

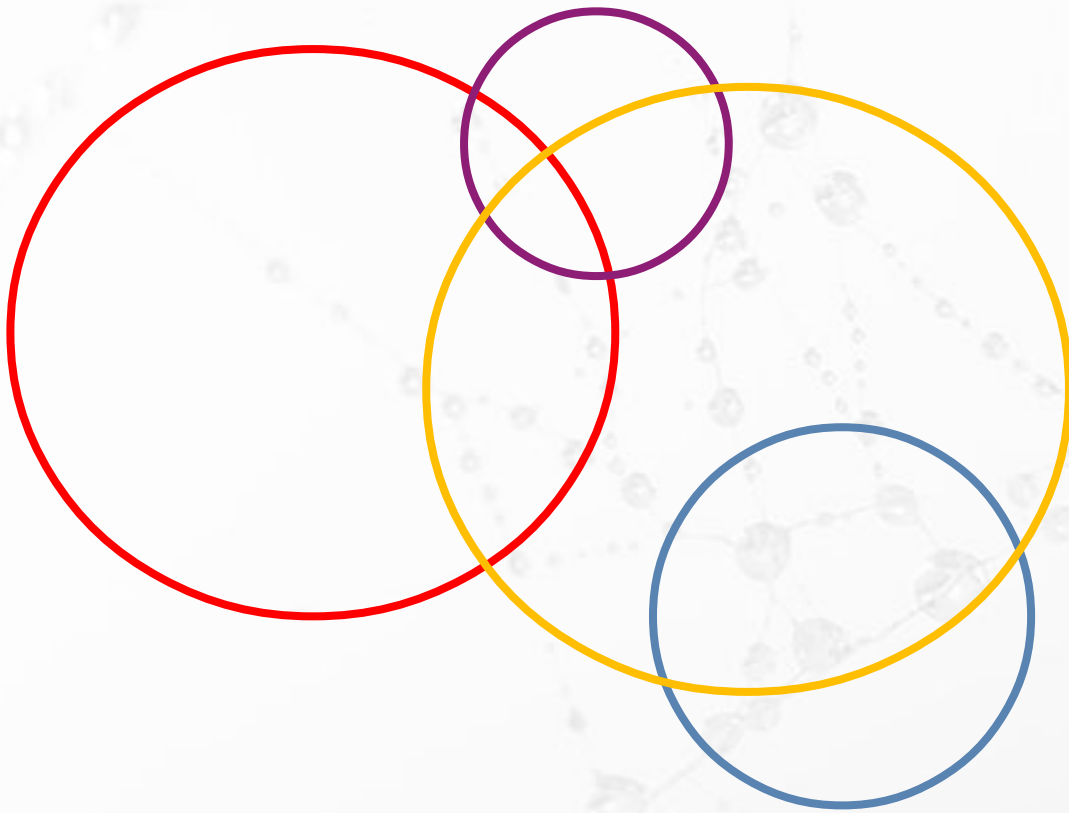
Programação Orientada a Objetos

Relacionamento entre Objetos e Classes

André Santanchè
Institute of Computing – UNICAMP
Março 2020

Retomando os Círculos

Classe e Objetos Círculo



Círculo

centroX: int

centroY: int

raio: int

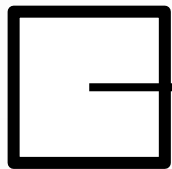
mostraArea()

Variável com Ponteiro

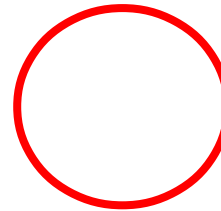
```
Circulo circ = new Circulo();  
circ.centroX = 5;  
circ.centroY = 3;  
circ.raio = 10;  
circ.mostraArea();
```

variável

circ



Círculo (1)



centroX: 5
centroY: 3
raio: 10

Objeto Identidade e Ponteiro

- Cada objeto tem uma identidade única
- Essa identidade é usada para alcançarmos o objeto
- Dizemos que uma variável tem um ponteiro para um objeto quando ela armazena essa identidade

Mostrando a Identidade

```
1 package pt.c02oo.s03relacionamento.s01circulo01;  
2  
3 public class AppCirculo06 {  
4     public static void main(String args[]) {  
5         Circulo circ = new Circulo(5, 3, 10);  
6  
7         System.out.println("Identidade: " + circ);  
8     }  
9 }  
10
```

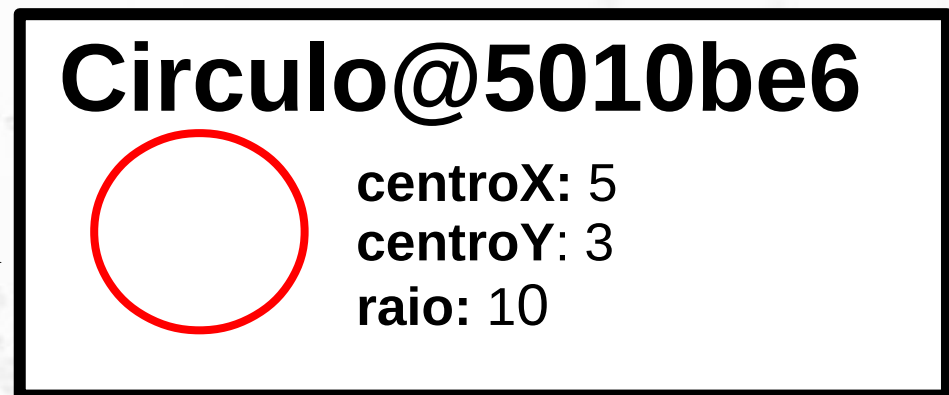
Problems @ Javadoc Declaration Console

<terminated> AppCirculo06 [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (29 de mar de 2020 20:04:36 – 20:04:37)

Identidade: pt.c02oo.s03relacionamento.s01circulo01.Circulo@5010be6

Variável com Ponteiro

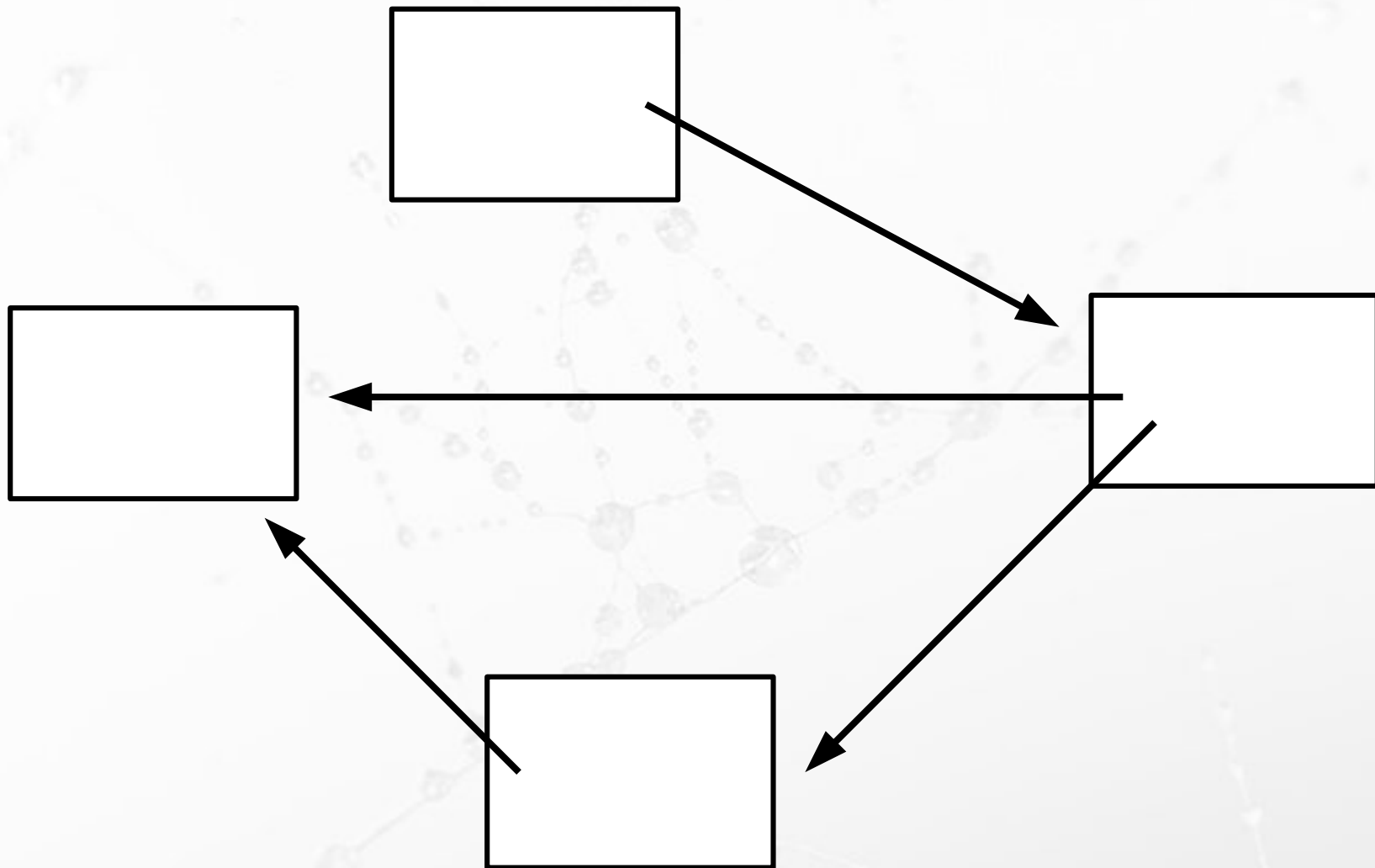
variável
`circ`



Como fazer dois objetos se comunicarem?

Como fazer dois objetos se comunicarem?
Por que?

Aplicação: objetos que atuam em conjunto (se comunicam)



Exemplo

Plotando um gráfico de potência

Exemplo

Objeto

Potencia

1, 2, 4, 8, 16, 32

Produz uma sequência
exponencial de
números.

Exemplo

Objeto

Potencia

1, 2, 4, 8, 16, 32

Produz uma sequência
exponencial de
números.

Objeto

Grafico


```
#  
##  
####  
#####  
#####
```

Plota um gráfico de
uma sequência de
números.

Exemplo

Objeto

Potencia



1, 2, 4, 8, 16, 32

Produz uma sequência
exponencial de
números.

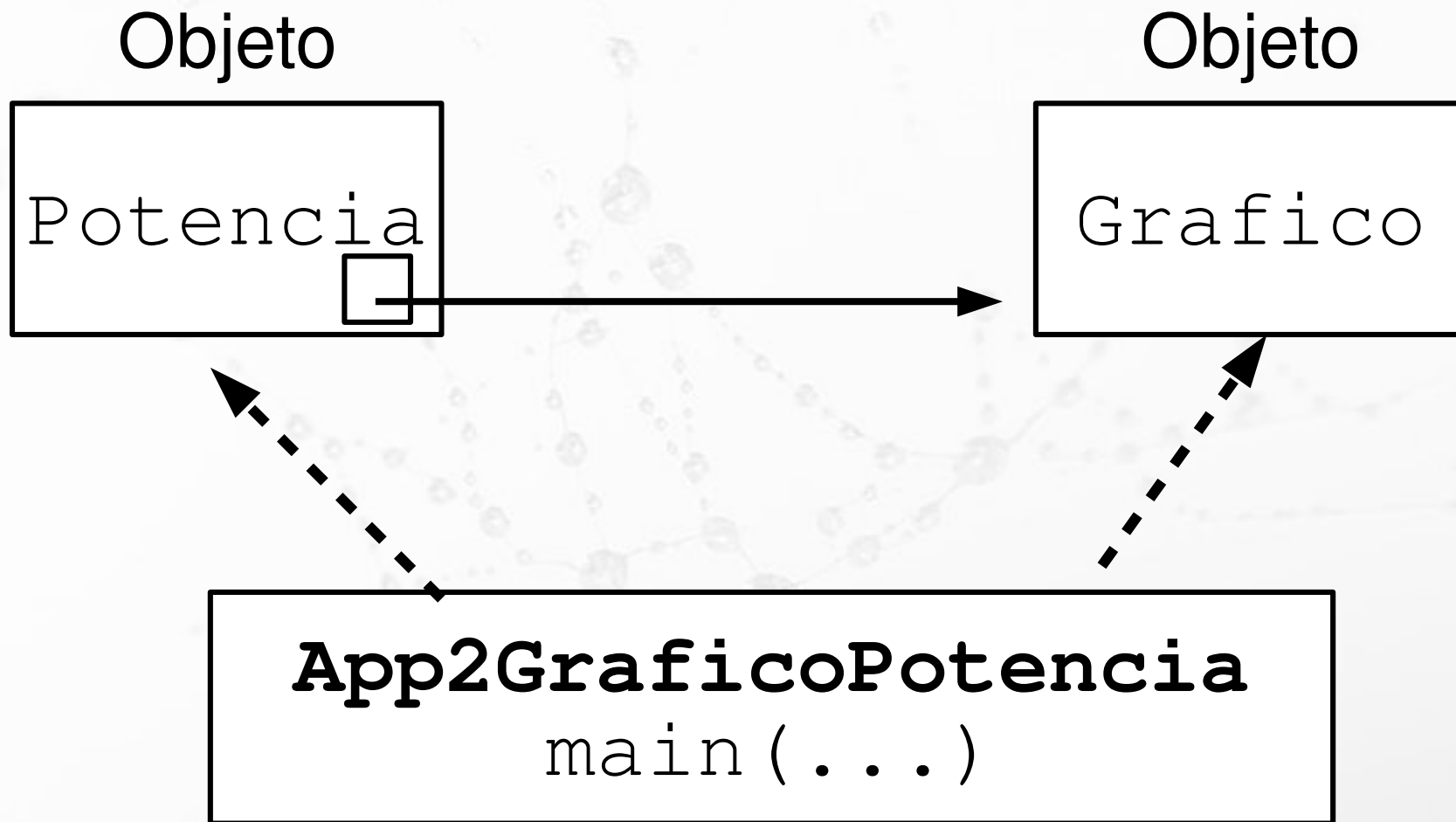
Objeto

Grafico

```
#  
##  
####  
#####  
#####
```

Plota um gráfico de
uma sequência de
números.

Exemplo



Cria e conecta objetos das classes, bem como ativa a sua ação.

Exemplo

Objeto

Grafico

preenchido: **true**
caractere: **#**

```
Grafico g1 = new Grafico(true, '#');
```


Exemplo

Objeto

Potencia

inicial: **1**
quantidade: **7**
saida ☐

Objeto

Grafico


preenchido: **true**
caractere: **#**

```
Grafico g1 = new Grafico(true, '#');  
Potencia p = new Potencia(1, 7);
```

Exemplo

Objeto

Potencia

inicial: 1
quantidade: 7
saida 

Objeto

Grafico

preenchido: **true**
caractere: **#**

```
Grafico g1 = new Grafico(true, '#');  
Potencia p = new Potencia(1, 7);  
p.conecta(g1);
```

Exemplo

Objeto

Potencia

inicial: 1
quantidade: 7
saida ☐

Objeto

Grafico

preenchido: **true**
caractere: **#**

```
Grafico g1 = new Grafico(true, '#');  
Potencia p = new Potencia(1, 7);  
p.conecta(g1);  
p.apresenta();  
#  
##  
####  
#####  
#####
```

Exemplo

Plotando dois gráficos de potência

Exemplo

Objeto

Potencia

inicial: 1
quantidade: 7
saida ☐

Objeto


Grafico

preenchido: **true**
caractere: **#**

```
Grafico g1 = new Grafico(true, '#');  
Potencia p = new Potencia(1, 7);  
p.conecta(g1);  
p.apresenta();  
#  
##  
####  
#####  
#####
```

Potencia

inicial: 1
quantidade: 7
saida ☐



Grafico

preenchido: **true**
caractere: **#**

Grafico

preenchido: **false**
caractere: *****

```
Grafico g2 = new Grafico(false, '*');  
p.conecta(g2);  
p.apresenta();
```

*

*

*

*

*

*

A faint, light gray background image showing a complex network of interconnected nodes and lines, resembling a molecular structure or a data network. The nodes are represented by small circles of varying sizes, and the lines are thin, connecting the nodes in a web-like pattern.

Múltiplos ponteiros e Agregação

Vetor

■ Declaração

```
<tipo>[] <declaração1>, ..., <declaraçãon>;  
<tipo> <declaração1>[], ..., <declaraçãon>[];
```

□ <declaração>

- Sintaxe: <nome> = <inicialização>
- Chaves são usadas para inicializar cada dimensão
- Ex.: `int primos[] = {1, 2, 3, 5, 7};`

■ Quando a inicialização não é inline o vetor ou matriz precisa ser instanciado

```
<nome> = new <tipo>[<tamanho>]
```

□ Ex.:

```
int primos[];  
primos = new int[5];
```


Empréstimo

- Escreva um módulo para calcular a próxima parcela de um financiamento
- Dados disponíveis
 - **S** - valor da primeira parcela
 - **N** - número de parcelas
 - **J** - percentual de juros mensal
- Cada nova parcela é sempre calculada em relação à anterior:

$$P_{\text{atual}} = P_{\text{anterior}} \text{ mais Juros}$$

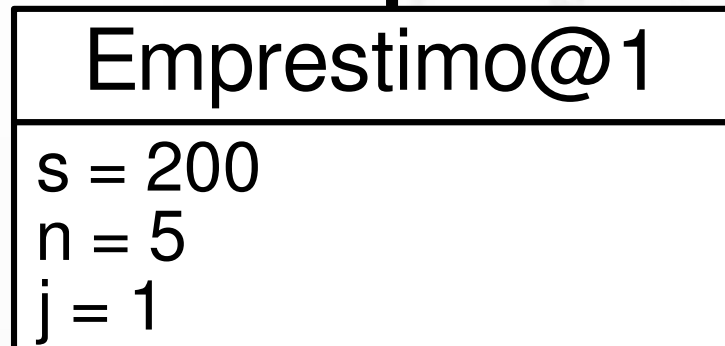


Empréstimo em UML

Emprestimo
s: float n: int j: float
proximaParcela(): float

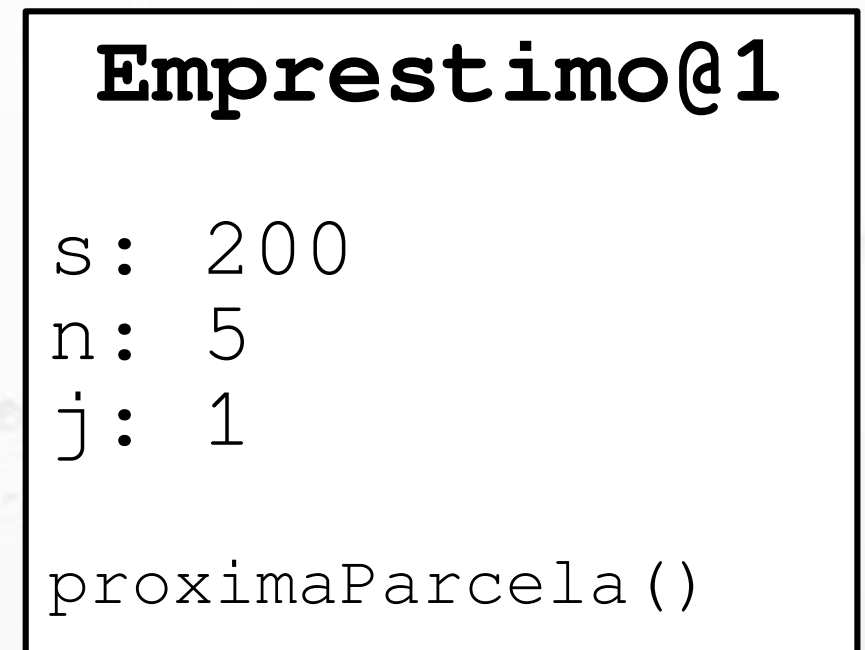
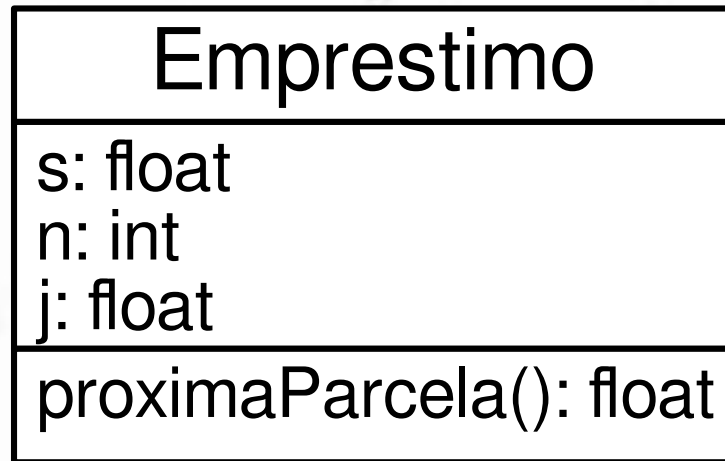
Emprestimo
s: float n: int j: float corrente: int p: float
«constructor» Emprestimo(s: float, n: int, j: float) proximaParcela(): float

Representação Simplificada



```
Emprestimo emprestimoA = new Emprestimo(200, 5, 1)
```

Representação Simplificada



Emprestimo `emprestimoA` = **new** Emprestimo(200, 5, 1)

Dois Empréstimos

Dois Empréstimos

emprestimoA



Emprestimo@1

s: 200

n: 5

j: 1

proximaParcela()

```
Emprestimo emprestimoA = new Emprestimo(200, 5, 1)
```

Dois Empréstimos

emprestimoA



Emprestimo@1

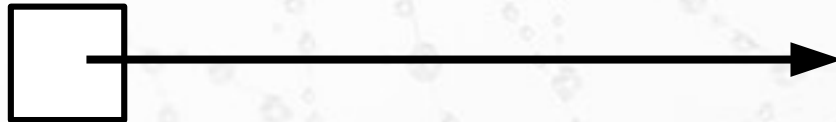
s: 200

n: 5

j: 1

proximaParcela()

emprestimoB



Emprestimo@2

s: 500

n: 7

j: 2

proximaParcela()

```
Emprestimo emprestimoA = new Emprestimo(200, 5, 1),  
emprestimoB = new Emprestimo(500, 7, 2);
```

Chamada de Método

emprestimoA



Emprestimo@1

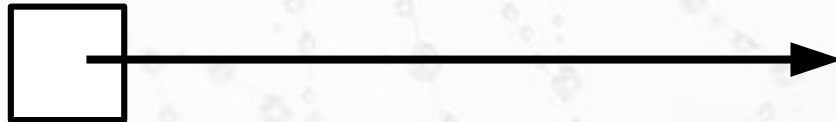
s: 200

n: 5

j: 1

proximaParcela()

emprestimoB



Emprestimo@2

s: 500

n: 7

j: 2

proximaParcela()

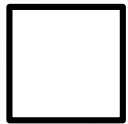
```
float pA = emprestimoA.proximaParcela();
```

```
float pB = emprestimoB.proximaParcela();
```


Múltiplos Empréstimos

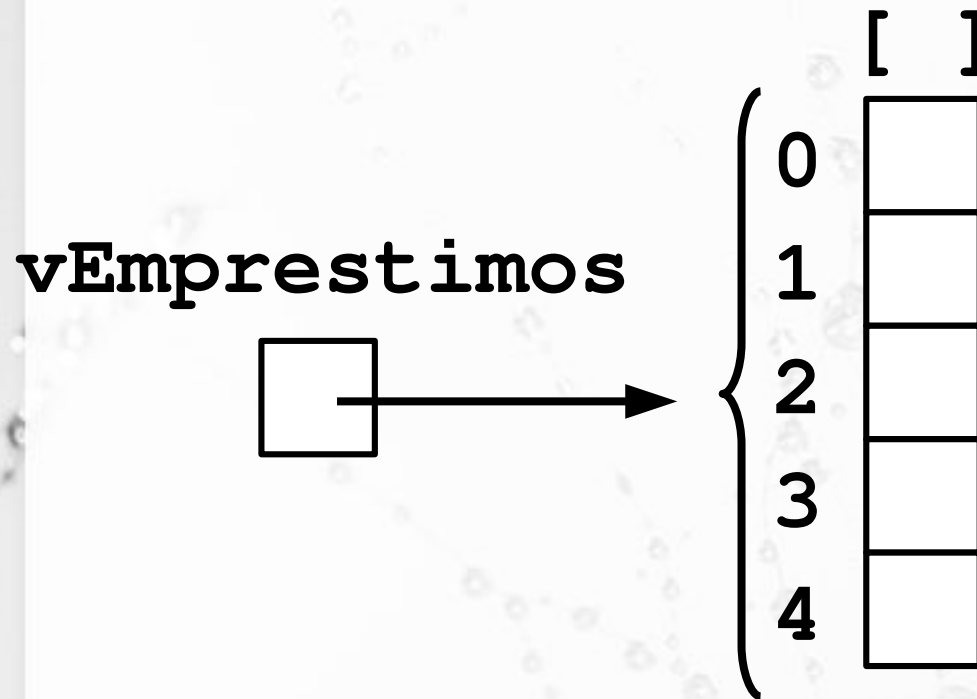
Vetor de Ponteiros

vEmprestimos



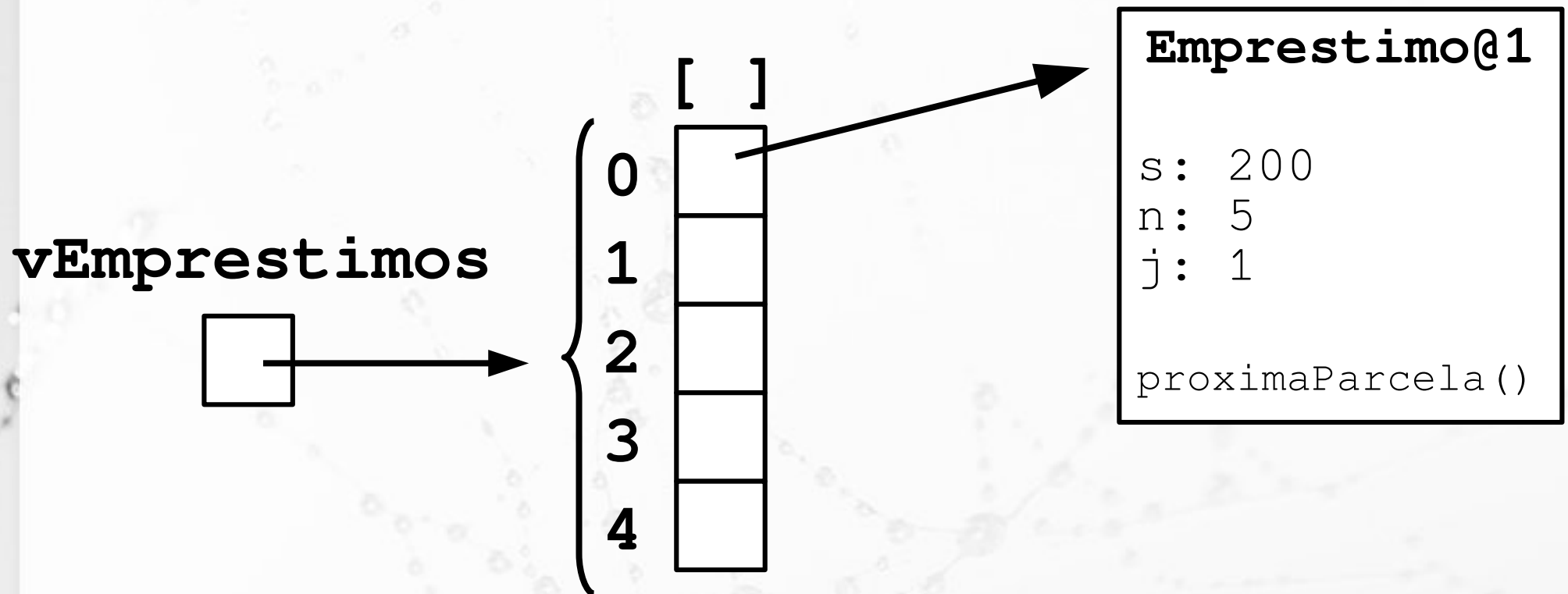
```
Emprestimo vEmprestimos[];
```

Vetor de Ponteiros



```
Emprestimo vEmprestimos[];  
vEmprestimos = new Emprestimo[5];
```

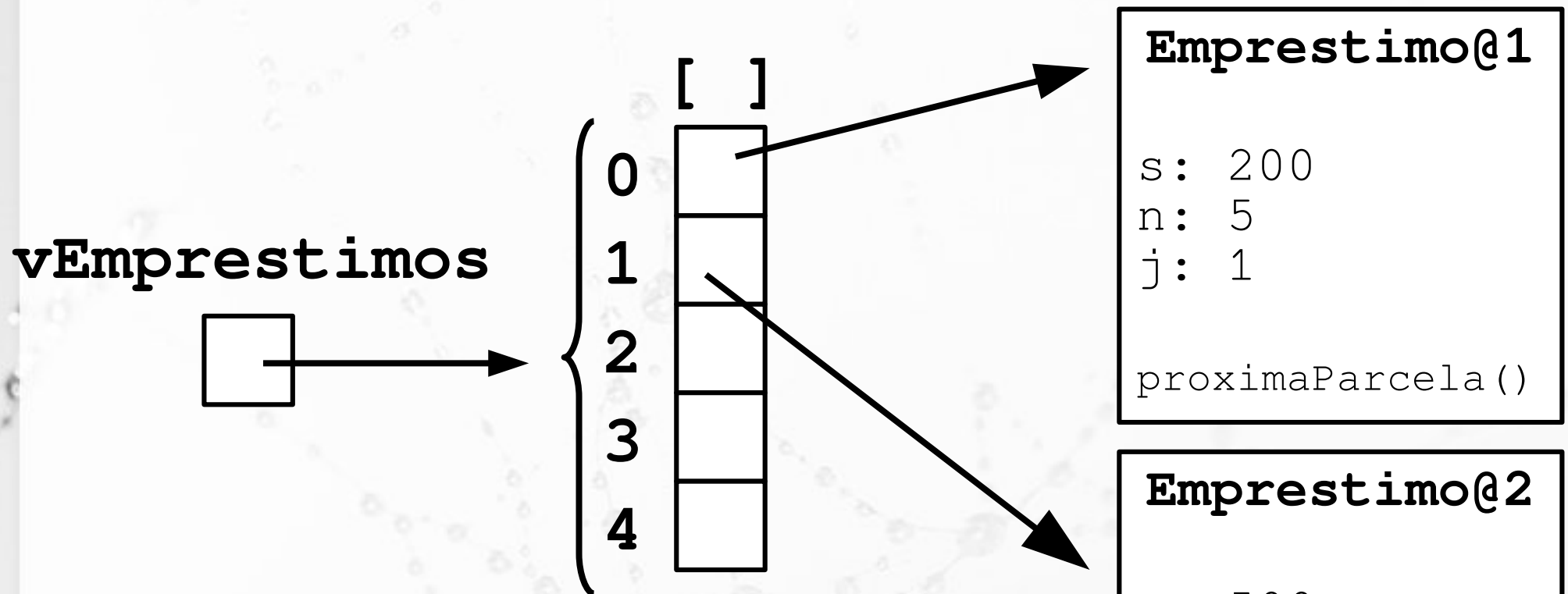
Vetor de Ponteiros



```
Emprestimo vEmprestimos[];  
vEmprestimos = new Emprestimo[5];
```

```
vEmprestimos[0] = new Emprestimo(200, 3, 1);
```

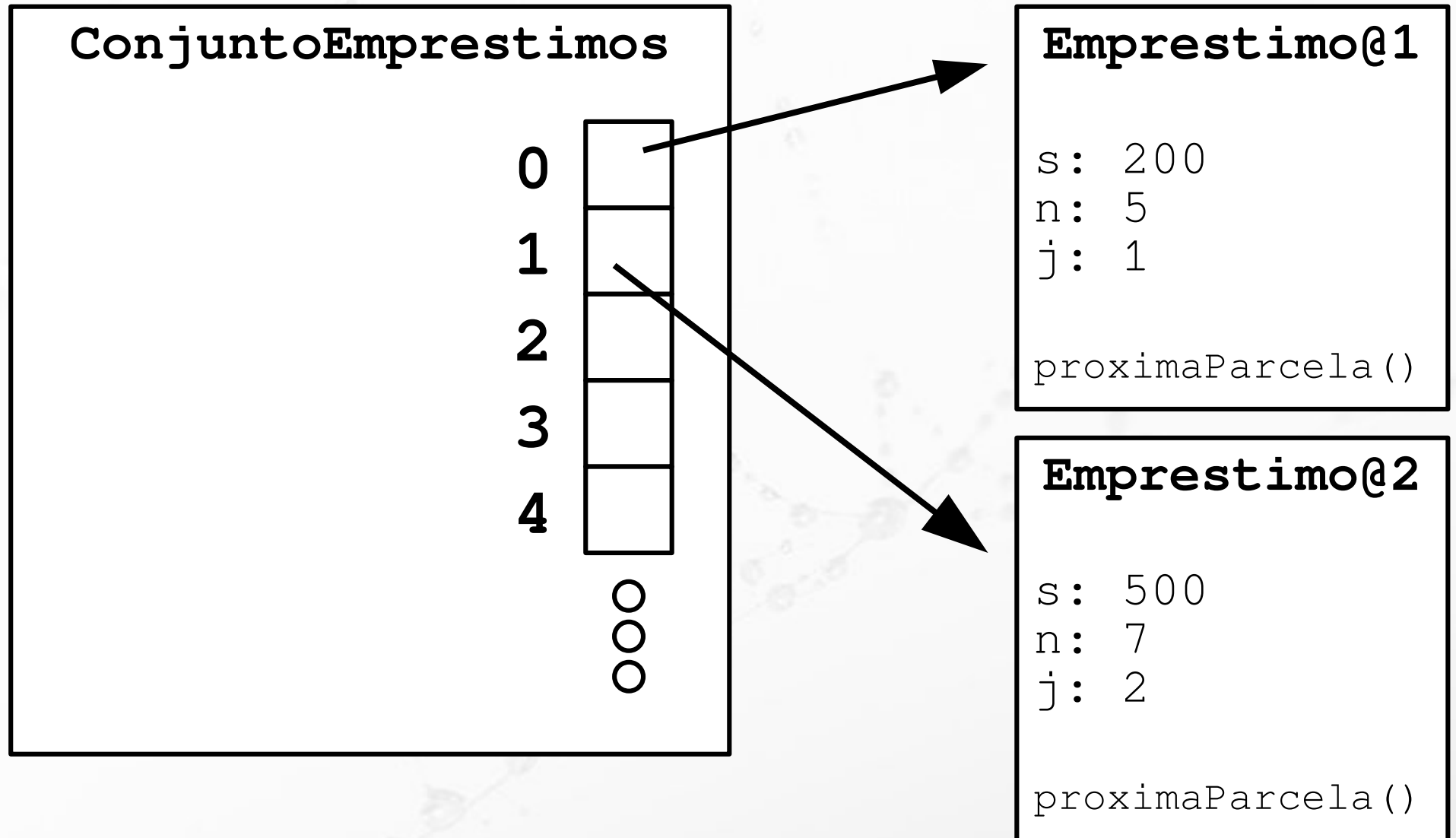
Vetor de Ponteiros



```
Emprestimo vEmprestimos[];  
vEmprestimos = new Emprestimo[5];
```

```
vEmprestimos[0] = new Emprestimo(200, 3, 1);  
vEmprestimos[1] = new Emprestimo(500, 4, 2);
```

Exercício



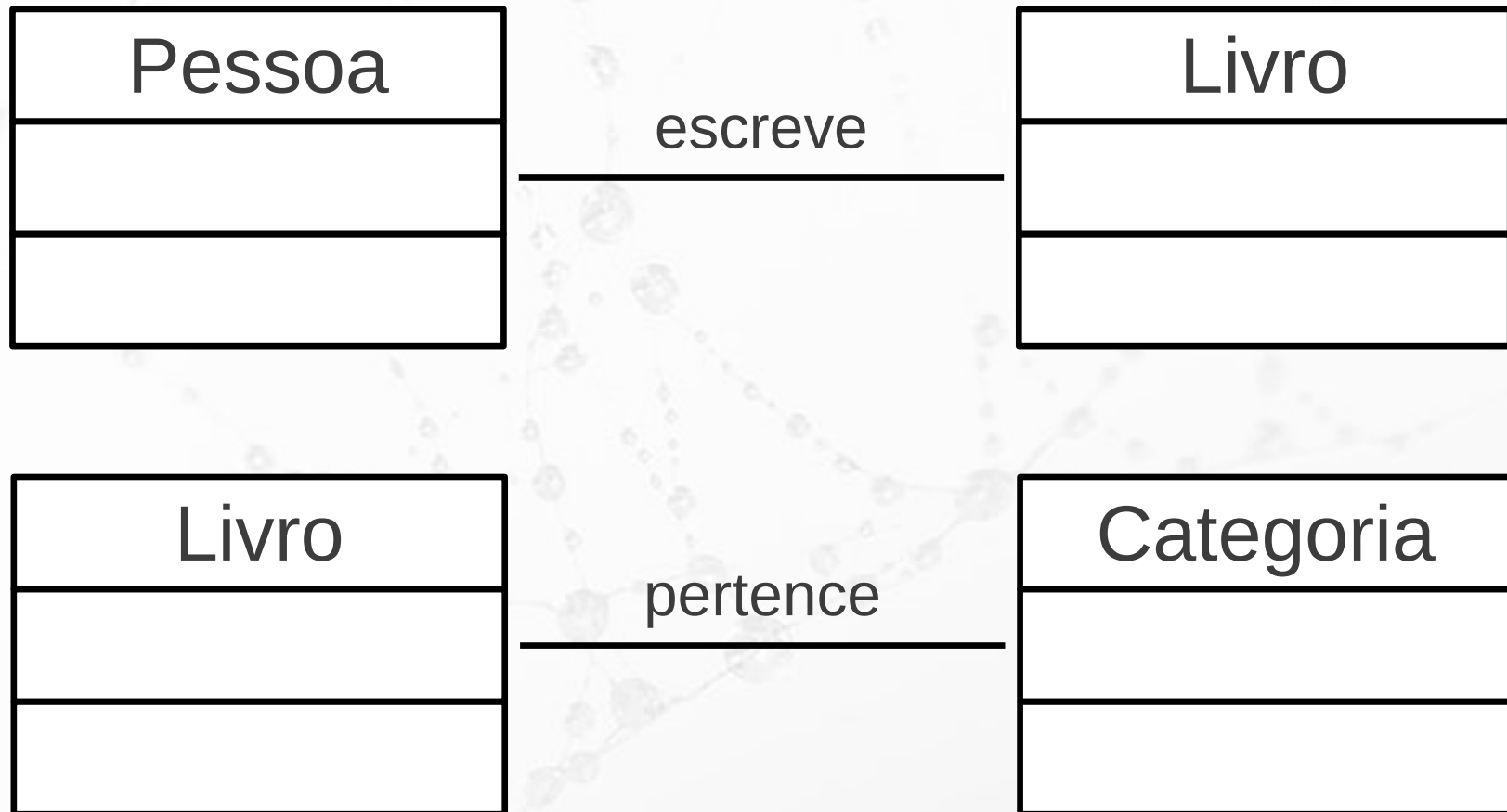
Relacionamento em UML

Relacionamento em UML

- Associação entre objetos
- Atributo de um objeto que se refere a outro
 - Atributo definido na classe



UML: Relacionamento



UML: Relacionamento Direcionado



Retomando o Gráfico de Potência

Objeto

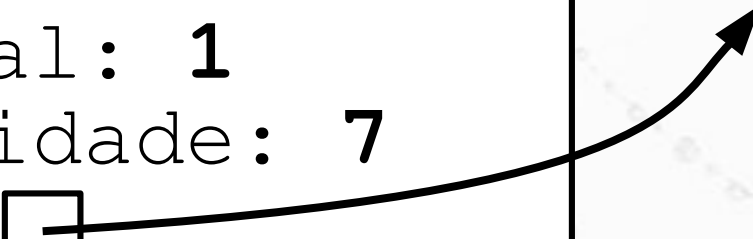
Potencia

inicial: **1**
quantidade: **7**
saida ☐

Objeto

Grafico

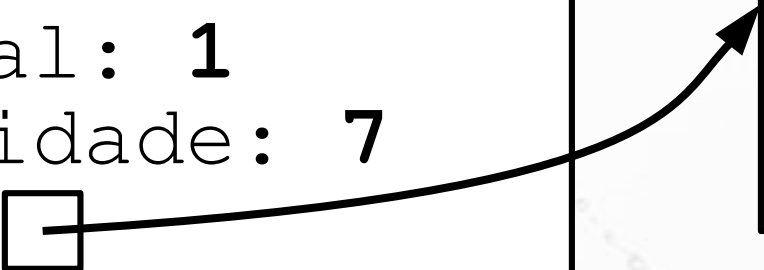
preenchido: **true**
caractere: **#**



Retomando o Gráfico de Potência

Potencia

inicial: **1**
quantidade: **7**
saida ☐



Grafico

preenchido: **true**
caractere: **#**

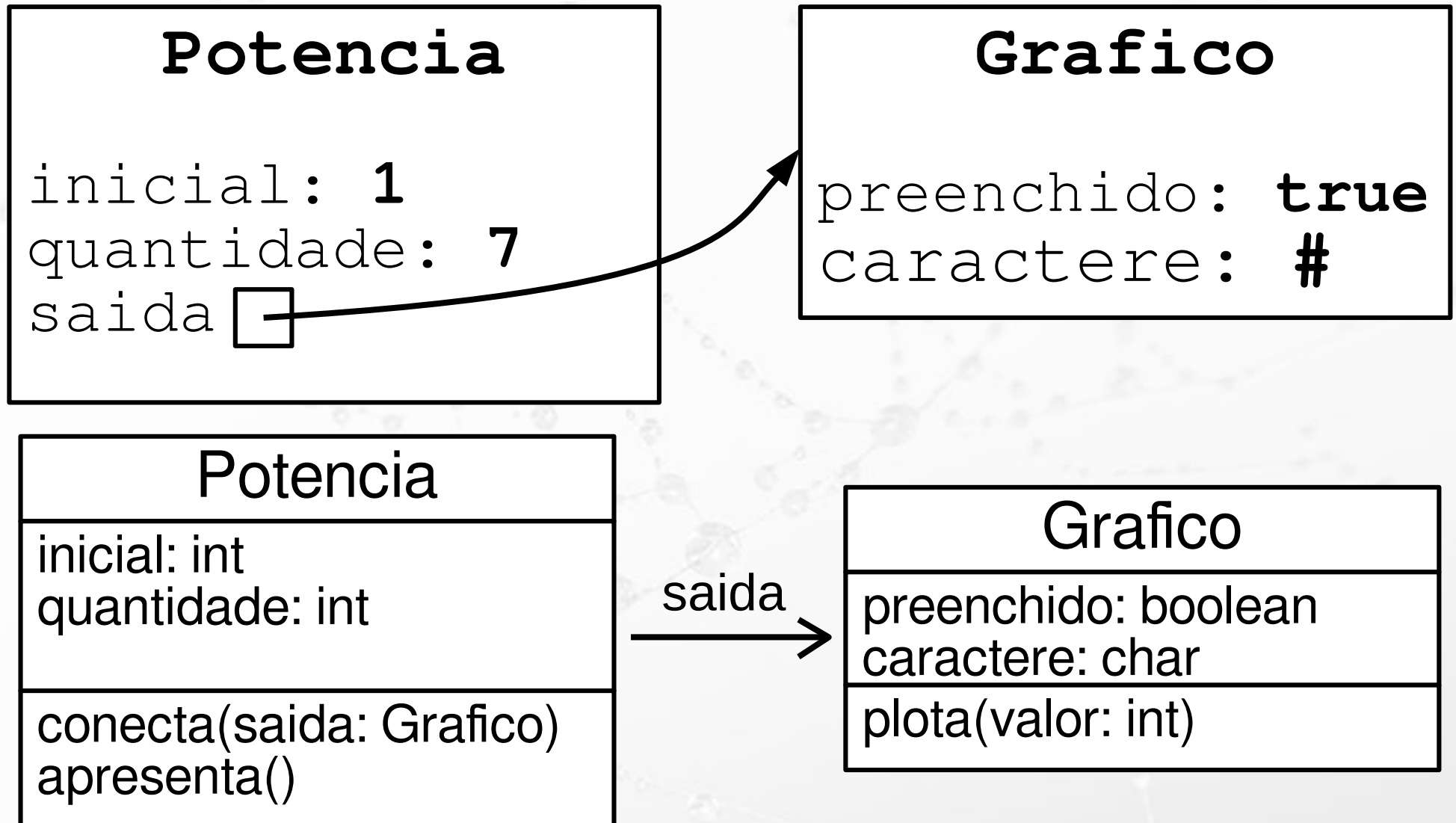
Potencia

inicial: int
quantidade: int
saida: Grafico
conecta(saida: Grafico)
apresenta()

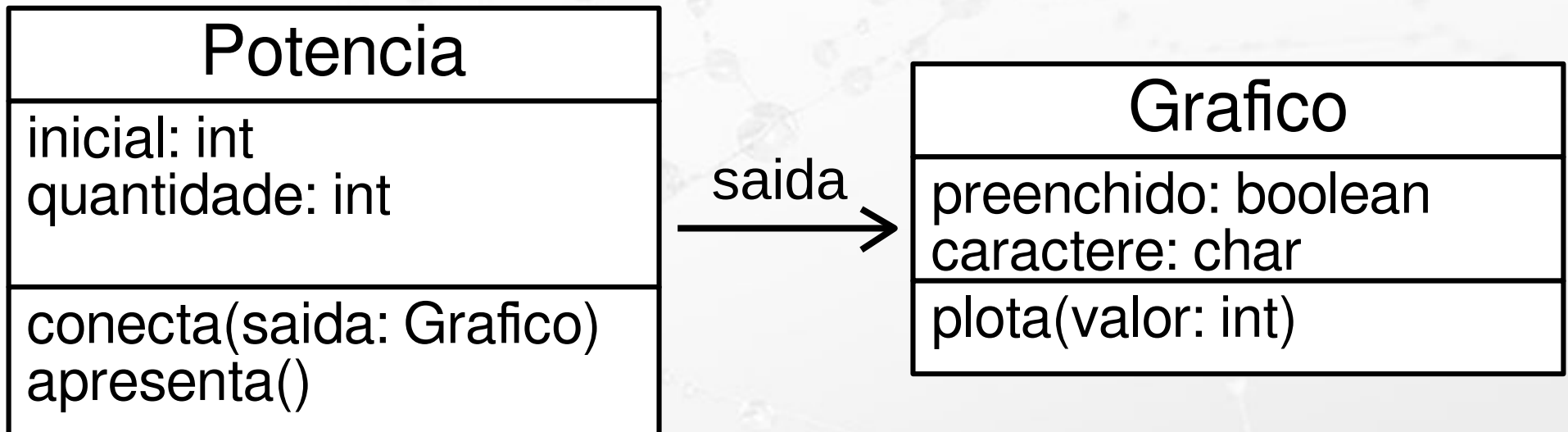
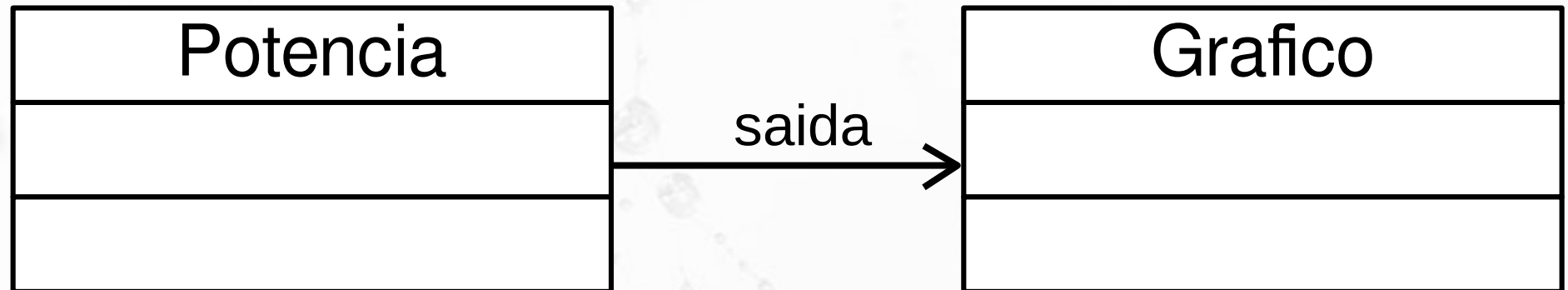
Grafico

preenchido: boolean
caractere: char
plota(valor: int)

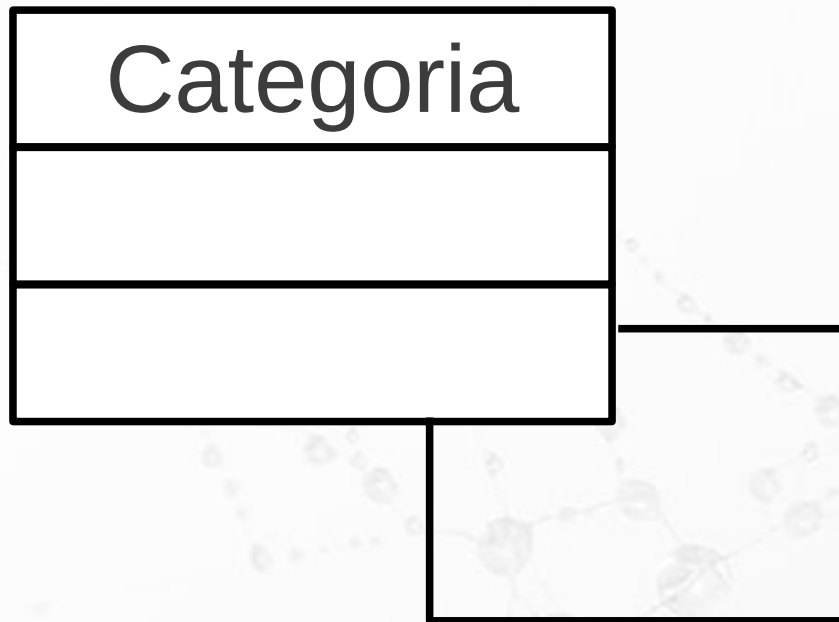
Retomando o Gráfico de Potência



Escondendo Atributos e Métodos

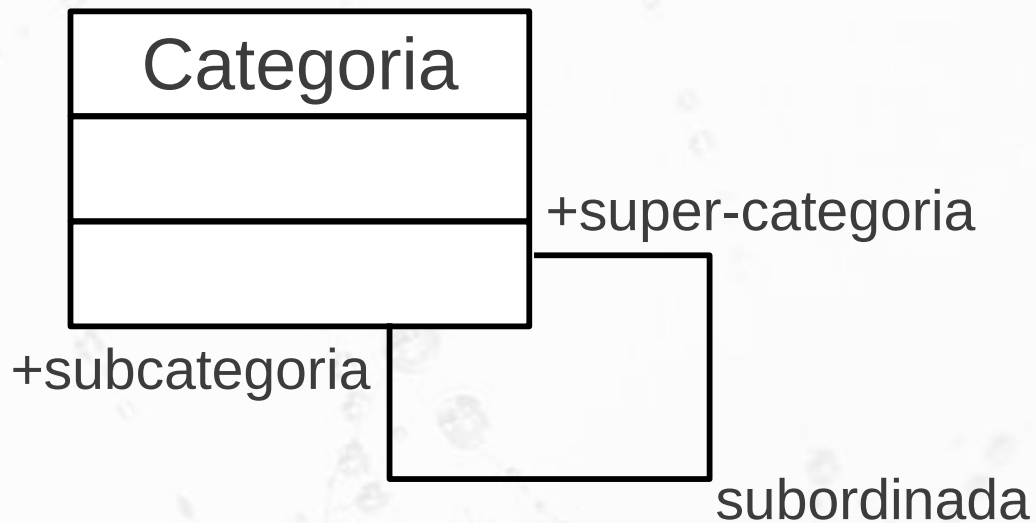


UML: Auto-relacionamento



subordinada

UML: Papéis



ER: Cardinalidade no Relacionamento

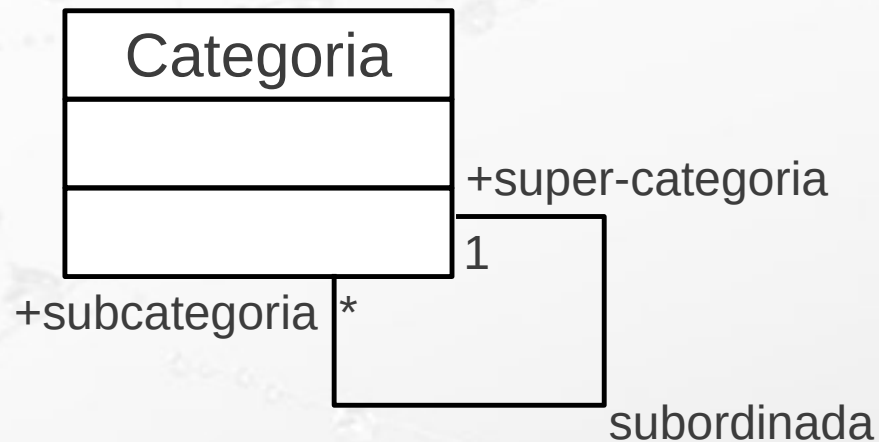
- Restrições que limitam a possibilidade de combinações de entidades em relacionamentos
- Cardinalidade:
 - Máxima
 - Mínima

Razão de Cardinalidade

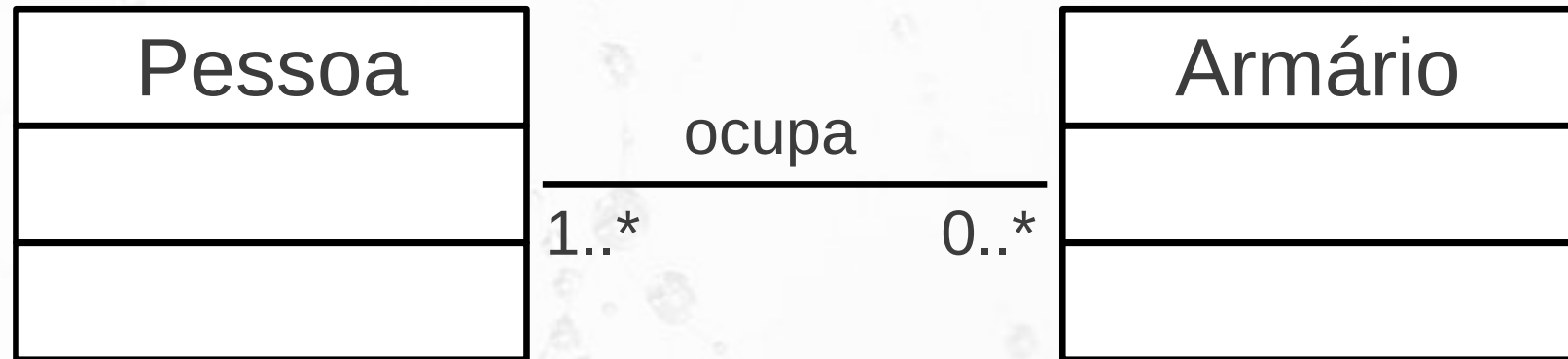
Razão de Cardinalidade

- É expressa a razão (ou proporção) de participação em um relacionamento.
- Transcrição gráfica das proporções: 1:1, 1:N, N:1 e N:N

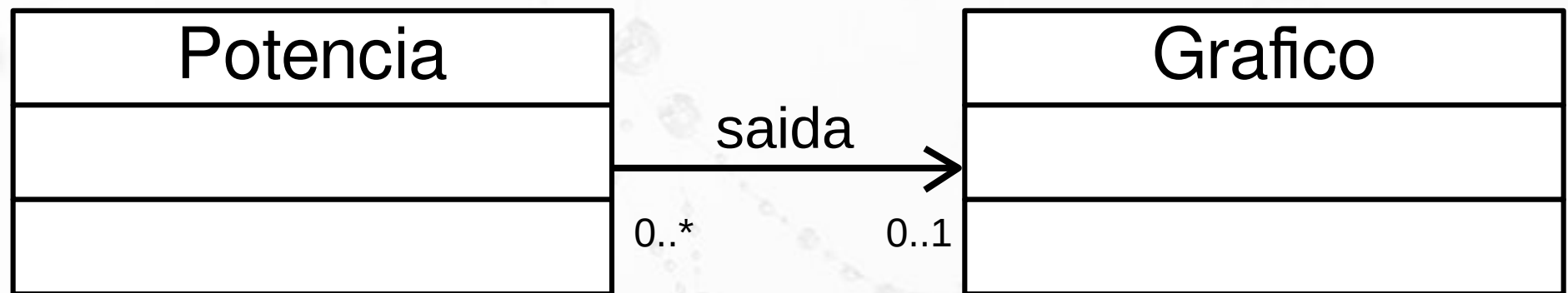
UML: Cardinalidade Máxima



UML: Cardinalidade Mínima

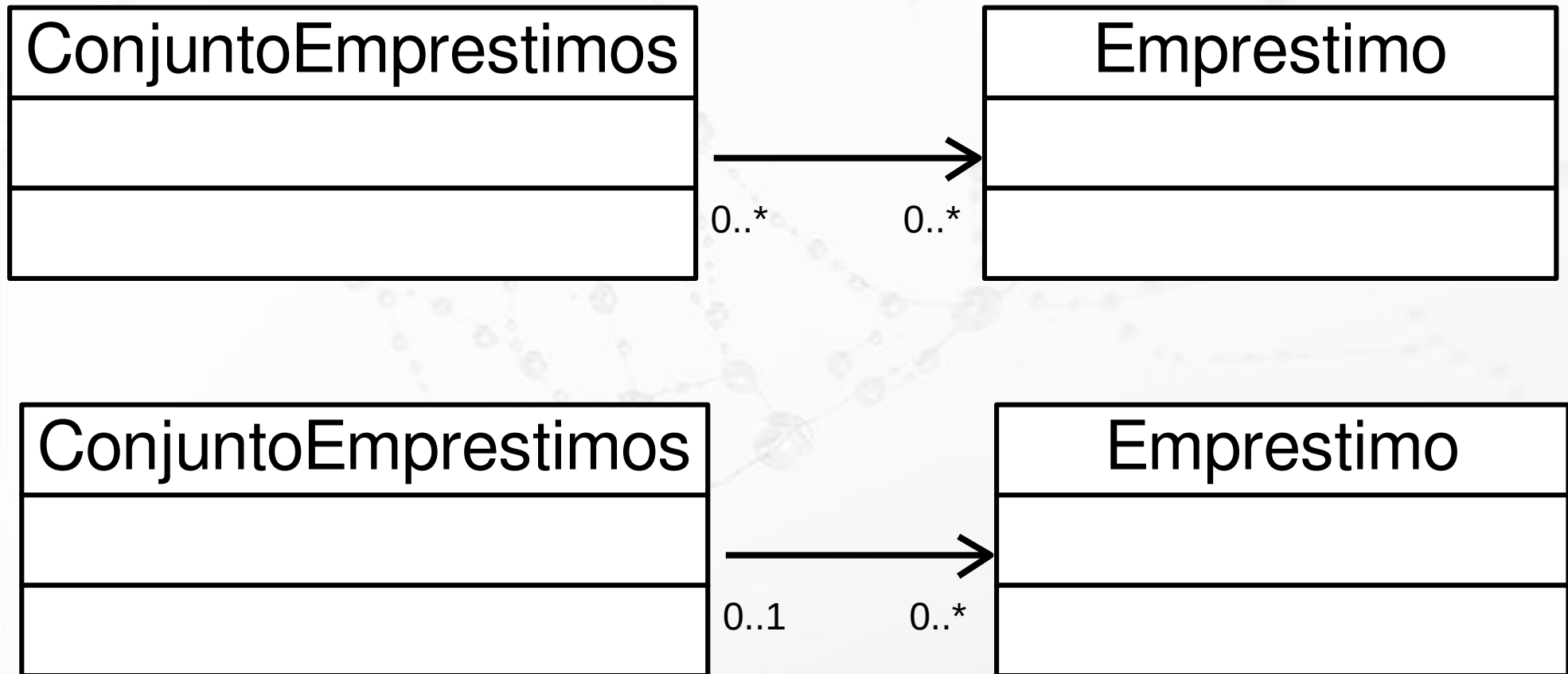


Cardinalidade Potência x Gráfico



Cardinalidade

ConjuntoEmprestimos x Empréstimo



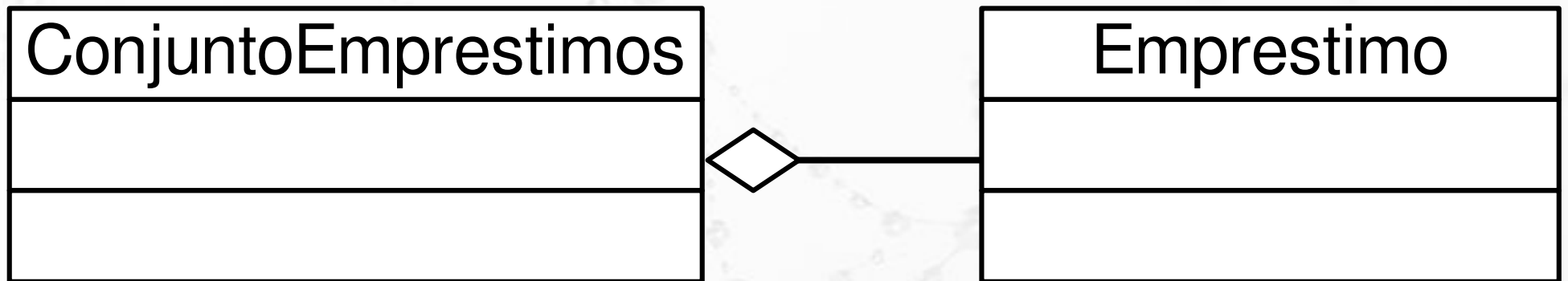
Agregação

UML: Agregação

- Uma classe agrega outra (não exclusivamente)

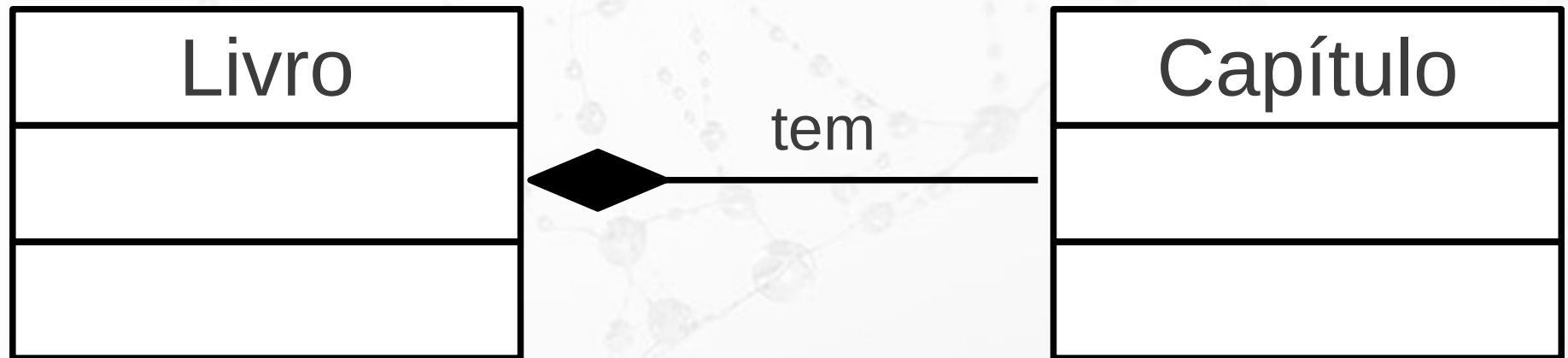


ConjuntoEmprestimos agrega objetos Empréstimo

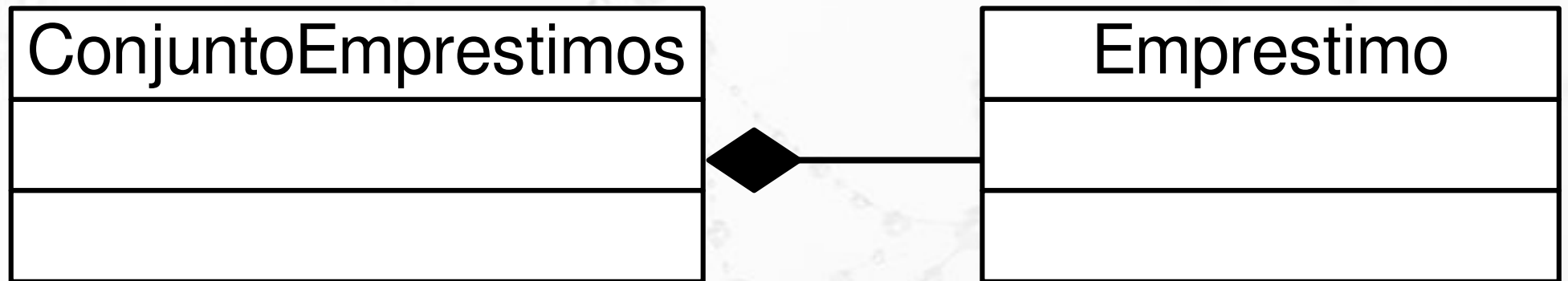


UML: Composição

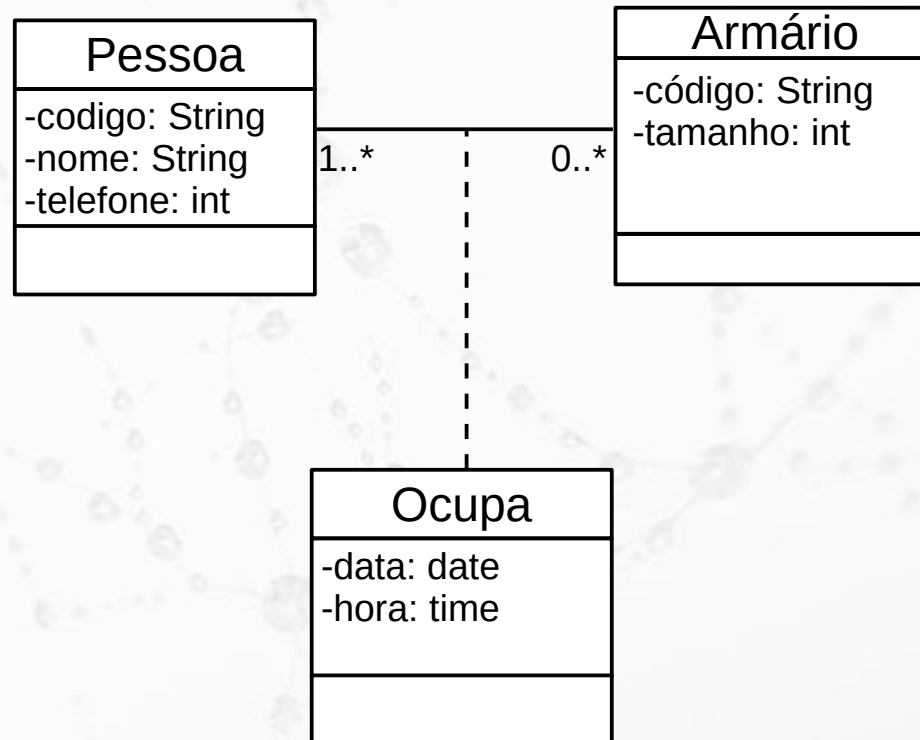
- Relação existencial (exclusiva) entre a parte e o todo



ConjuntoEmprestimos é
composto de objetos
Emprestimo



Classe de Associação



Exercício – Zombie Health



Exercício

■ Modele duas classes:

■ Classe Zumbi:

- atributos: nome

- Métodos:

 - getNome - retorna o nome do Zumbi

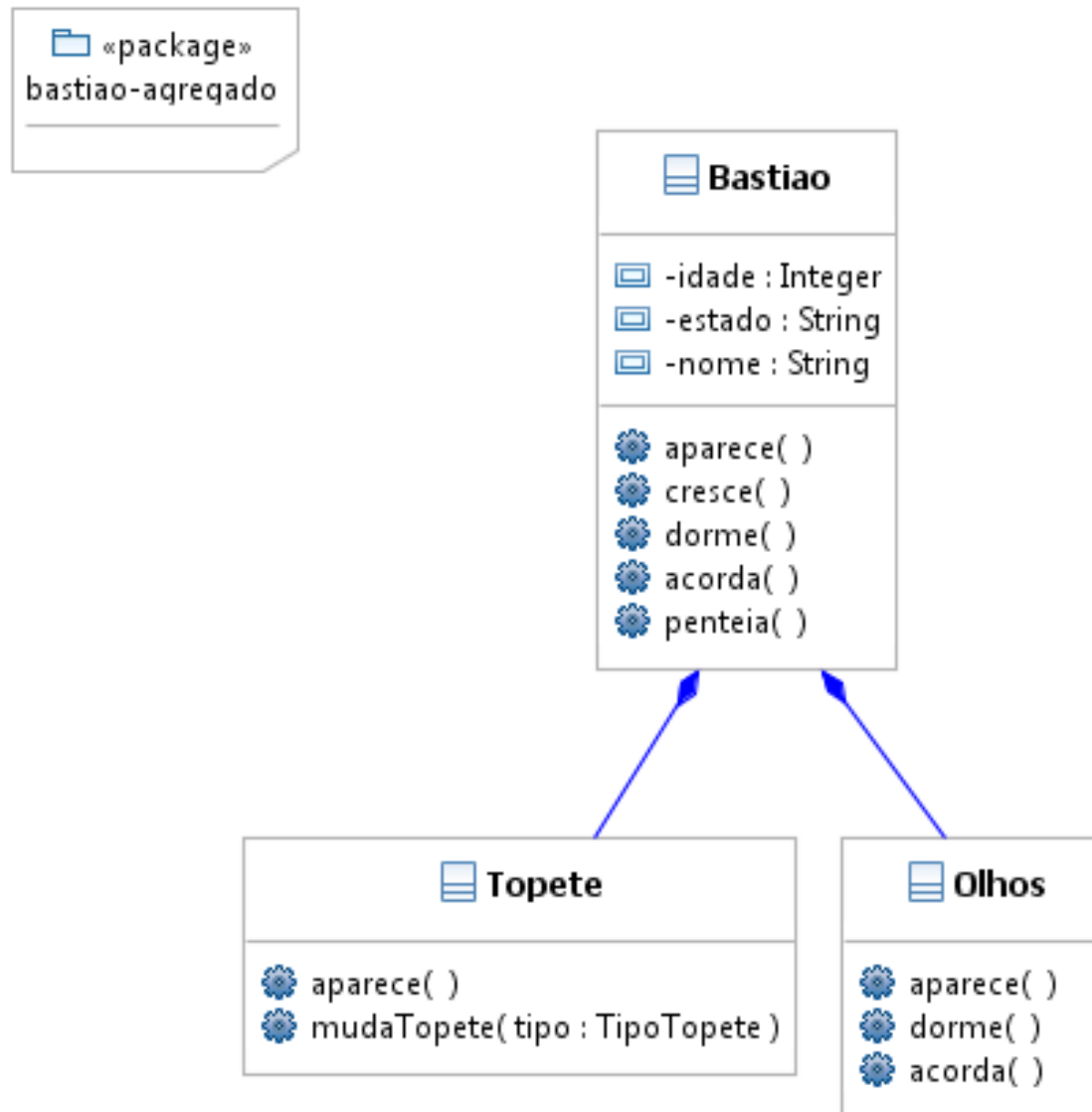
■ Classe Médico:

- Métodos:

 - indicaPaciente – recebe referência para o paciente e a guarda

 - consulta – pergunta o nome do paciente e o imprime na tela

Exemplo Bastião



Referências

- Rubira, Cecília Mary Fischer (2011). **Introdução à Programação Orientada a Objetos Usando Java**. Slides de aula, IC – Unicamp.



André Santanchè

<http://www.ic.unicamp.br/~santanche>

License

- These slides are shared under a Creative Commons License. Under the following conditions: Attribution, Noncommercial and Share Alike.
- See further details about this Creative Commons license at:
<http://creativecommons.org/licenses/by-nc-sa/3.0/>