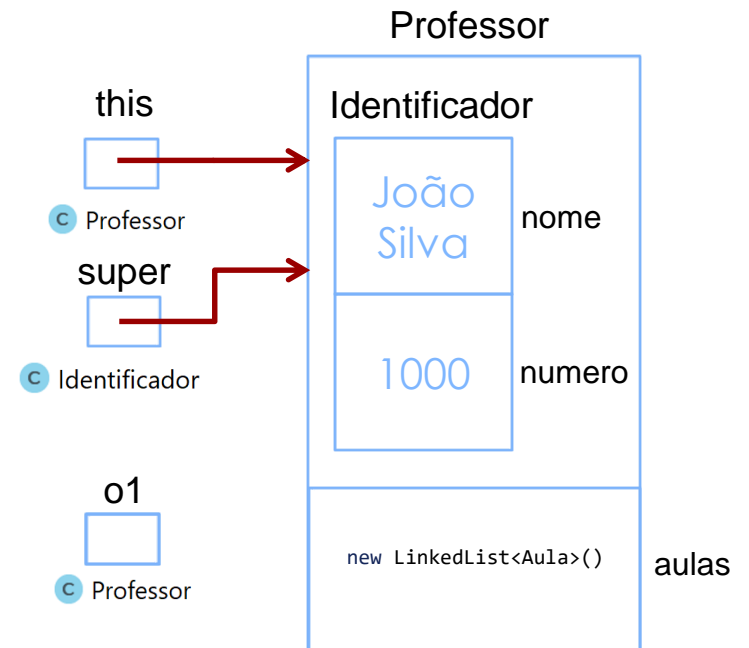
Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```



Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

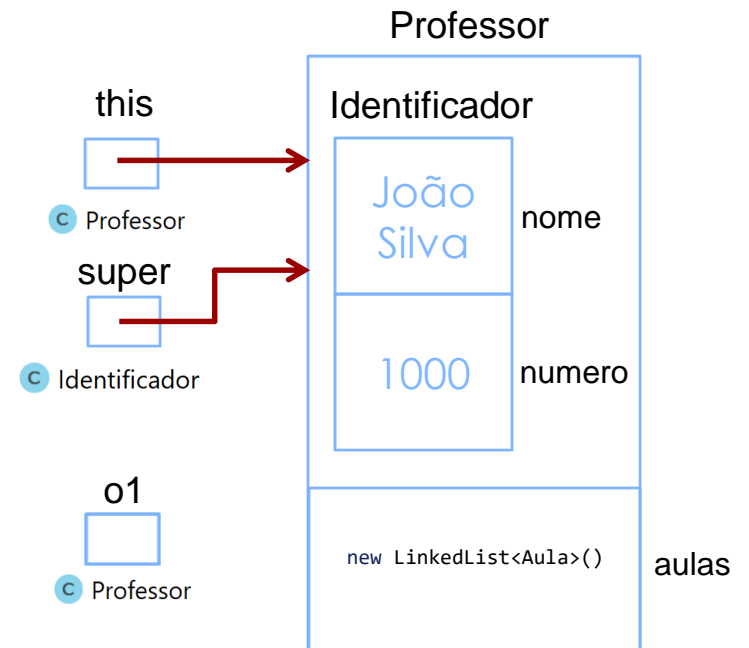


## 3.5. EXEMPLO

37

Ⓢ Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

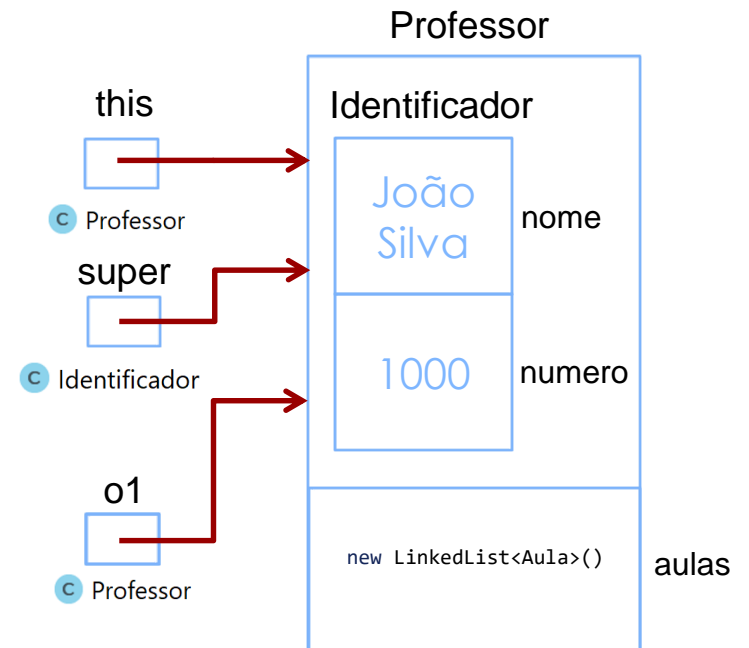


## 3.5. EXEMPLO

38

☐ Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

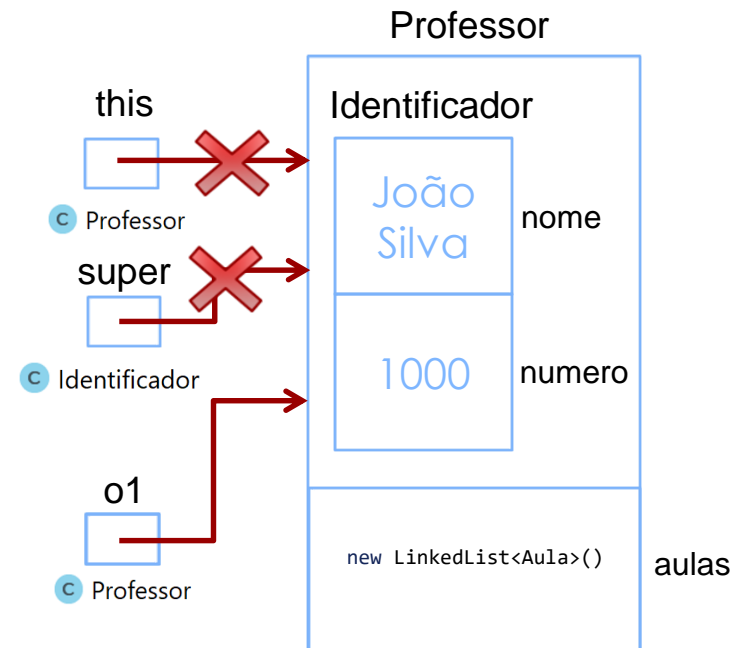


## 3.5. EXEMPLO

39

Ⓒ Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

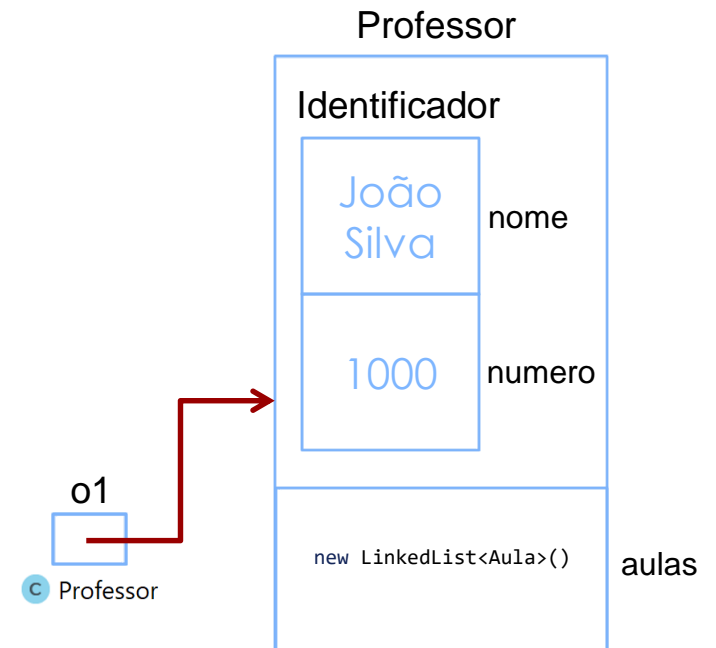


## 3.5. EXEMPLO

40

Ⓢ Main

```
public Main() {  
    ...  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

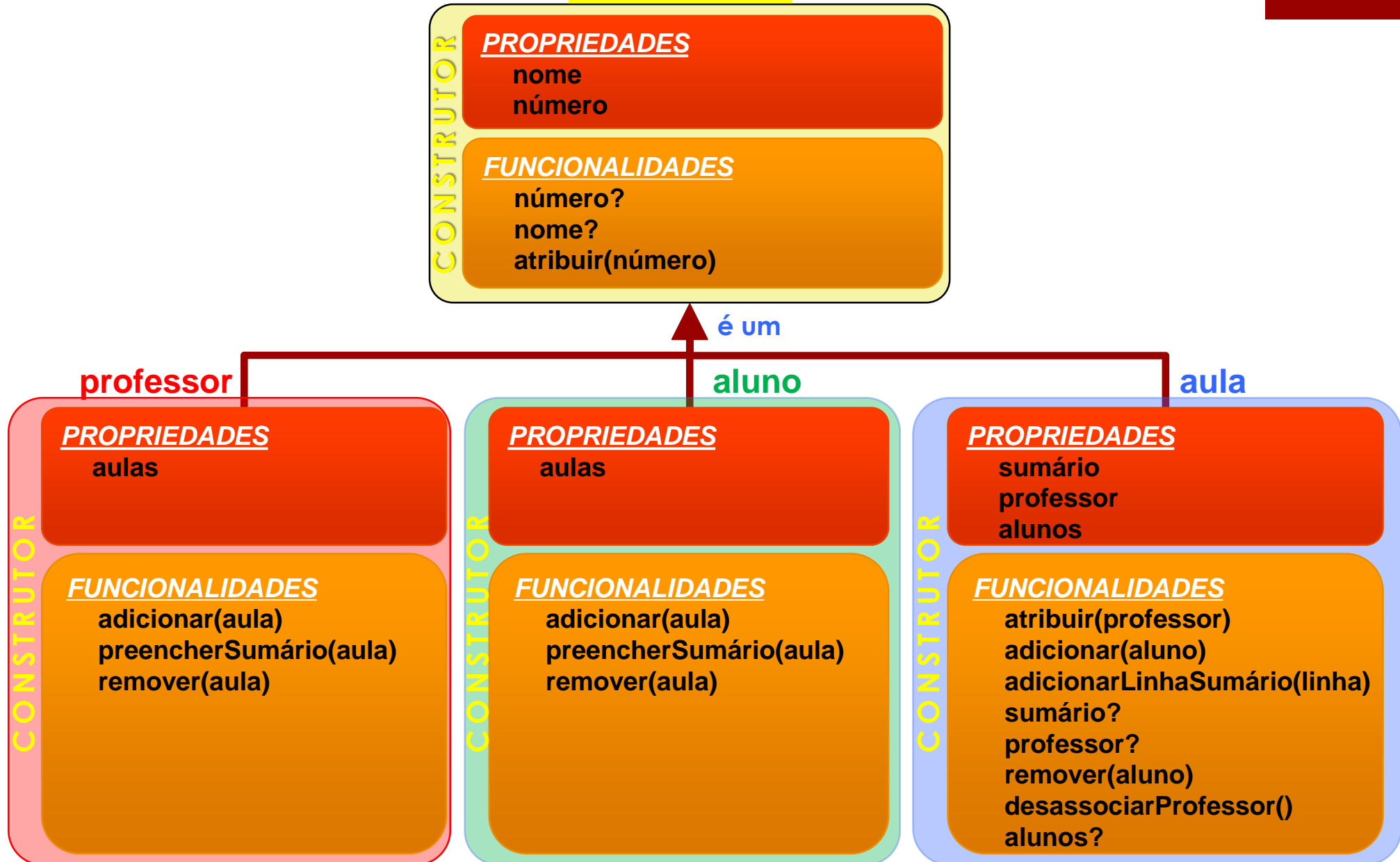




**VAMOS AGORA  
ANALISAR O QUE HÁ  
AINDA EM COMUM**

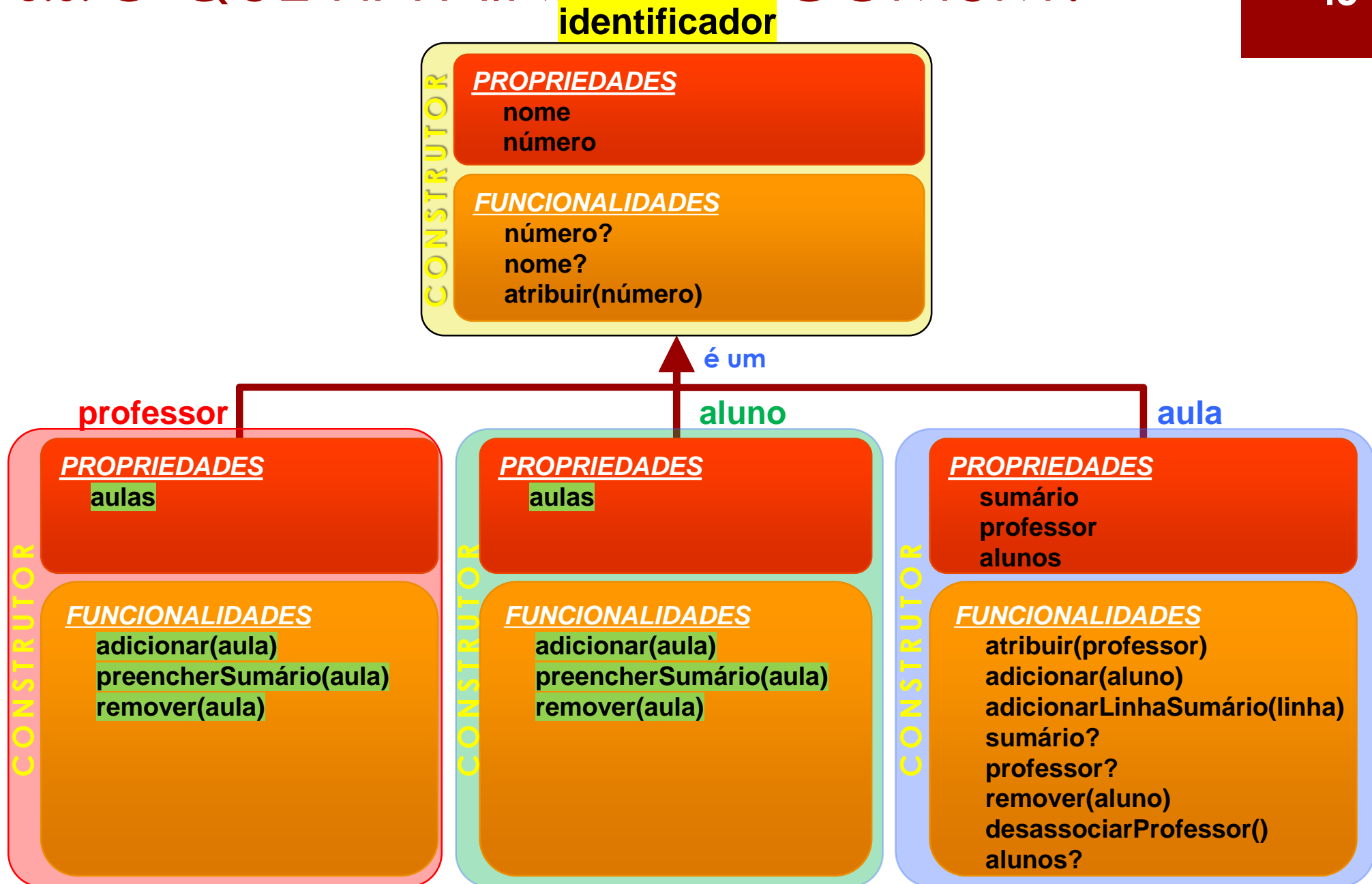
### 3.6. O QUE HÁ AINDA EM COMUM?

42



### 3.6. O QUE HÁ AINDA EM COMUM?

43



## 3.6. O QUE HÁ AINDA EM COMUM?

44

```
public class Professor extends Identificador {  
    ...  
    public void adicionar(Aula aula) {  
        if (aula == null || aulas.contains(aula)) {  
            return;  
        }  
        aulas.add(aula);  
        aula.setProfessor(this);  
    }  
  
    public void remover(Aula aula) {  
        if (!aulas.contains(aula)) {  
            return;  
        }  
        aulas.remove(aula);  
        aula.desassociarProfessor();  
    }  
  
    public void preencherSumario(Aula aula) {  
        if (!aulas.contains(aula)) {  
            return;  
        }  
        aula.adicionarLinhaSumario(aula.getNome());  
        aula.adicionarLinhaSumario(  
            String.valueOf(aula.getNumero());  
        );  
        aula.adicionarLinhaSumario(nome);  
        for (Aluno aluno : aula.getAlunos()) {  
            aluno.preencherSumario(aula);  
        }  
    }  
}
```

```
public class Aluno extends Identificador{  
    ...  
    public void adicionar(Aula aula) {  
        if (aula == null ||  
            aulas.contains(aula)) {  
            return;  
        }  
        aulas.add(aula);  
        aula.adicionar(this);  
    }  
  
    public void remover(Aula aula) {  
        if (!aulas.contains(aula)) {  
            return;  
        }  
        aulas.remove(aula);  
        aula.remover(this);  
    }  
  
    public void preencherSumario(Aula aula) {  
        if (!aulas.contains(aula)) {  
            return;  
        }  
        aula.adicionarLinhaSumario(nome);  
    }  
}
```

CLASSE  
Pessoa

## 3.6. O QUE HÁ AINDA EM COMUM?

45

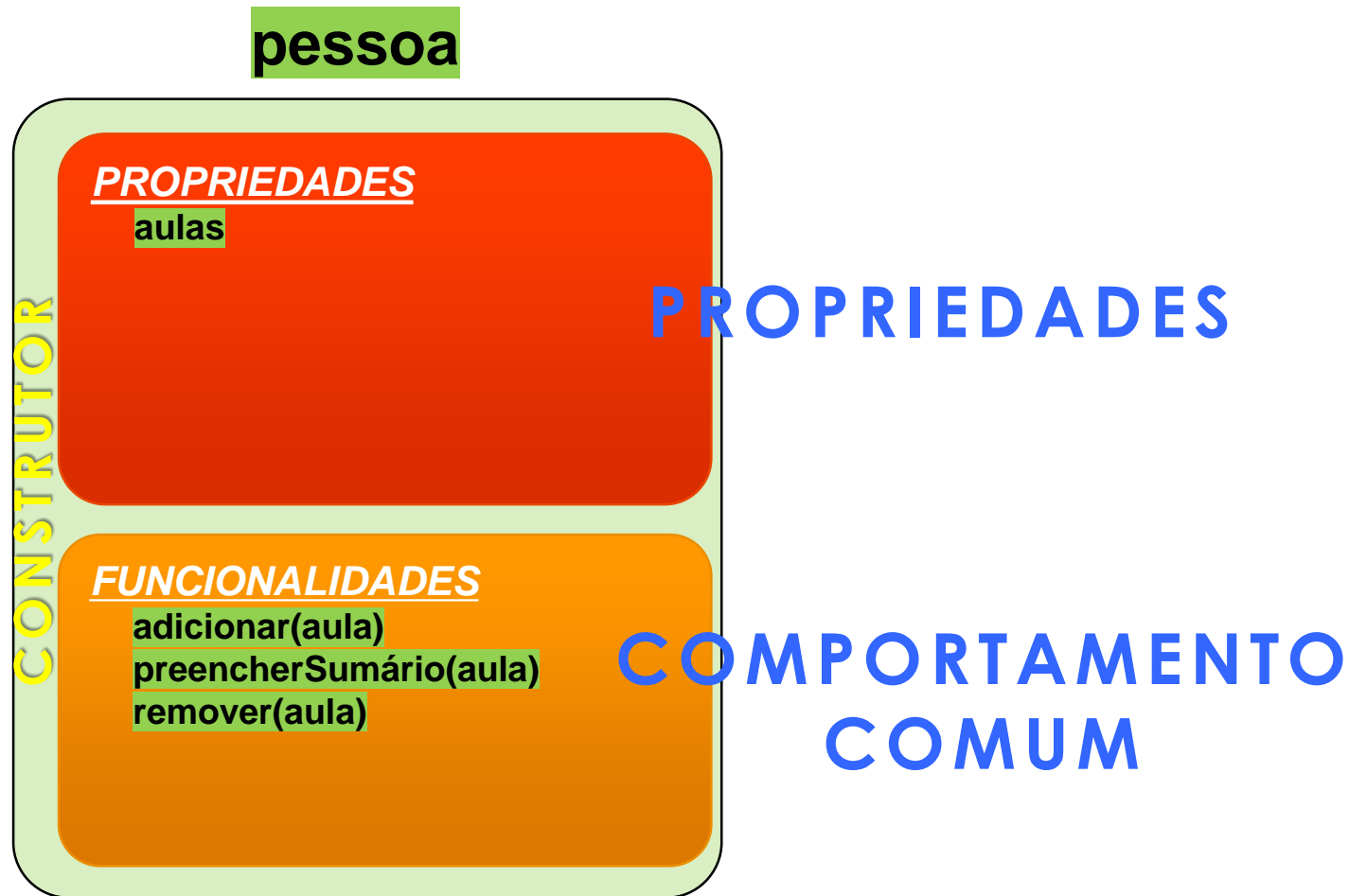
```
public class Professor extends Identificador {  
    ...  
    public void adicionar(Aula aula) {  
        if (aula == null || aulas.contains(aula)) {  
            return;  
        }  
        aulas.add(aula);  
        aula.setProfessor(this);  
    }  
  
    public void remover(Aula aula) {  
        if (!aulas.contains(aula)) {  
            return;  
        }  
        aulas.remove(aula);  
        aula.desassociarProfessor();  
    }  
  
    public void preencherSumario(Aula aula) {  
        if (!aulas.contains(aula)) {  
            return;  
        }  
        aula.adicionarLinhaSumario(aula.getNome());  
        aula.adicionarLinhaSumario(  
            String.valueOf(aula.getNumero());  
        );  
        aula.adicionarLinhaSumario(nome);  
        for (Aluno aluno : aula.getAlunos()) {  
            aluno.preencherSumario(aula);  
        }  
    }  
}
```

```
public class Aluno extends Identificador{  
    ...  
    public void adicionar(Aula aula) {  
        if (aula == null ||  
            aulas.contains(aula)) {  
            return;  
        }  
        aulas.add(aula);  
        aula.adicionar(this);  
    }  
  
    public void remover(Aula aula) {  
        if (!aulas.contains(aula)) {  
            return;  
        }  
        aulas.remove(aula);  
        aula.remover(this);  
    }  
  
    public void preencherSumario(Aula aula) {  
        if (!aulas.contains(aula)) {  
            return;  
        }  
        aula.adicionarLinhaSumario(nome);  
    }  
}
```

COMPORTAMENTO  
DISTINTO  
NÃO PODE SER  
HERDADO

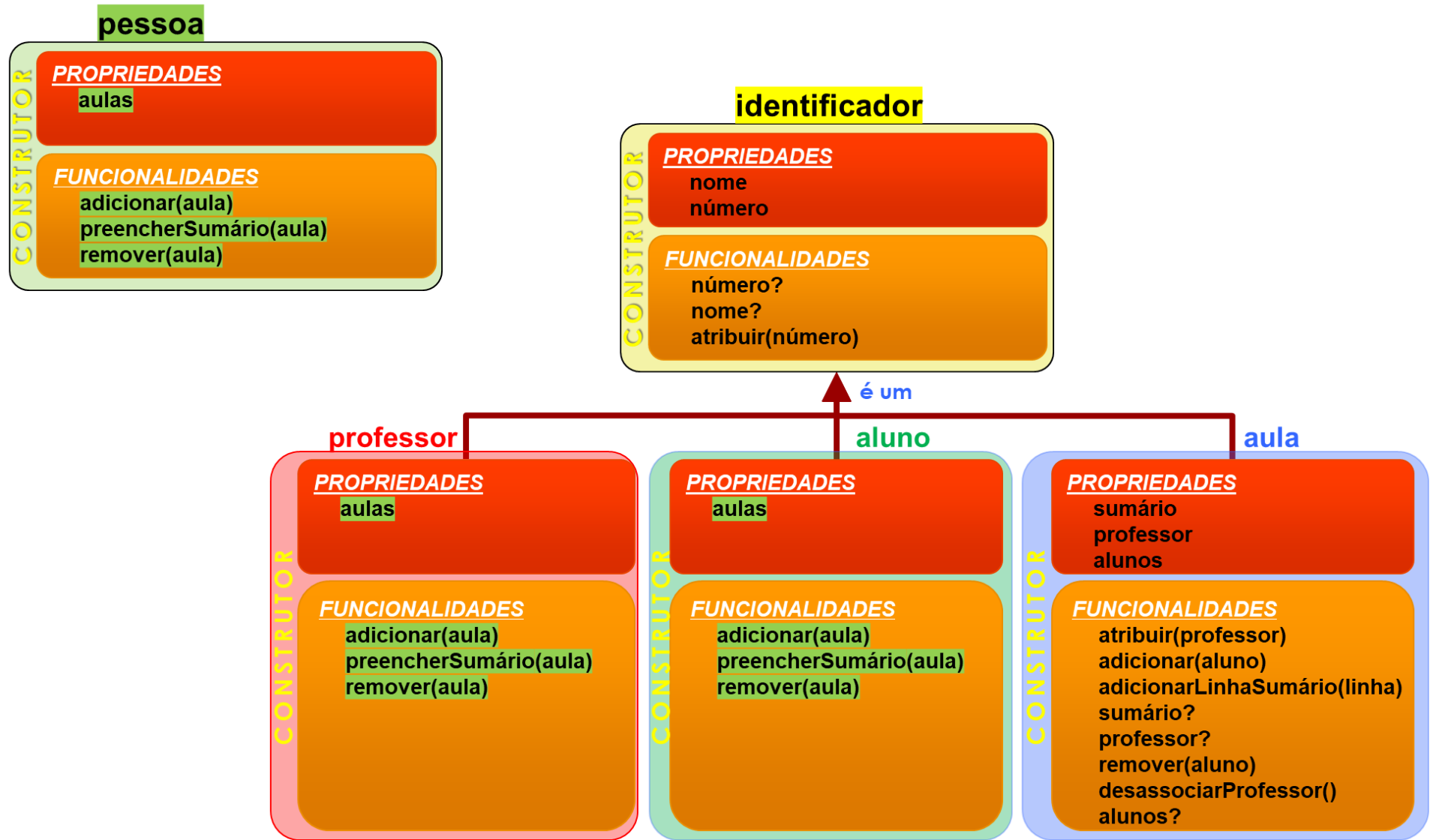
### 3.6. O QUE HÁ AINDA EM COMUM?

46



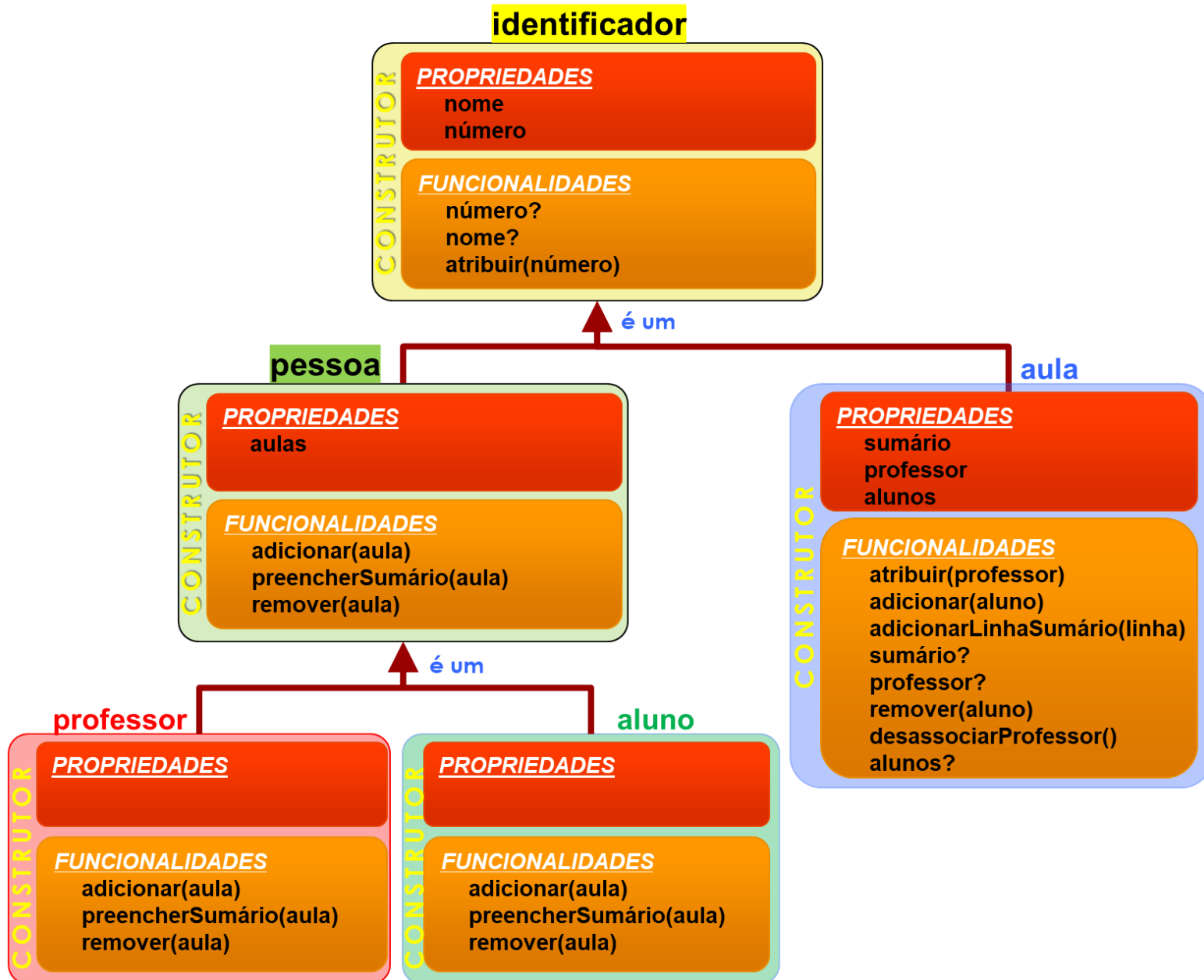
### 3.7. O QUE AINDA PODE SER HERDADO?

47



### 3.7. O QUE AINDA PODE SER HERDADO?

48





**OUTRA ABORDAGEM PARA  
IDENTIFICAR O QUE É  
COMUM EM DIAGRAMAS  
DE CLASSES COMPLEXOS**

## 3.7. O QUE AINDA PODE SER HERDADO?

50

|                   |                               | Identificador | Identificador | Identificador |               |
|-------------------|-------------------------------|---------------|---------------|---------------|---------------|
|                   | superclasses/interfaces       |               |               |               |               |
|                   | classes                       | Professor     | Aluno         | Aula          | Identificador |
| <b>tipo</b>       | <b>atributo</b>               |               |               |               |               |
| String            | nome                          |               |               |               | x             |
| long              | numero                        |               |               |               | x             |
| LinkedList<Aula>  | aulas                         | x             | x             |               |               |
| String            | sumario                       |               |               | x             |               |
| Professor         | professor                     |               |               | x             |               |
| LinkedList<Aluno> | alunos                        |               |               | x             |               |
| <b>return</b>     | <b>método</b>                 |               |               |               |               |
| void              | setProfessor(Professor)       |               |               | x             |               |
| void              | adicionar(Aula)               | x             | x             |               |               |
| void              | preencherSumario(Aula)        | x             | x             |               |               |
| void              | adicionar(Aluno)              |               |               | x             |               |
| void              | adicionarLinhaSumario(String) |               |               | x             |               |
| String            | getNome()                     |               |               |               | x             |
| long              | getNumero()                   |               |               |               | x             |
| void              | setNumero(long)               |               |               |               | x             |
| String            | getSumario()                  |               |               | x             |               |
| Professor         | getProfessor()                |               |               | x             |               |
| void              | desassociarProfessor()        |               |               | x             |               |
| void              | remover(Aula)                 | x             | x             |               |               |
| LinkedList<Aluno> | getAlunos()                   |               |               | x             |               |
| void              | remover(Aluno)                |               |               | x             |               |

COMUM AO  
MAIOR  
NÚMERO DE  
CLASSES A  
HERDAR DA  
MESMA  
SUPERCLASSE?

## 3.7. O QUE AINDA PODE SER HERDADO?

51

|                   | superclasses/interfaces       | Identificador | Identificador | Identificador |               | Identificador |
|-------------------|-------------------------------|---------------|---------------|---------------|---------------|---------------|
|                   | classes                       | Professor     | Aluno         | Aula          | Identificador | Pessoa        |
| <b>tipo</b>       | <b>atributo</b>               |               |               |               |               |               |
| String            | nome                          |               |               |               | x             |               |
| long              | numero                        |               |               |               | x             |               |
| LinkedList<Aula>  | aulas                         | x             | x             |               |               | x             |
| String            | sumario                       |               |               | x             |               |               |
| Professor         | professor                     |               |               | x             |               |               |
| LinkedList<Aluno> | alunos                        |               |               | x             |               |               |
| <b>return</b>     | <b>método</b>                 |               |               |               |               |               |
| void              | setProfessor(Professor)       |               |               | x             |               |               |
| void              | adicionar(Aula)               | x             | x             |               |               | x             |
| void              | preencherSumario(Aula)        | x             | x             |               |               | x             |
| void              | adicionar(Aluno)              |               |               | x             |               |               |
| void              | adicionarLinhaSumario(String) |               |               | x             |               |               |
| String            | getNome()                     |               |               |               | x             |               |
| long              | getNumero()                   |               |               |               | x             |               |
| void              | setNumero(long)               |               |               |               | x             |               |
| String            | getSumario()                  |               |               | x             |               |               |
| Professor         | getProfessor()                |               |               | x             |               |               |
| void              | desassociarProfessor()        |               |               | x             |               |               |
| void              | remover(Aula)                 | x             | x             |               |               | x             |
| LinkedList<Aluno> | getAlunos()                   |               |               | x             |               |               |
| void              | remover(Aluno)                |               |               | x             |               |               |

## 3.7. O QUE AINDA PODE SER HERDADO?

52

|                   | superclasses/interfaces       | Pessoa    | Pessoa | Identificador |               | Identificador |
|-------------------|-------------------------------|-----------|--------|---------------|---------------|---------------|
|                   | classes                       | Professor | Aluno  | Aula          | Identificador | Pessoa        |
| <b>tipo</b>       | <b>atributo</b>               |           |        |               |               |               |
| String            | nome                          |           |        |               | x             |               |
| long              | numero                        |           |        |               | x             |               |
| LinkedList<Aula>  | aulas                         |           |        |               |               | x             |
| String            | sumario                       |           |        | x             |               |               |
| Professor         | professor                     |           |        | x             |               |               |
| LinkedList<Aluno> | alunos                        |           |        | x             |               |               |
| <b>return</b>     | <b>método</b>                 |           |        |               |               |               |
| void              | setProfessor(Professor)       |           |        | x             |               |               |
| void              | adicionar(Aula)               | R         | R      |               |               | x             |
| void              | preencherSumario(Aula)        | R         | R      |               |               | x             |
| void              | adicionar(Aluno)              |           |        | x             |               |               |
| void              | adicionarLinhaSumario(String) |           |        | x             |               |               |
| String            | getNome()                     |           |        |               | x             |               |
| long              | getNumero()                   |           |        |               | x             |               |
| void              | setNumero(long)               |           |        |               | x             |               |
| String            | getSumario()                  |           |        | x             |               |               |
| Professor         | getProfessor()                |           |        | x             |               |               |
| void              | desassociarProfessor()        |           |        | x             |               |               |
| void              | remover(Aula)                 | R         | R      |               |               | x             |
| LinkedList<Aluno> | getAlunos()                   |           |        | x             |               |               |
| void              | remover(Aluno)                |           |        | x             |               |               |

**CASO 2: PARTE DO CÓDIGO PASSA PARA A SUPERCLASSE**

**QUANDO APENAS PARTE DO COMPORTAMENTO DE UM MÉTODO PODE SER HERDADO DIZ-SE QUE É REDEFINIDO (OVERRIDE)**

## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

53

```
public class Pessoa extends Identificador {  
    protected LinkedList<Aula> aulas;
```

ATRIBUTOS COMUNS COM  
ACESSO PROTEGIDO

```
    public Pessoa(String nome, long numero, LinkedList<Aula> aulas) {  
        super(nome, numero);  
        this.aulas = new LinkedList<>();  
        for (Aula aula : aulas) {  
            adicionar(aula);  
        }  
    }  
}
```

```
    public void adicionar(Aula aula) {  
        if (aula == null || aulas.contains(aula)) {  
            return;  
        }  
        aulas.add(aula);  
    }  
}
```

```
    public void remover(Aula aula) {  
        if (!aulas.contains(aula)) {  
            return;  
        }  
        aulas.remove(aula);  
    }  
}
```

```
    public void preencherSumario(Aula aula) {  
        if (!aulas.contains(aula)) {  
            return;  
        }  
    }  
}
```

COMPORTAMENTO  
COMUM

```
}
```

## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

54

```
public class Professor extends Pessoa {
    public Professor(String nome, long numero) {
        this(nome, numero, new LinkedList<>());
    }

    public Professor(String nome, long numero, LinkedList<Aula> aulas) {
        super(nome, numero, aulas);
    }

    @Override
    public void adicionar(Aula aula) {
        super.adicionar(aula);
        aula.setProfessor(this);
    }

    @Override
    public void remover(Aula aula) {
        super.remover(aula);
        aula.desassociarProfessor();
    }

    @Override
    public void preencherSumario(Aula aula) {
        super.preencherSumario(aula);
        aula.adicionarLinhaSumario(aula.getNome());
        aula.adicionarLinhaSumario(String.valueOf(aula.getNumero()));
        aula.adicionarLinhaSumario(nome);
        for (Aluno aluno : aula.getAlunos()) {
            aluno.preencherSumario(aula);
        }
    }
}
```

## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

55

```
public class Professor extends Pessoa {
    public Professor(String nome, long numero) {
        this(nome, numero, new LinkedList<>());
    }

    public Professor(String nome, long numero, LinkedList<Aula> aulas) {
        super(nome, numero, aulas);
    }

    @Override
    public void adicionar(Aula aula) {
        super.adicionar(aula);
        aula.setProfessor(this);
    }

    @Override
    public void remover(Aula aula) {
        super.remover(aula);
        aula.desassociarProfessor();
    }

    @Override
    public void preencherSumario(Aula aula) {
        super.preencherSumario(aula);
        aula.adicionarLinhaSumario(aula.getNome());
        aula.adicionarLinhaSumario(String.valueOf(aula.getNumero()));
        aula.adicionarLinhaSumario(nome);
        for (Aluno aluno : aula.getAlunos()) {
            aluno.preencherSumario(aula);
        }
    }
}
```



The diagram illustrates a code transformation. A red arrow points from the `super.adicionar(aula);` line in the `adicionar` method to a new code block. A red 'X' is placed over the original `super.adicionar(aula);` line, indicating it is to be removed. The new code block, enclosed in a blue box, contains the following lines:

```
super(nome, numero);
this.aulas = new LinkedList<>();
for (Aula aula : aulas) {
    adicionar(aula);
}
```

## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

56

```
public class Professor extends Pessoa {
    public Professor(String nome, long numero) {
        this(nome, numero, new LinkedList<>());
    }

    public Professor(String nome, long numero, LinkedList<Aula> aulas) {
        super(nome, numero, aulas);
    }

    @Override
    public void adicionar(Aula aula) {
        super.adicionar(aula);
        aula.setProfessor(this);
    }

    @Override
    public void remover(Aula aula) {
        super.remover(aula);
        aula.desassociarProfessor();
    }

    @Override
    public void preencherSumario(Aula aula) {
        super.preencherSumario(aula);
        aula.adicionarLinhaSumario(aula.getNome());
        aula.adicionarLinhaSumario(String.valueOf(aula.getNumero()));
        aula.adicionarLinhaSumario(nome);
        for (Aluno aluno : aula.getAlunos()) {
            aluno.preencherSumario(aula);
        }
    }
}
```

HERDADO DA CLASSE Identificador

nome  
numero  
getNome()  
getNumero()  
setNumero(long)

HERDADO DA CLASSE Pessoa

aulas  
adicionar(Aula)  
remover(Aula)  
preencherSumario(Aula)

é um





## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

57

```
public class Professor extends Pessoa {  
    public Professor(String nome, long numero) {  
        this(nome, numero, new LinkedList<>());  
    }  
  
    public Professor(String nome, long numero, LinkedList<Aula> aulas) {  
        super(nome, numero, aulas);  
    }  
  
    @Override  
    public void adicionar(Aula aula) {  
        super.adicionar(aula);  
        aula.setProfessor(this);  
    }  
  
    @Override  
    public void remover(Aula aula) {  
        super.remover(aula);  
        aula.desassociarProfessor();  
    }  
  
    @Override  
    public void preencherSumario(Aula aula) {  
        super.preencherSumario(aula);  
        aula.adicionarLinhaSumario(aula.getNome());  
        aula.adicionarLinhaSumario(String.valueOf(aula.getNumero()));  
        aula.adicionarLinhaSumario(nome);  
        for (Aluno aluno : aula.getAlunos()) {  
            aluno.preencherSumario(aula);  
        }  
    }  
}
```

HERDADO DA CLASSE Identificador

nome  
numero  
getNome()  
getNumero()  
setNumero(long)

HERDADO DA CLASSE Pessoa

aulas  
adicionar(Aula)  
remover(Aula)  
preencherSumario(Aula)

é um

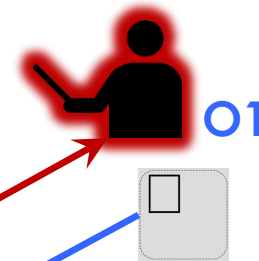
QUANDO SE REDEFINE UM MÉTODO  
PODE SER INVOCADO O MÉTODO  
ORIGINAL DEFINIDO NA  
SUPERCLASSE ATRAVÉS DO `super`

## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

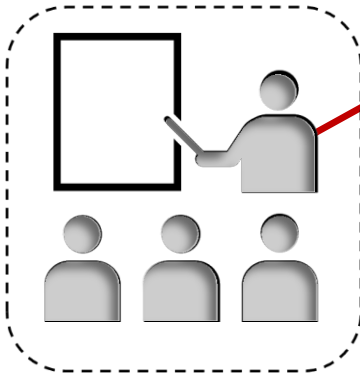
58

Pessoa

```
public void adicionar(Aula aula) {  
    if (aula == null || aulas.contains(aula)) {  
        return;  
    }  
    aulas.add(aula);  
}
```



O9



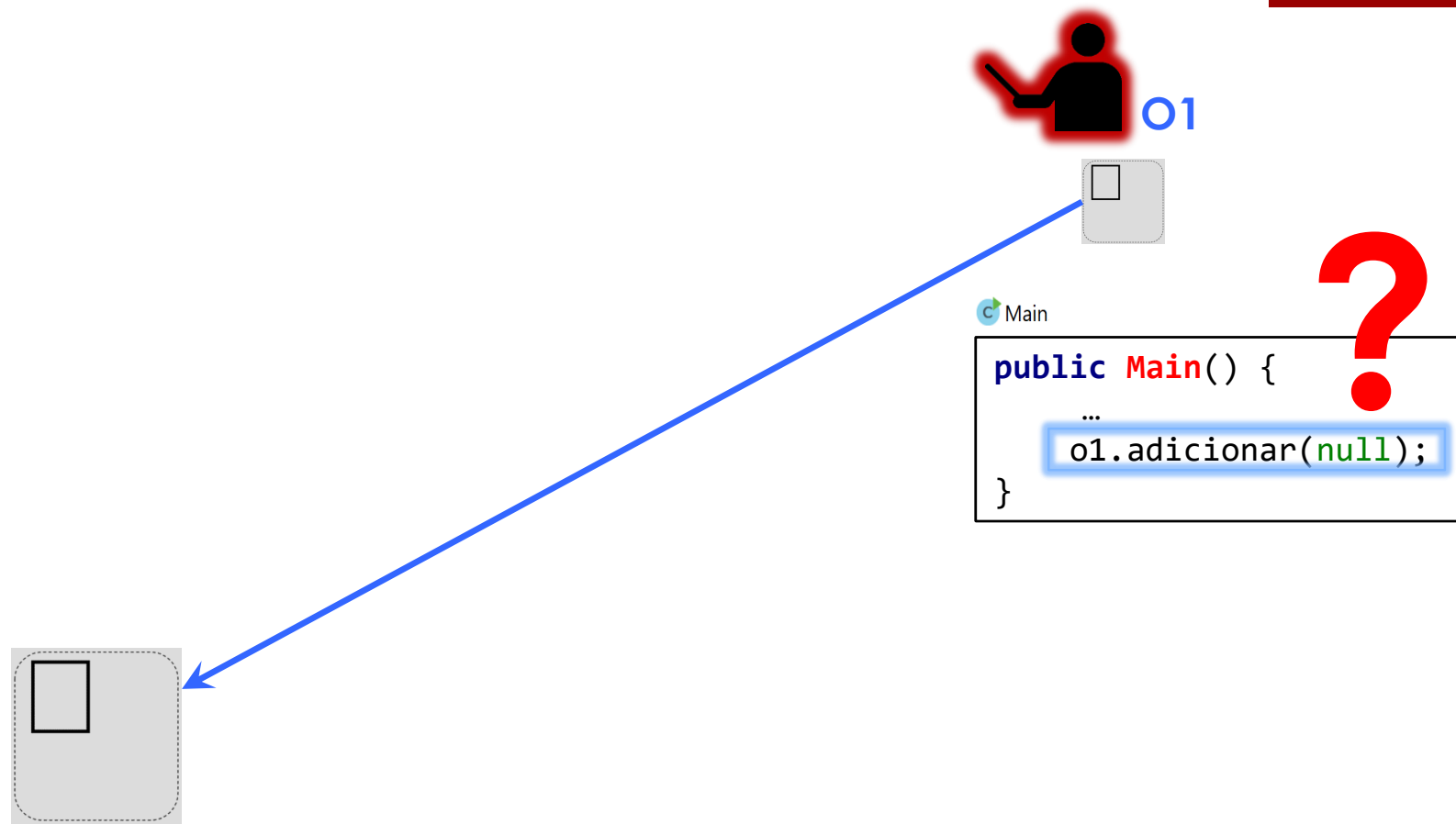
Professor

```
@Override  
public void adicionar(Aula aula) {  
    super.adicionar(aula);  
    aula.setProfessor(this);  
}
```

QUANDO SE REDEFINE UM MÉTODO PODE SER  
INVOCADO O MÉTODO ORIGINAL DEFINIDO NA  
SUPERCLASSE ATRAVÉS DO **super**

## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

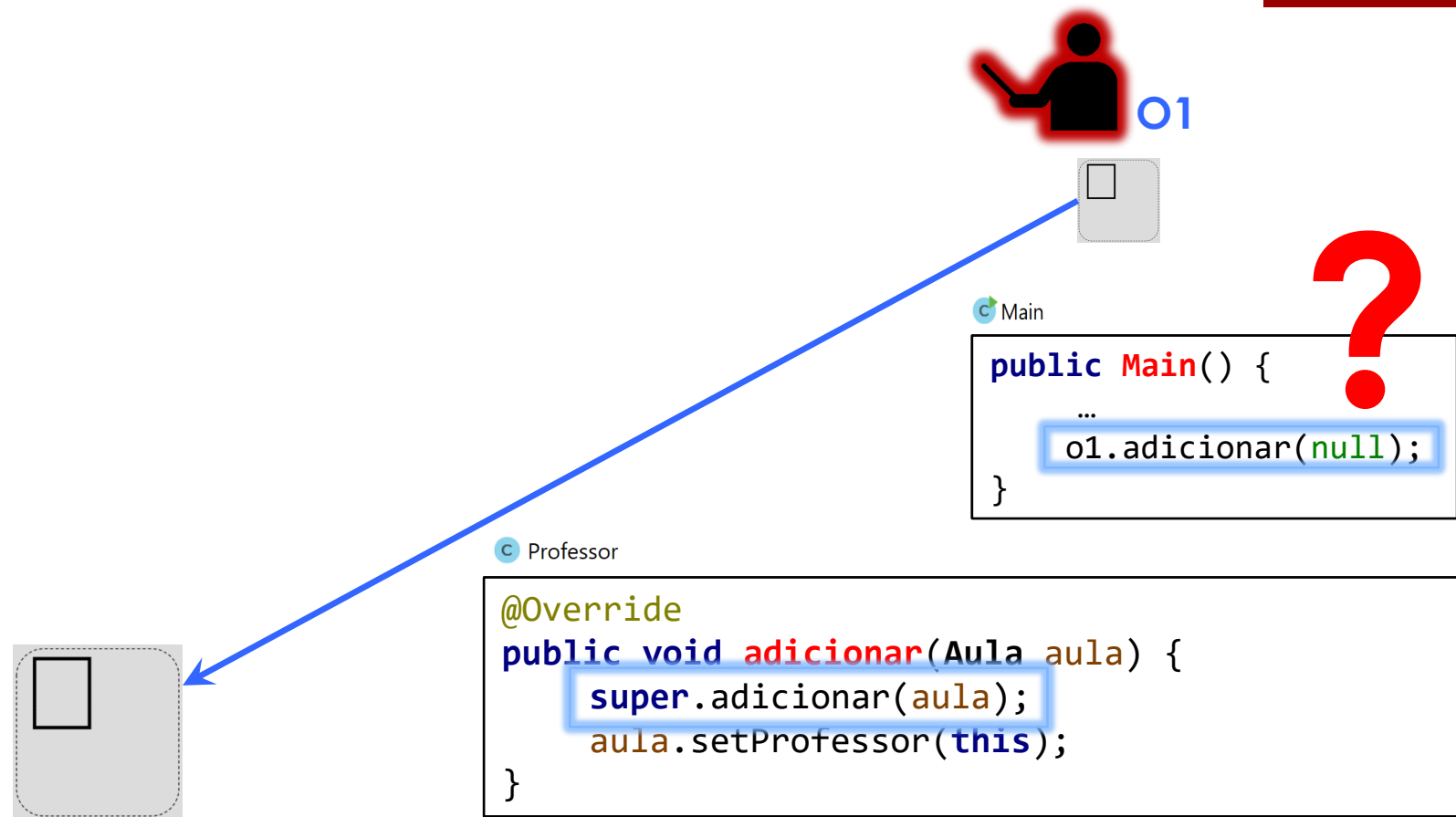
59



**E O QUE ACONTECE SE ADICIONARMOS A UM PROFESSOR UMA AULA ATRAVÉS DE UMA REFERÊNCIA NULA?**

## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

60



E O QUE ACONTECE SE ADICIONARMOS A UM PROFESSOR UMA AULA ATRAVÉS DE UMA REFERÊNCIA NULA?

# 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

61

C Pessoa

```
public void adicionar(Aula aula) {  
    if (aula == null || aulas.contains(aula)) {  
        return;  
    }  
    aulas.add(aula);  
}
```

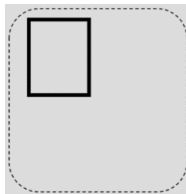


C Main

```
public Main() {  
    ...  
    o1.adicionar(null);  
}
```

C Professor

```
@Override  
public void adicionar(Aula aula) {  
    super.adicionar(aula);  
    aula.setProfessor(this);  
}
```



## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

62

C Pessoa

```
public void adicionar(Aula aula) {  
    if (aula == null || aulas.contains(aula)) {  
        return;  
    }  
    aulas.add(aula);  
}
```



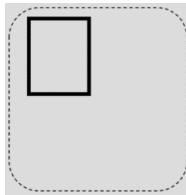
C Main

```
public Main() {  
    ...  
    o1.adicionar(null);  
}
```



C Professor

```
@Override  
public void adicionar(Aula aula) {  
    super.adicionar(aula);  
    aula.setProfessor(this);  
}
```



## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

63

C Pessoa

```
public void adicionar(Aula aula) {  
    if (aula == null || aulas.contains(aula)) {  
        return;  
    }  
    aulas.add(aula);  
}
```



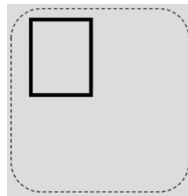
C Main

```
public Main() {  
    ...  
    o1.adicionar(null);  
}
```



C Professor

```
@Override  
public void adicionar(Aula aula) {  
    super.adicionar(aula);  
    aula.setProfessor(this);  
}
```



## 3.8. EXTRAÇÃO DE OUTRA SUPERCLASSE

64

C Pessoa

```
public void adicionar(Aula aula) {  
    if (aula == null || aulas.contains(aula)) {  
        return;  
    }  
    aulas.add(aula);  
}
```



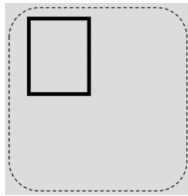
C Main

```
public Main() {  
    ...  
    o1.adicionar(null);  
}
```



C Professor

```
@Override  
public void adicionar(Aula aula) {  
    super.adicionar(aula);  
    aula.setProfessor(this);  
}
```



**ERRO - APESAR DA PROTEÇÃO NA CLASSE  
PESSOA, CONTINUA A SER EXECUTADO O  
CÓDIGO DISTINTO DO PROFESSOR**



**VAMOS AGORA  
CORRIGIR O ERRO**

## 3.9. CLASSES ABSTRATAS

66

SEMPRE QUE NÃO SE CONSEGUE MANTER O FLUXO DA EXECUÇÃO ANTES DA EXTRAÇÃO DEVE CRIAR-SE, NA SUPERCLASSE, UM NOVO MÉTODO ABSTRATO QUE DEFINA O CONCEITO DO COMPORTAMENTO DISTINTO

QUALQUER CLASSE QUE TENHA PELO MENOS UM MÉTODO ABSTRATO TEM QUE OBRIGATORIAMENTE SER ABSTRATA

A CONCRETIZAÇÃO DO CONCEITO ABSTRATO TEM QUE SER IMPLEMENTADA NAS SUAS SUBCLASSES NÃO ABSTRATAS (OU POR HERANÇA)

## 3.10. EXTRAÇÃO DE OUTRA SUPERCLASSE - CORRIGIDO

67

```
import java.util.LinkedList;
```

```
public abstract class Pessoa extends Identificador {  
    protected LinkedList<Aula> aulas;
```

ATRIBUTOS COMUNS  
COM ACESSO  
PROTEGIDO

```
    public Pessoa(String nome, long numero, LinkedList<Aula> aulas) {  
        super(nome, numero);  
        this.aulas = new LinkedList<>();  
        for (Aula aula : aulas) {  
            adicionar(aula);  
        }  
    }
```

```
    public void adicionar(Aula aula) {  
        if (aula == null || aulas.contains(aula)) {  
            return;  
        }  
        aulas.add(aula);  
        associar(aula);  
    }
```

COMPORTAMENTO  
COMUM

```
    protected abstract void associar(Aula aula);
```

COMPORTAMENTO DISTINTO  
A SER IMPLEMENTADO NAS  
SUBCLASSES

...

...

```
public void remover(Aula aula) {  
    if (!aulas.contains(aula)) {  
        return;  
    }  
    aulas.remove(aula);  
    desassociar(aula);  
}
```

COMPORTAMENTO  
COMUM

```
protected abstract void desassociar(Aula aula);
```

COMPORTAMENTO A  
SER IMPLEMENTADO  
NAS SUBCLASSES

```
public void preencherSumario(Aula aula) {  
    if (!aulas.contains(aula)) {  
        return;  
    }  
    escreverSumario(aula);  
}
```

COMPORTAMENTO  
COMUM

```
protected abstract void escreverSumario(Aula aula);
```

COMPORTAMENTO A  
SER IMPLEMENTADO  
NAS SUBCLASSES

}

## 3.10. EXTRAÇÃO DE OUTRA SUPERCLASSE - CORRIGIDO

69

```
import java.util.LinkedList;

public class Professor extends Pessoa {
    public Professor(String nome, long numero) {
        this(nome, numero, new LinkedList<>());
    }

    public Professor(String nome, long numero, LinkedList<Aula> aulas) {
        super(nome, numero, aulas);
    }

    @Override
    protected void associar(Aula aula) {
        aula.setProfessor(this);
    }

    @Override
    protected void desassociar(Aula aula) {
        aula.desassociarProfessor();
    }

    @Override
    protected void escreverSumario(Aula aula) {
        aula.adicionarLinhaSumario(aula.getNome());
        aula.adicionarLinhaSumario(String.valueOf(aula.getNumero()));
        aula.adicionarLinhaSumario(nome);
        for (Aluno aluno : aula.getAlunos()) {
            aluno.preencherSumario(aula);
        }
    }
}
```

## 3.10. EXTRAÇÃO DE OUTRA SUPERCLASSE - CORRIGIDO

70

```
import java.util.LinkedList;
```

```
public class Professor extends Pessoa {  
    public Professor(String nome, long numero) {  
        this(nome, numero, new LinkedList<>());  
    }
```

```
    public Professor(String nome, long numero, LinkedList<Aula> aulas) {  
        super(nome, numero, aulas);  
    }
```

```
@Override  
protected void associar(Aula aula) {  
    aula.setProfessor(this);  
}
```

```
@Override  
protected void desassociar(Aula aula) {  
    aula.desassociarProfessor();  
}
```

```
@Override  
protected void escreverSumario(Aula aula) {  
    aula.adicionarLinhaSumario(aula.getNome());  
    aula.adicionarLinhaSumario(String.valueOf(aula.getNumero()));  
    aula.adicionarLinhaSumario(nome);  
    for (Aluno aluno : aula.getAlunos()) {  
        aluno.preencherSumario(aula);  
    }  
}
```

HERDADO DA CLASSE Identificador

nome  
numero  
getNome()  
getNumero()  
setNumero(long)

é um

HERDADO DA CLASSE Pessoa

aulas  
adicionar(Aula)  
remover(Aula)  
preencherSumario(Aula)  
associar(Aula aula)  
desassociar(Aula aula)  
escreverSumario(Aula aula)

## 3.10. EXTRAÇÃO DE OUTRA SUPERCLASSE - CORRIGIDO

71

```
import java.util.LinkedList;

public class Aluno extends Pessoa {
    public Aluno(String nome, long numero) {
        this(nome, numero, new LinkedList<>());
    }

    public Aluno(String nome, long numero, LinkedList<Aula> aulas) {
        super(nome, numero, aulas);
    }

    @Override
    protected void associar(Aula aula) {
        aula.adicionar(this);
    }

    @Override
    protected void desassociar(Aula aula) {
        aula.remover(this);
    }

    @Override
    protected void escreverSumario(Aula aula) {
        aula.adicionarLinhaSumario(nome);
    }
}
```

## 3.10. EXTRAÇÃO DE OUTRA SUPERCLASSE - CORRIGIDO

72

```
import java.util.LinkedList;
```

```
public class Aluno extends Pessoa {  
    public Aluno(String nome, long numero) {  
        this(nome, numero, new LinkedList<>());  
    }
```

```
    public Aluno(String nome, long numero, LinkedList<Aula> aulas) {  
        super(nome, numero, aulas);  
    }
```

```
    @Override  
    protected void associar(Aula aula) {  
        aula.adicionar(this);  
    }
```

```
    @Override  
    protected void desassociar(Aula aula) {  
        aula.remover(this);  
    }
```

```
    @Override  
    protected void escreverSumario(Aula aula) {  
        aula.adicionarLinhaSumario(nome);  
    }  
}
```

HERDADO DA CLASSE Identificador

nome  
numero  
getNome()  
getNumero()  
setNumero(long)

é um

HERDADO DA CLASSE Pessoa

aulas  
adicionar(Aula)  
remover(Aula)  
preencherSumario(Aula)  
associar(Aula aula)  
desassociar(Aula aula)  
escreverSumario(Aula aula)



## 3.11. O QUE HÁ AINDA EM COMUM?

73

```
public class Professor extends Pessoa {  
    ...  
  
    @Override  
    protected void escreverSumario(Aula aula) {  
        aula.adicionarLinhaSumario(aula.getNome());  
        aula.adicionarLinhaSumario(  
            String.valueOf(aula.getNumero());  
        );  
        aula.adicionarLinhaSumario(nome);  
        for (Aluno aluno : aula.getAlunos()) {  
            aluno.preencherSumario(aula);  
        }  
    }  
}
```

```
public class Aluno extends Pessoa {  
    ...  
  
    @Override  
    protected void escreverSumario(Aula aula){  
        aula.adicionarLinhaSumario(nome);  
    }  
}
```

COMPORTAMENTO  
COMUM

## 3.11. O QUE HÁ AINDA EM COMUM?

74

```
public class Professor extends Pessoa {  
    ...  
  
    @Override  
    protected void escreverSumario(Aula aula) {  
        aula.adicionarLinhaSumario(aula.getNome());  
        aula.adicionarLinhaSumario(  
            String.valueOf(aula.getNumero());  
        assinarSumario(aula);  
        for (Aluno aluno : aula.getAlunos()) {  
            aluno.preencherSumario(aula);  
        }  
    }  
}
```

```
public class Aluno extends Pessoa {  
    ...  
  
    @Override  
    protected void escreverSumario(Aula aula){  
        assinarSumario(aula);  
    }  
}
```

```
public abstract class Pessoa extends Identificador {  
    ...  
  
    protected void assinarSumario(Aula aula) {  
        aula.adicionarLinhaSumario(nome);  
    }  
}
```

# UMA CLASSE ABSTRATA NÃO PODE SER INSTANCIADA

## EXISTEM CLASSES ABSTRATAS:

```
public abstract class Pessoa extends Identificador {  
    ...  
    protected abstract void associar(Aula aula);  
    protected abstract void desassociar(Aula aula);  
    protected abstract void escreverSumario(Aula aula);  
}
```

POR IMPOSIÇÃO

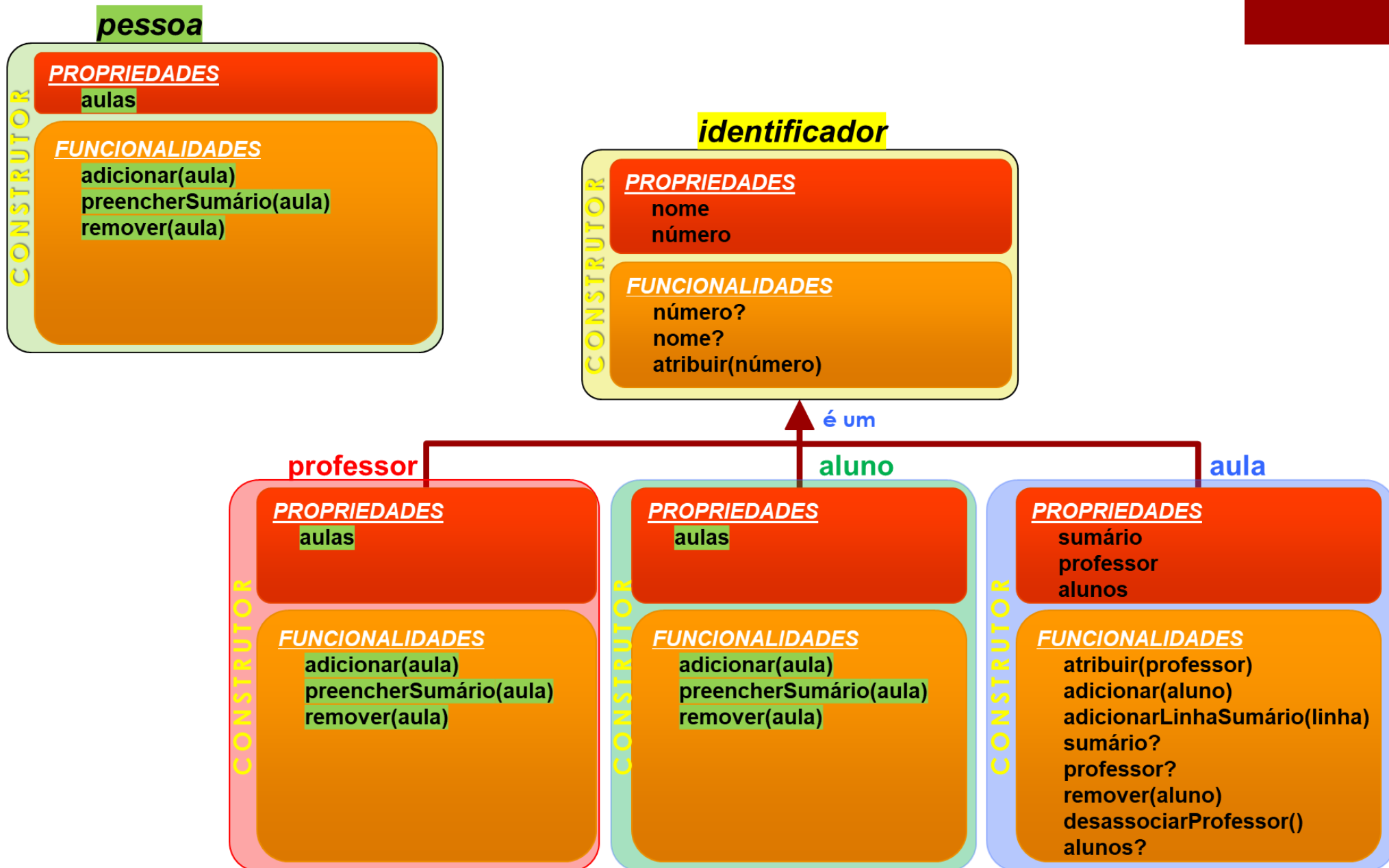
```
public abstract class Identificador {  
    ...  
}
```

POR OPÇÃO

# HERANÇA COM CORREÇÃO

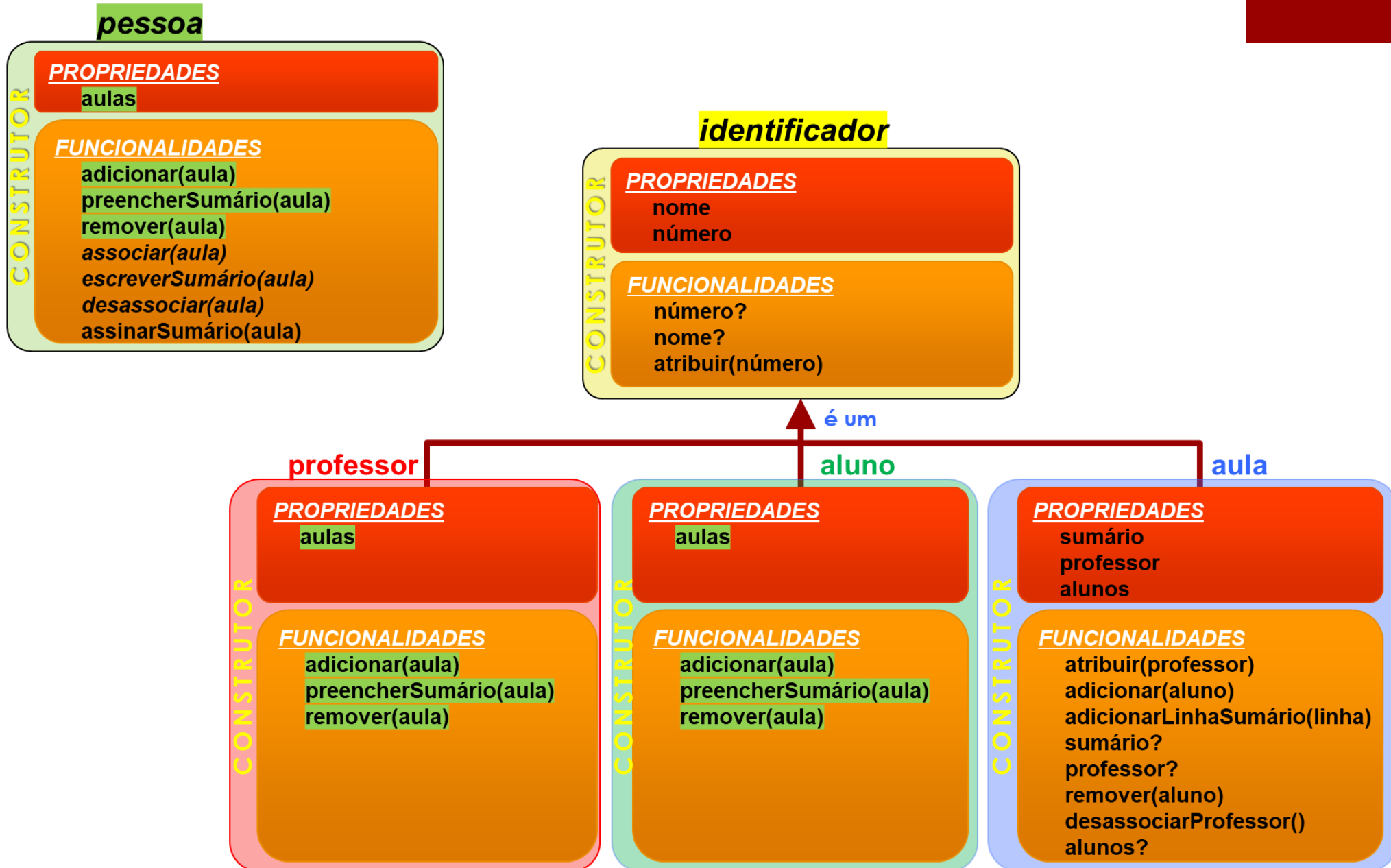
### 3.13. O QUE AINDA PODE SER HERDADO? - CORRIGIDO

77



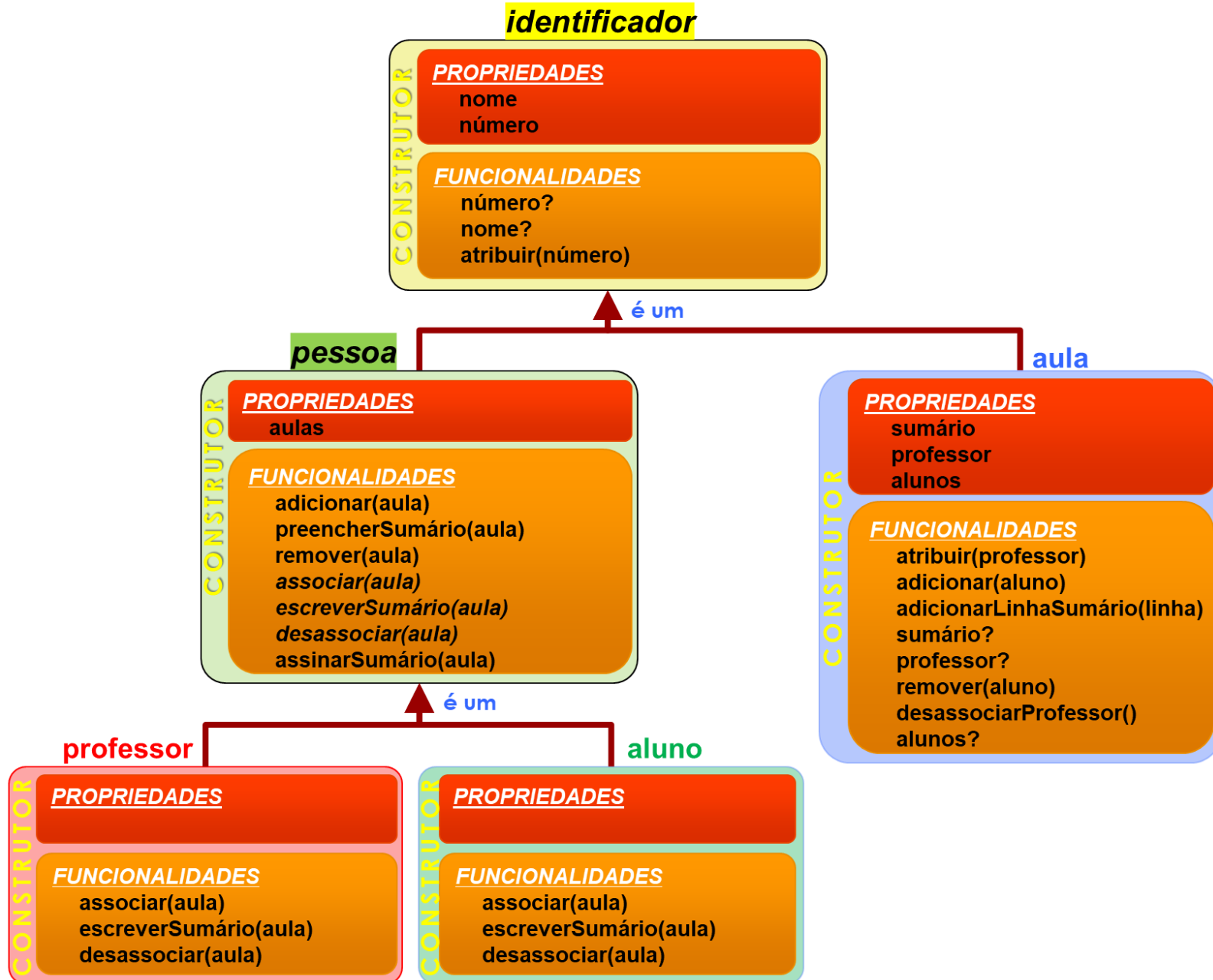
### 3.13. O QUE AINDA PODE SER HERDADO? - CORRIGIDO

78



### 3.13. O QUE AINDA PODE SER HERDADO? - CORRIGIDO

79



### 3.13. O QUE AINDA PODE SER HERDADO? - CORRIGIDO

80

|                   |                               | Identificador | Identificador | Identificador |               |
|-------------------|-------------------------------|---------------|---------------|---------------|---------------|
|                   | superclasses/interfaces       |               |               |               |               |
|                   | classes                       | Professor     | Aluno         | Aula          | Identificador |
| <b>tipo</b>       | <b>atributo</b>               |               |               |               |               |
| String            | nome                          |               |               |               | x             |
| long              | numero                        |               |               |               | x             |
| LinkedList<Aula>  | aulas                         | x             | x             |               |               |
| String            | sumario                       |               |               | x             |               |
| Professor         | professor                     |               |               | x             |               |
| LinkedList<Aluno> | alunos                        |               |               | x             |               |
| <b>return</b>     | <b>método</b>                 |               |               |               |               |
| void              | setProfessor(Professor)       |               |               | x             |               |
| void              | adicionar(Aula)               | x             | x             |               |               |
| void              | preencherSumario(Aula)        | x             | x             |               |               |
| void              | adicionar(Aluno)              |               |               | x             |               |
| void              | adicionarLinhaSumario(String) |               |               | x             |               |
| String            | getNome()                     |               |               |               | x             |
| long              | getNumero()                   |               |               |               | x             |
| void              | setNumero(long)               |               |               |               | x             |
| String            | getSumario()                  |               |               | x             |               |
| Professor         | getProfessor()                |               |               | x             |               |
| void              | desassociarProfessor()        |               |               | x             |               |
| void              | remover(Aula)                 | x             | x             |               |               |
| LinkedList<Aluno> | getAlunos()                   |               |               | x             |               |
| void              | remover(Aluno)                |               |               | x             |               |

COMUM AO  
MAIOR  
NÚMERO DE  
CLASSES A  
HERDAR DA  
MESMA  
SUPERCLASSE?



### 3.13. O QUE AINDA PODE SER HERDADO? - CORRIGIDO

81

|                   |                               | Identificador | Identificador | Identificador |               |        |
|-------------------|-------------------------------|---------------|---------------|---------------|---------------|--------|
|                   | superclasses/interfaces       |               |               |               |               |        |
|                   | classes                       | Professor     | Aluno         | Aula          | Identificador | Pessoa |
| <i>tipo</i>       | <i>atributo</i>               |               |               |               |               |        |
| String            | nome                          |               |               |               | x             |        |
| long              | numero                        |               |               |               | x             |        |
| LinkedList<Aula>  | aulas                         | x             | x             |               |               | x      |
| String            | sumario                       |               |               | x             |               |        |
| Professor         | professor                     |               |               | x             |               |        |
| LinkedList<Aluno> | alunos                        |               |               | x             |               |        |
| <i>return</i>     | <i>método</i>                 |               |               |               |               |        |
| void              | setProfessor(Professor)       |               |               | x             |               |        |
| void              | adicionar(Aula)               | x             | x             |               |               | x      |
| void              | preencherSumario(Aula)        | x             | x             |               |               | x      |
| void              | adicionar(Aluno)              |               |               | x             |               |        |
| void              | adicionarLinhaSumario(String) |               |               | x             |               |        |
| String            | getNome()                     |               |               |               | x             |        |
| long              | getNumero()                   |               |               |               | x             |        |
| void              | setNumero(long)               |               |               |               | x             |        |
| String            | getSumario()                  |               |               | x             |               |        |
| Professor         | getProfessor()                |               |               | x             |               |        |
| void              | desassociarProfessor()        |               |               | x             |               |        |
| void              | remover(Aula)                 | x             | x             |               |               | x      |
| LinkedList<Aluno> | getAlunos()                   |               |               | x             |               |        |
| void              | remover(Aluno)                |               |               | x             |               |        |

### 3.13. O QUE AINDA PODE SER HERDADO? - CORRIGIDO

82

## CASO 3: MÉTODO ABSTRATO NA SUPERCLASSE

|                   |                               | superclasses/interfaces |        |               |               |
|-------------------|-------------------------------|-------------------------|--------|---------------|---------------|
|                   |                               | Pessoa                  | Pessoa | Identificador | Identificador |
|                   | classes                       | Professor               | Aluno  | Aula          |               |
| tipo              | atributo                      |                         |        |               |               |
| String            | nome                          |                         |        | x             |               |
| long              | numero                        |                         |        | x             |               |
| LinkedList<Aula>  | aulas                         |                         |        |               | x             |
| String            | sumario                       |                         |        | x             |               |
| Professor         | professor                     |                         |        | x             |               |
| LinkedList<Aluno> | alunos                        |                         |        | x             |               |
| return            | método                        |                         |        |               |               |
| void              | setProfessor(Professor)       |                         |        | x             |               |
| void              | adicionar(Aula)               |                         |        |               | x             |
| void              | preencherSumario(Aula)        |                         |        |               | x             |
| void              | adicionar(Aluno)              |                         |        | x             |               |
| void              | adicionarLinhaSumario(String) |                         |        | x             |               |
| String            | getNome()                     |                         |        | x             |               |
| long              | getNumero()                   |                         |        | x             |               |
| void              | setNumero(long)               |                         |        | x             |               |
| String            | getSumario()                  |                         |        | x             |               |
| Professor         | getProfessor()                |                         |        | x             |               |
| void              | desassociarProfessor()        |                         |        | x             |               |
| void              | remover(Aula)                 |                         |        |               | x             |
| LinkedList<Aluno> | getAlunos()                   |                         |        | x             |               |
| void              | remover(Aluno)                |                         |        | x             |               |
| void              | associar(Aula)                |                         |        |               | x             |
| void              | desassociar(Aula)             |                         |        |               | x             |
| void              | escreverSumario(Aula)         |                         |        |               | x             |
| void              | assinarSumario(Aula)          |                         |        |               | x             |

## 3.13. O QUE AINDA PODE SER HERDADO? - CORRIGIDO

83

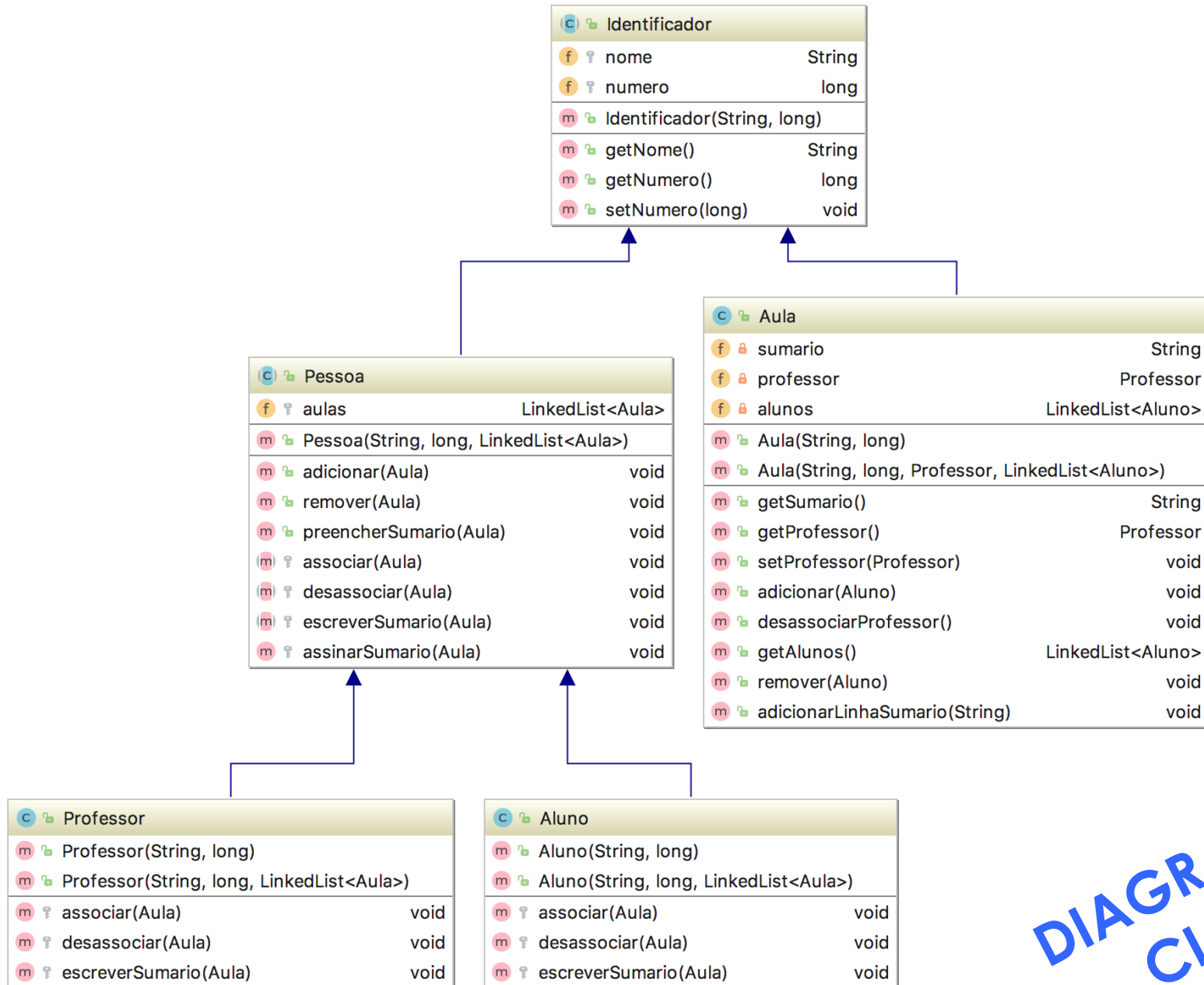


DIAGRAMA DE  
CLASSES

## 3.14. EXEMPLO

84

 Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```


## 3.14. EXEMPLO

85

 Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

o1

 Professor


## 3.14. EXEMPLO

86

 Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

o1

 Professor

## 3.14. EXEMPLO

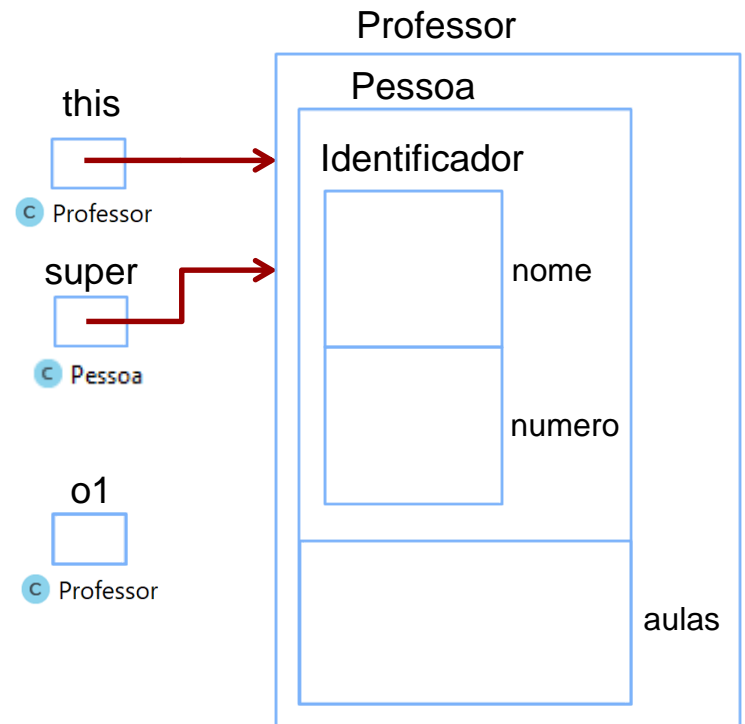
87

Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```



# 3.14. EXEMPLO

88

Main

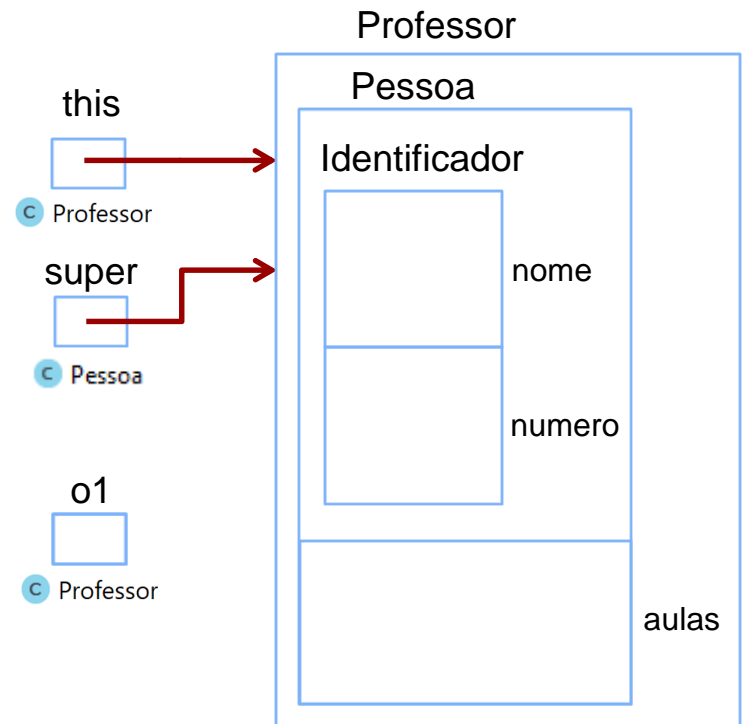
```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

Professor

```
public Professor(String nome, long numero,  
                  LinkedList<Aula> aulas) {  
    super(nome, numero, aulas);  
}
```





# 3.14. EXEMPLO

89

Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

Professor

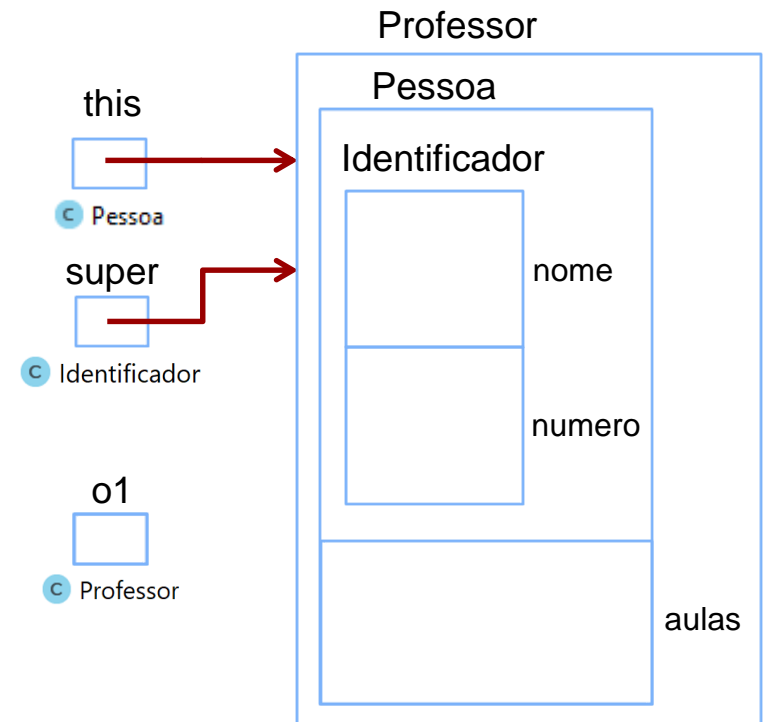
```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

Professor

```
public Professor(String nome, long numero,  
                 LinkedList<Aula> aulas) {  
    super(nome, numero, aulas);  
}
```

Pessoa

```
public Pessoa(String nome, long numero,  
              LinkedList<Aula> aulas) {  
    super(nome, numero);  
    this.aulas = new LinkedList<>();  
    for (Aula aula : aulas) {  
        adicionar(aula);  
    }  
}
```



# 3.14. EXEMPLO

90

Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

Professor

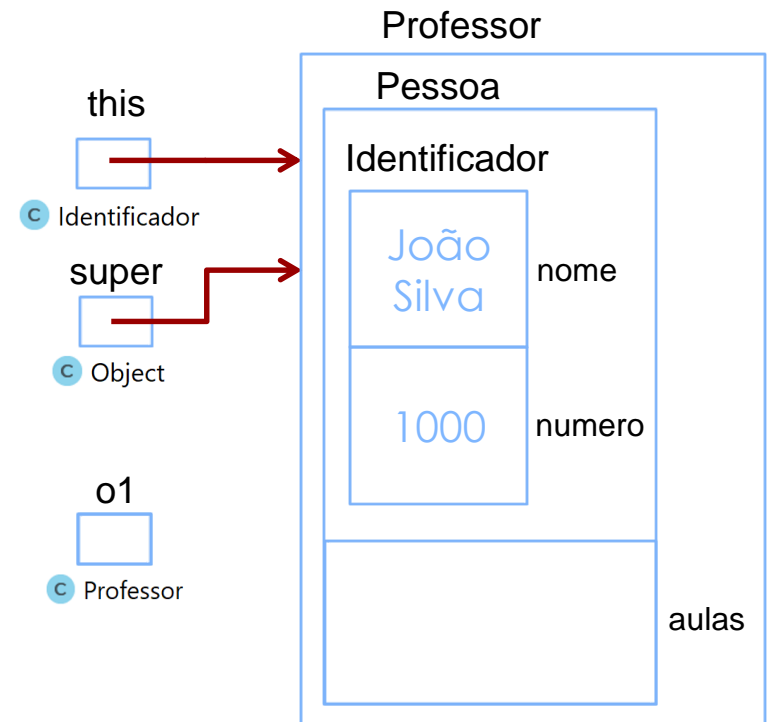
```
public Professor(String nome, long numero,  
                 LinkedList<Aula> aulas) {  
    super(nome, numero, aulas);  
}
```

Pessoa

```
public Pessoa(String nome, long numero,  
              LinkedList<Aula> aulas) {  
    super(nome, numero);  
    this.aulas = new LinkedList<>();  
    for (Aula aula : aulas) {  
        adicionar(aula);  
    }  
}
```

Identificador

```
public Identificador(String nome, long numero) {  
    this.nome = nome;  
    this.numero = numero;  
}
```



# 3.14. EXEMPLO

91

Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

Professor

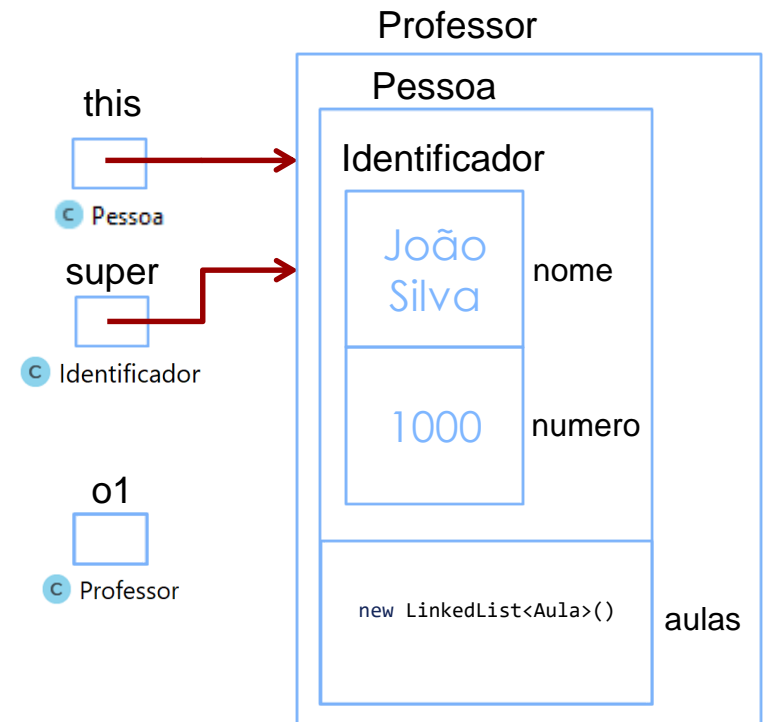
```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

Professor

```
public Professor(String nome, long numero,  
                 LinkedList<Aula> aulas) {  
    super(nome, numero, aulas);  
}
```

Pessoa

```
public Pessoa(String nome, long numero,  
              LinkedList<Aula> aulas) {  
    super(nome, numero);  
    this.aulas = new LinkedList<>();  
    for (Aula aula : aulas) {  
        adicionar(aula);  
    }  
}
```



## 3.14. EXEMPLO

92

Main

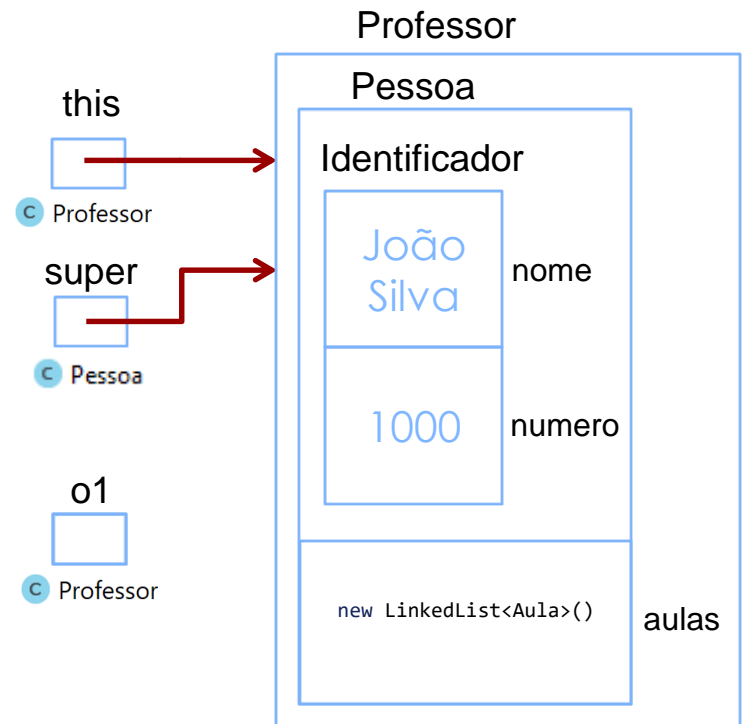
```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

Professor

```
public Professor(String nome, long numero,  
                  LinkedList<Aula> aulas) {  
    super(nome, numero, aulas);  
}
```



## 3.14. EXEMPLO

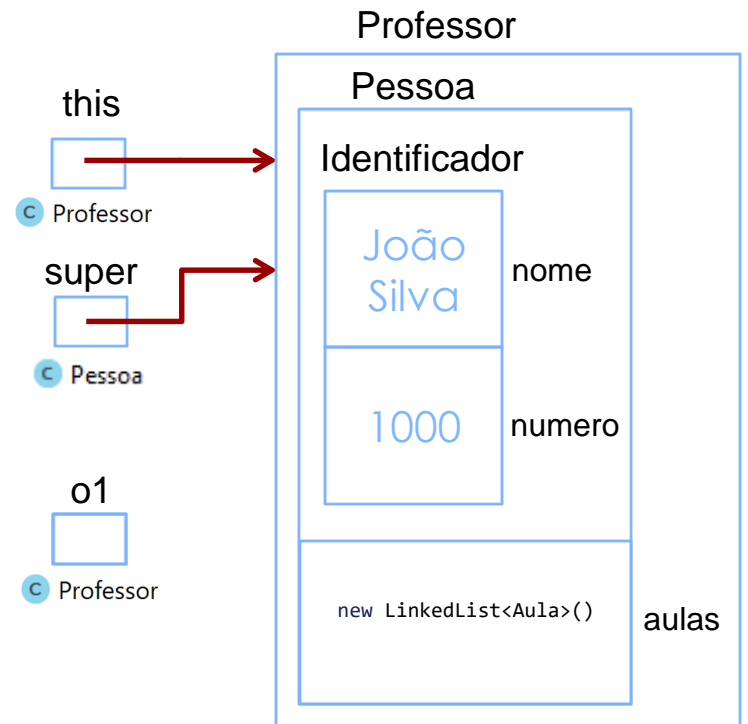
93

Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

Professor

```
public Professor(String nome, long numero) {  
    this(nome, numero, new LinkedList<>());  
}
```

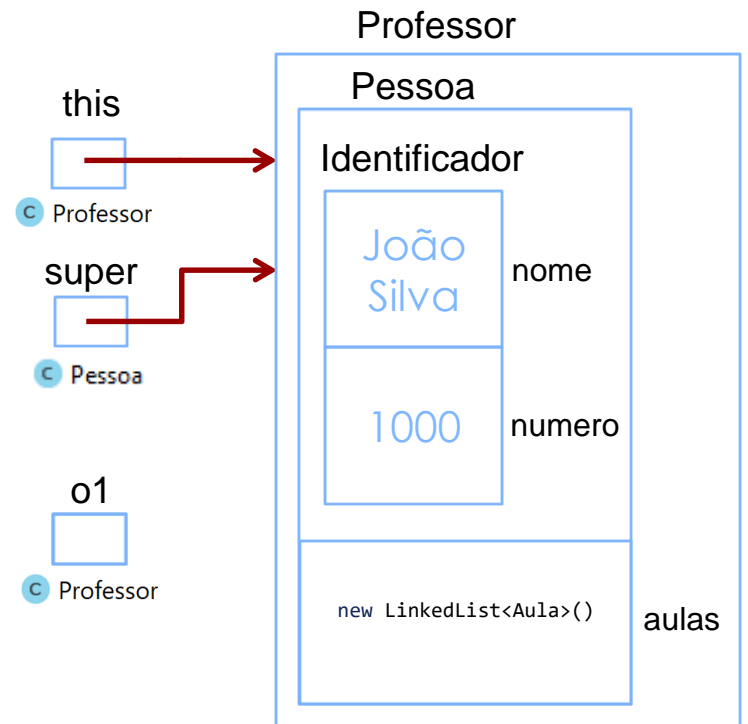


## 3.14. EXEMPLO

94

Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

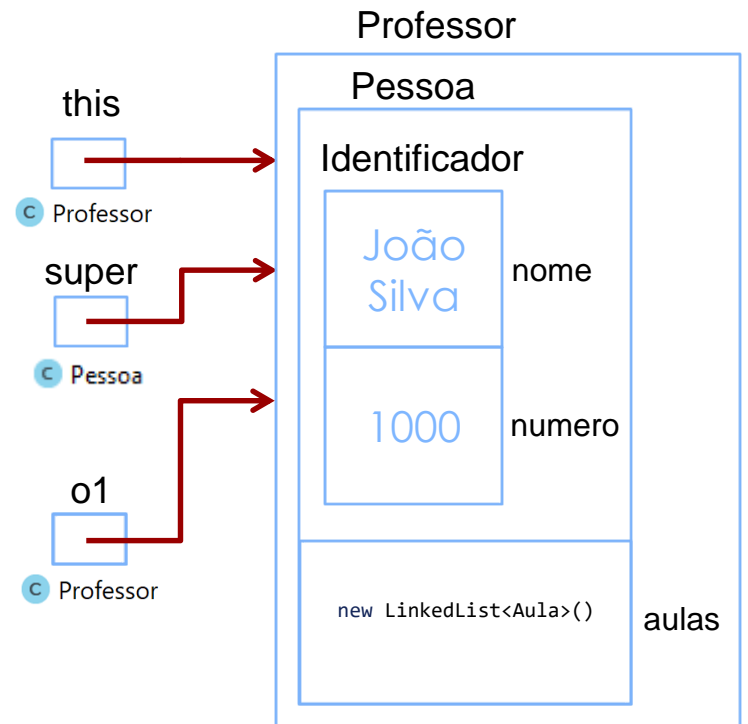


## 3.14. EXEMPLO

95

Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```

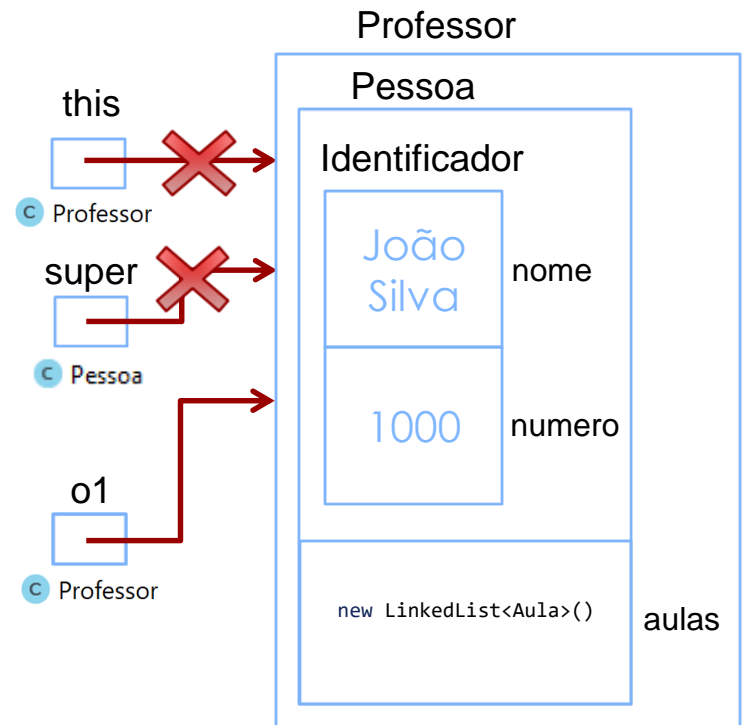


# 3.14. EXEMPLO

96

Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```



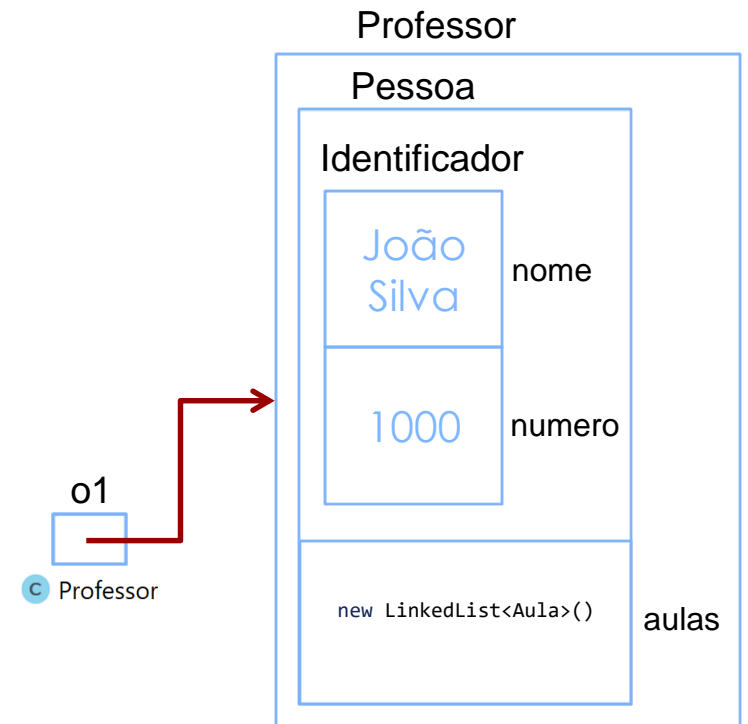


# 3.14. EXEMPLO

97

Main

```
public Main() {  
    Professor o1 = new Professor("João Silva", 1000);  
}
```



### 3.15. REFERÊNCIAS `this` E `super`

98

É UMA REFERÊNCIA PARA O OBJETO AO SER CONSTRUÍDO OU PARA O OBJETO AO QUAL SE APLICA UM MÉTODO DE UMA DADA CLASSE

A REFERÊNCIA `this` É SEMPRE USADA IMPLICITAMENTE (OU EXPLICITAMENTE SEMPRE QUE SE PRETENDA) PARA INVOCAR MÉTODOS OU ACEDER A ATRIBUTOS DE UMA CLASSE OU UMA DAS SUAS SUPERCLASSES

AO INVOCAR UM MÉTODO NUMA DADA CLASSE A PARTIR DA REFERÊNCIA `this`:

- 1- É EFETUADA A VERIFICAÇÃO SE NESSA CLASSE OU NUMA DAS SUAS SUPERCLASSES EXISTE ESSE MÉTODO
- 2- SE 1 OBTIVE SUCESSO, COMEÇA A PROCURA DESSE MÉTODO NA CLASSE DO OBJETO REFERENCIADO POR `this`

A PROCURA DE UM DADO MÉTODO OU ATRIBUTO É INICIADA NUMA DADA CLASSE E PERCORRE TODA A HIERARQUIA DE FORMA ASCENDENTE A COMEÇAR NESSA CLASSE

`this`

## 3.15. REFERÊNCIAS `this` E `super`

99



É UMA REFERÊNCIA PARA O OBJETO REFERENCIADO POR `this` CONSIDERANDO APENAS A SUPERCLASSE DA CLASSE ONDE É INVOCADO O `super`

A REFERÊNCIA `super` É SEMPRE USADA EXPLICITAMENTE PARA INVOCAR MÉTODOS OU ACEDER A ATRIBUTOS DE UMA SUPERCLASSE

AO INVOCAR UM MÉTODO NUMA DADA CLASSE A PARTIR DA REFERÊNCIA `super` COMEÇA A PROCURA DESSE MÉTODO NA SUPERCLASSE DESSA CLASSE

A PROCURA DE UM DADO MÉTODO OU ATRIBUTO É INICIADA NUMA DADA CLASSE E PERCORRE TODA A HIERARQUIA DE FORMA ASCENDENTE A COMEÇAR NESSA CLASSE