Programação Orientada a Objetos

Polimorfismo

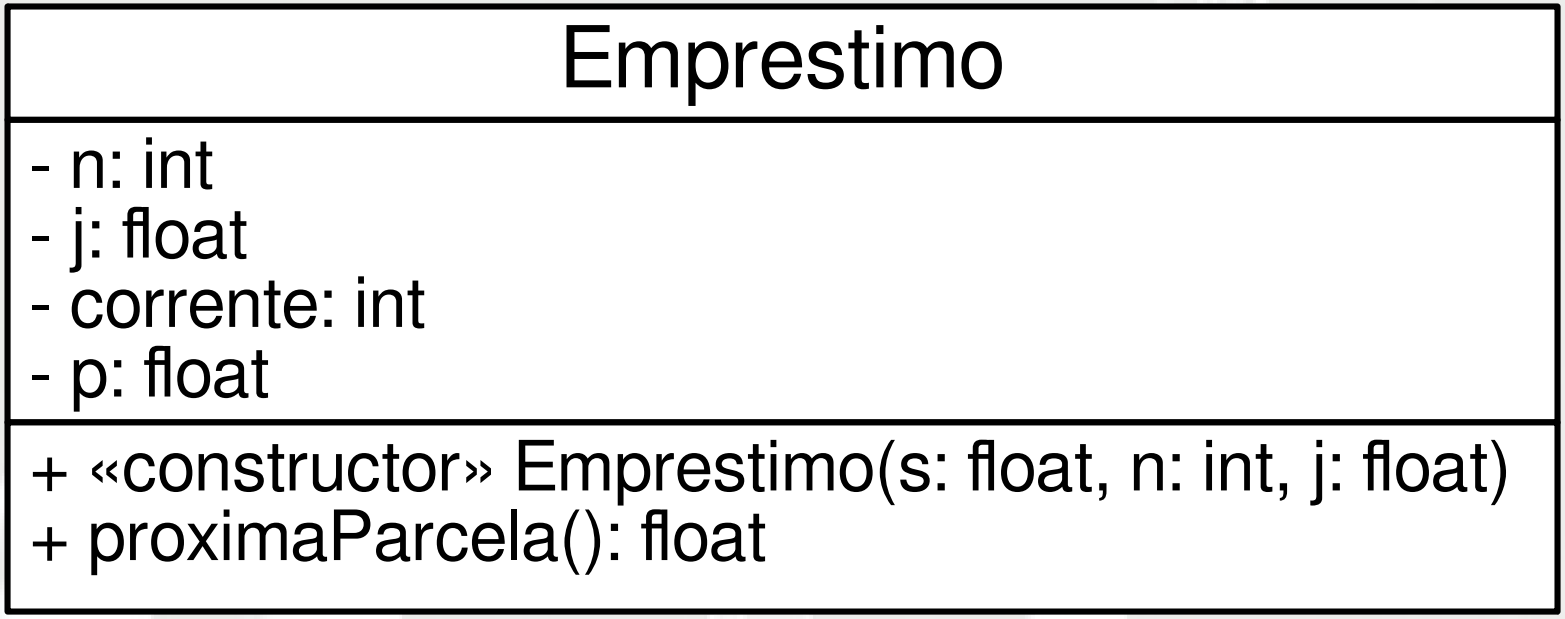
André Santanchè
Laboratory of Information Systems – LIS
Instituto de Computação – UNICAMP
Abril 2020

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

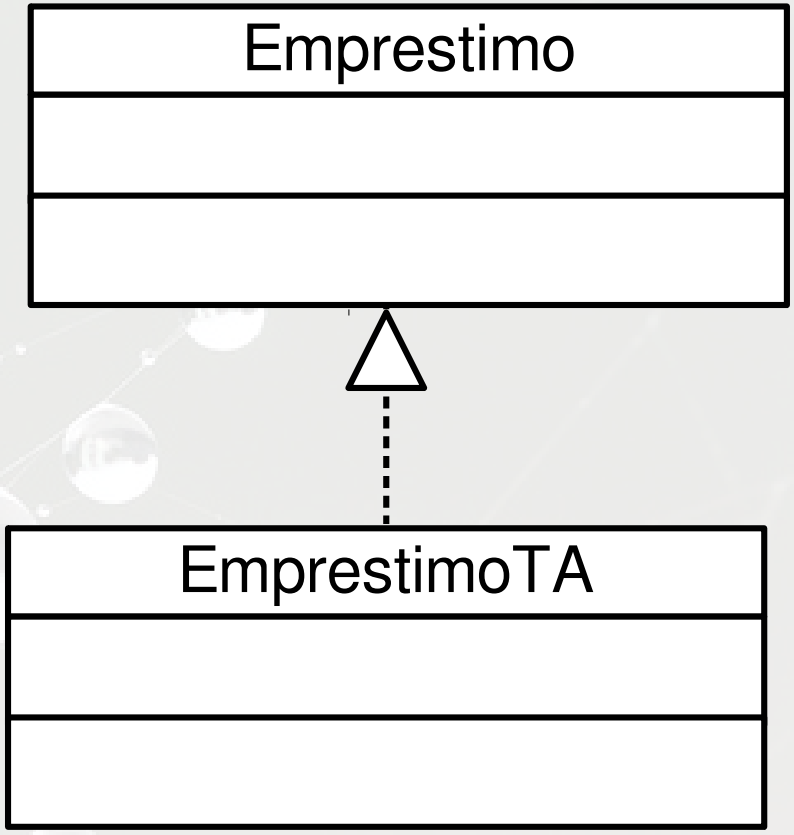
- Cada nova parcela é sempre calculada em relação à anterior:
  - $P_{atual} = P_{anterior}$ mais Juros

Classe Empréstimo



Empréstimo com Taxa de Administração

- Taxa de valor fixo adicionada sobre o valor do empréstimo.



Tipos de Sobrecarga de Métodos

- Sobrecarga na mesma classe
  - assinaturas diferentes (aula passada)
- Sobrecarga em classes herdeiras
  - assinaturas podem ser iguais ou diferentes

Sobrecarga na Classe `EmprestimoTA`

- Sobrecarga na mesma classe
  - assinaturas diferentes
- **Sobrecarga em classes herdeiras**
  - **assinaturas podem ser iguais ou diferentes**

Sobrecarga de `proximaParcela` em `EmprestimoTA`

- Sobrecarga na mesma classe
  - assinaturas diferentes
- **Sobrecarga em classes herdeiras**
  - **assinaturas** podem ser **iguais** ou diferentes

Empréstimo com Taxa de Administração

- Taxa de valor fixo adicionada sobre o valor do empréstimo

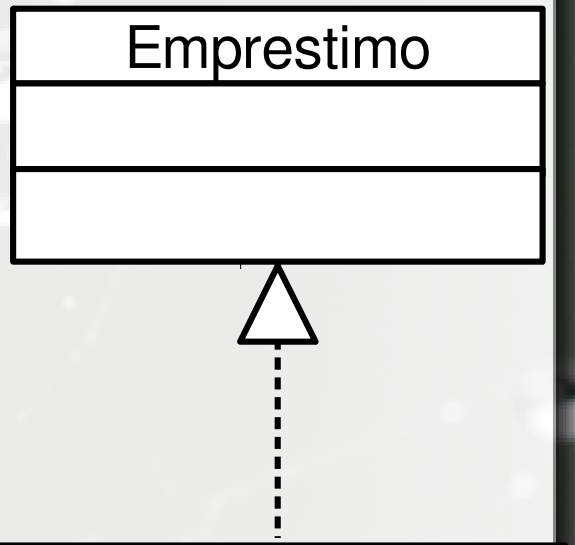
Empréstimo com Taxa de Administração

```
public class EmprestimoTA extends Emprestimo {
   private float ta;

   EmprestimoTA(float s, int n, float j, float ta) {
      super(s, n, j);
      this.ta = ta;
   }

   public float getTa() {
      return ta;
   }

   public float proximaParcela() {
      float pp = super.proximaParcela();
      if (pp > 0)
         pp += ta;
      return pp;
   }
}
```



Empréstimo com Taxa de Administração

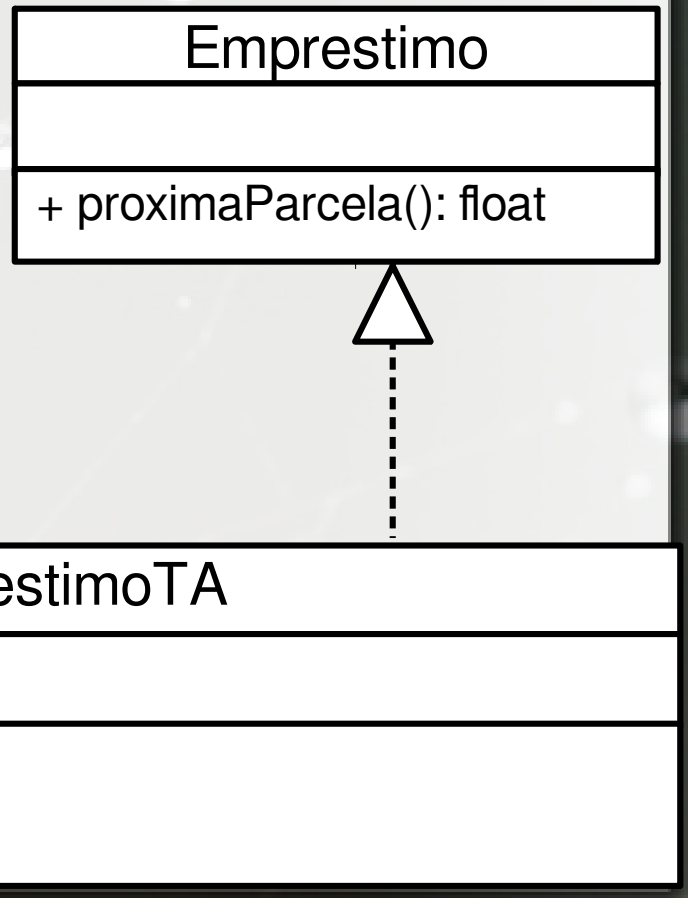
```
public class Emprestimo {

    ...

    public float proximaParcela() { … }
}


public class EmprestimoTA extends Emprestimo {
    ...

    public float proximaParcela() { … }
}
```

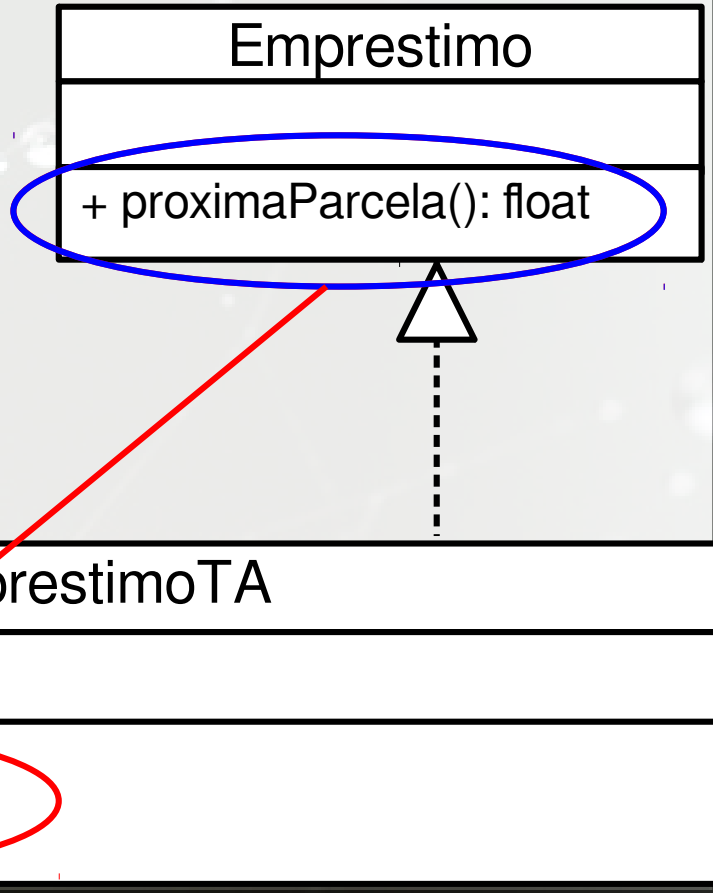


Sobrescrita do Método `proximaParcela`

```
public class Emprestimo {

   ...
   public float proximaParcela() { … }
}


public class EmprestimoTA extends Emprestimo {
   ...
   public float proximaParcela() { … }
}
```

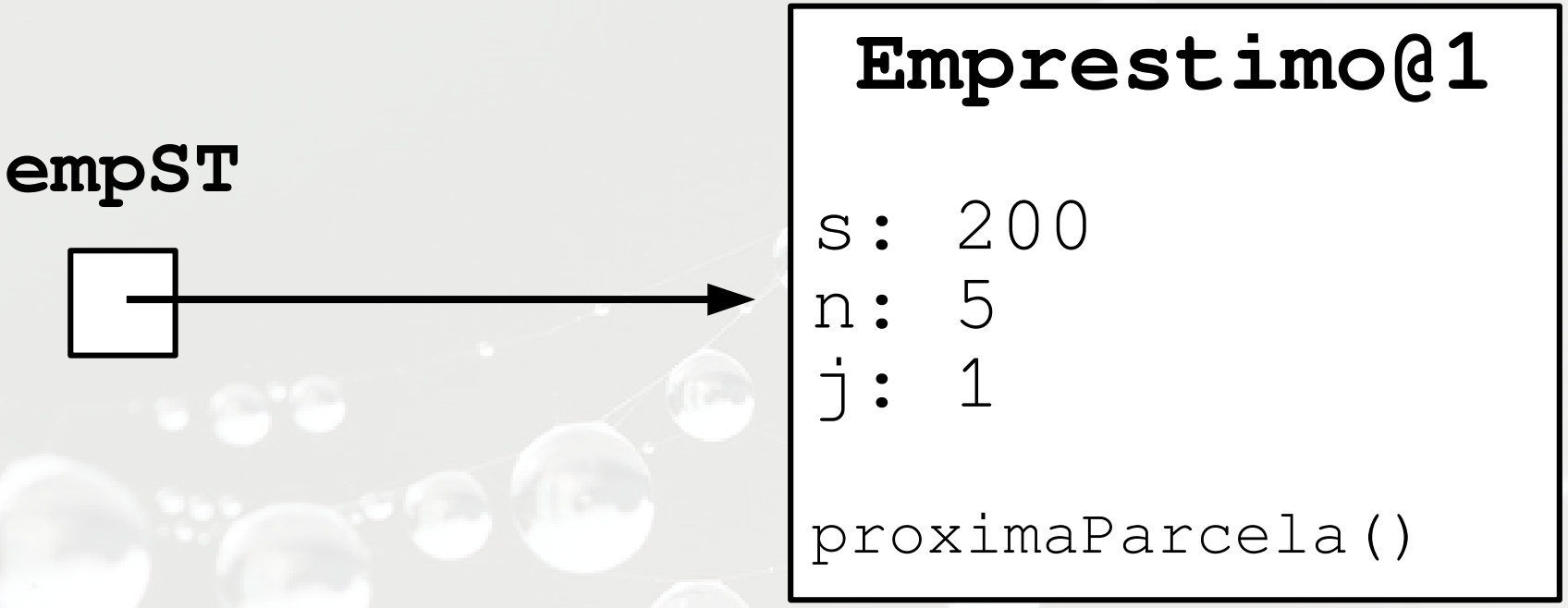


Sobrescrita:
Sobrecarga de método
com a mesma assinatura

Usando Métodos Sobrescritos

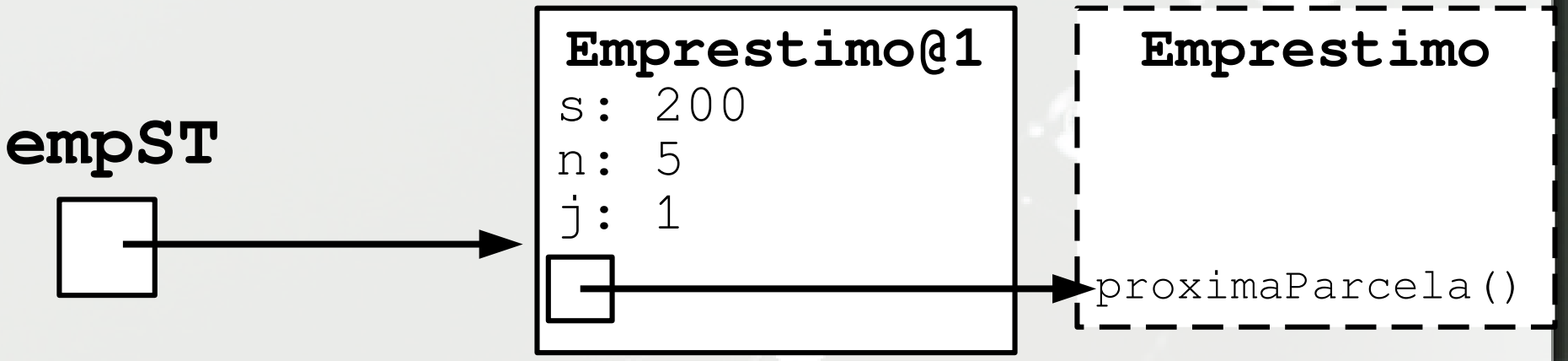
- Para um objeto da subclasse B, um método sobrescrito em B tem prioridade sobre o mesmo método da superclasse

Objeto Empréstimo Sem Tarifa



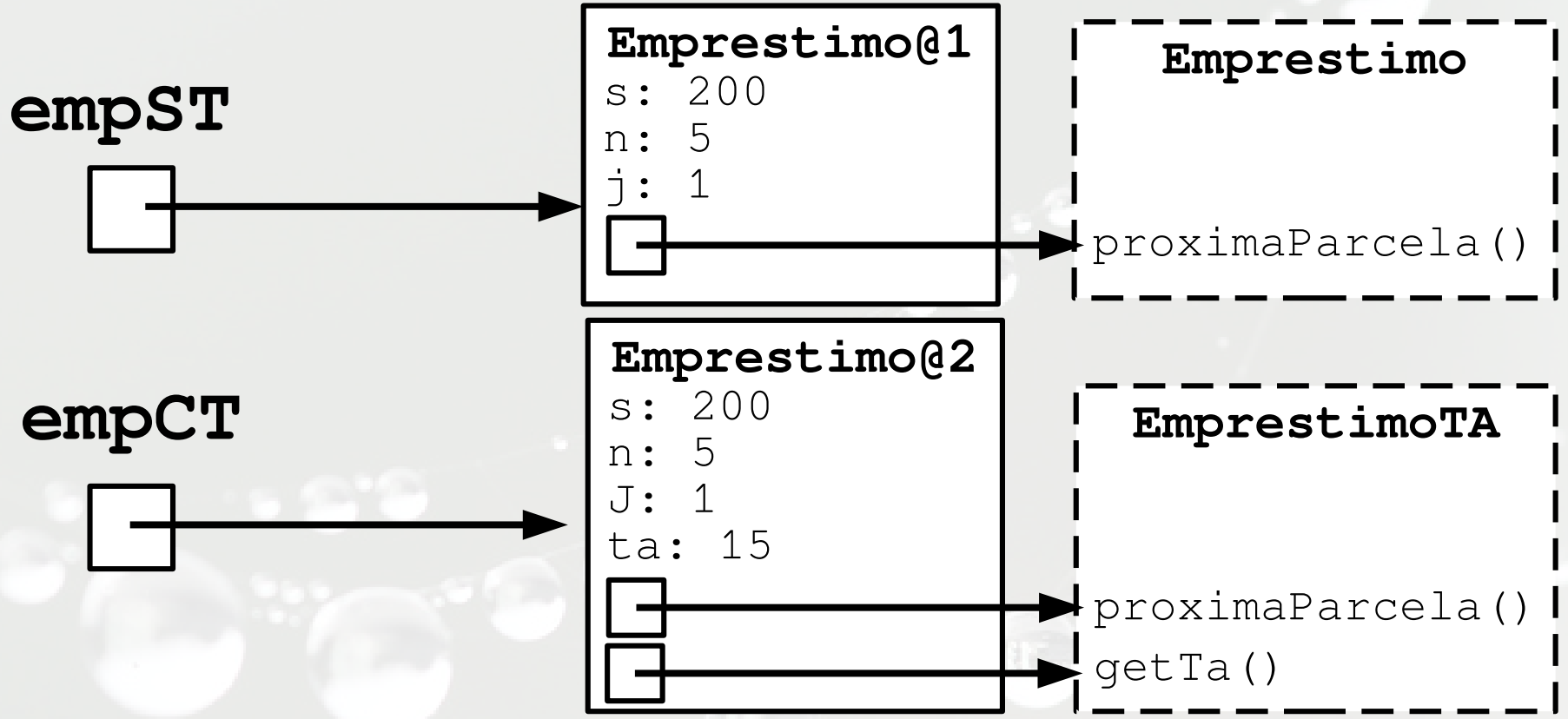

```
Emprestimo empST = new Emprestimo(200, 5, 1)
```

Método na Classe



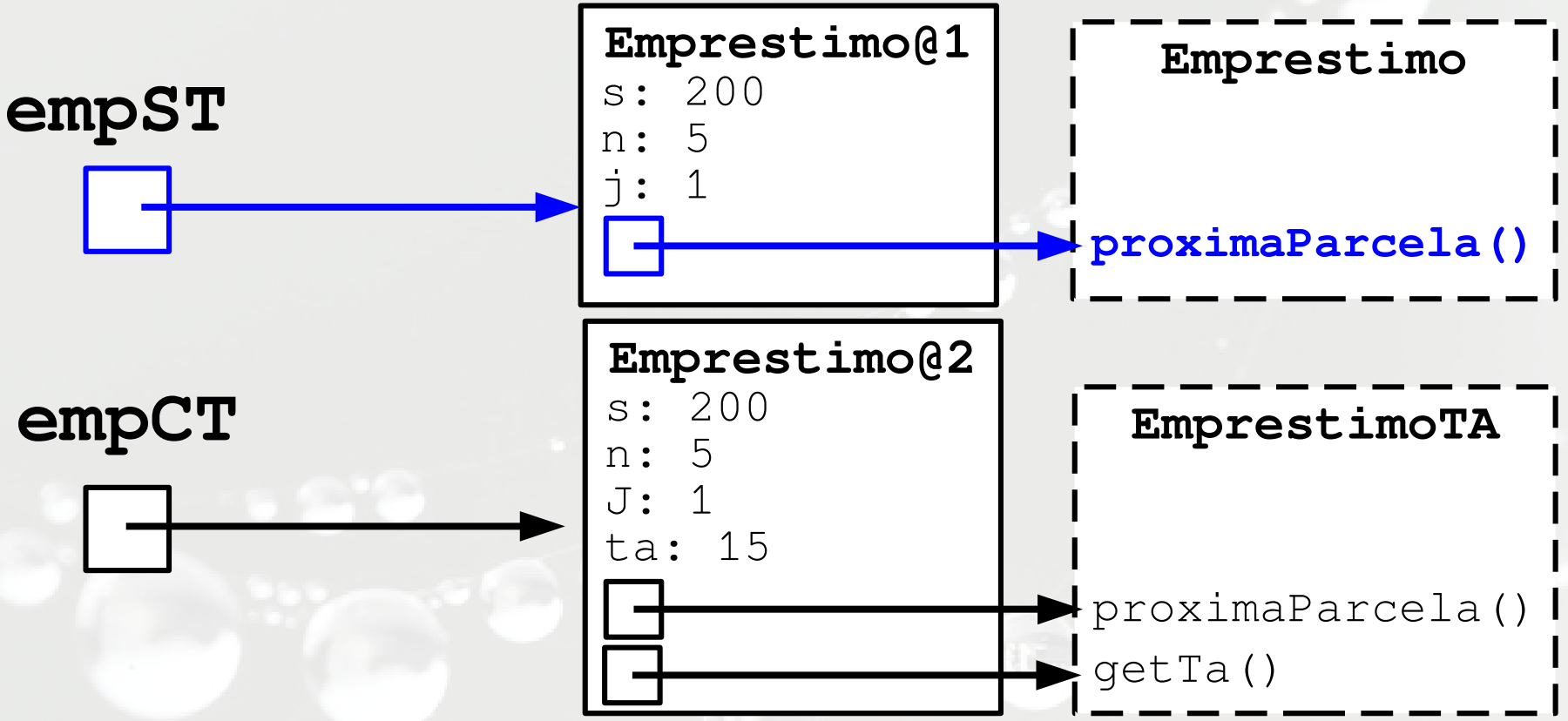

```
Emprestimo empST = new Emprestimo(200, 5, 1)
```

Método na Classe



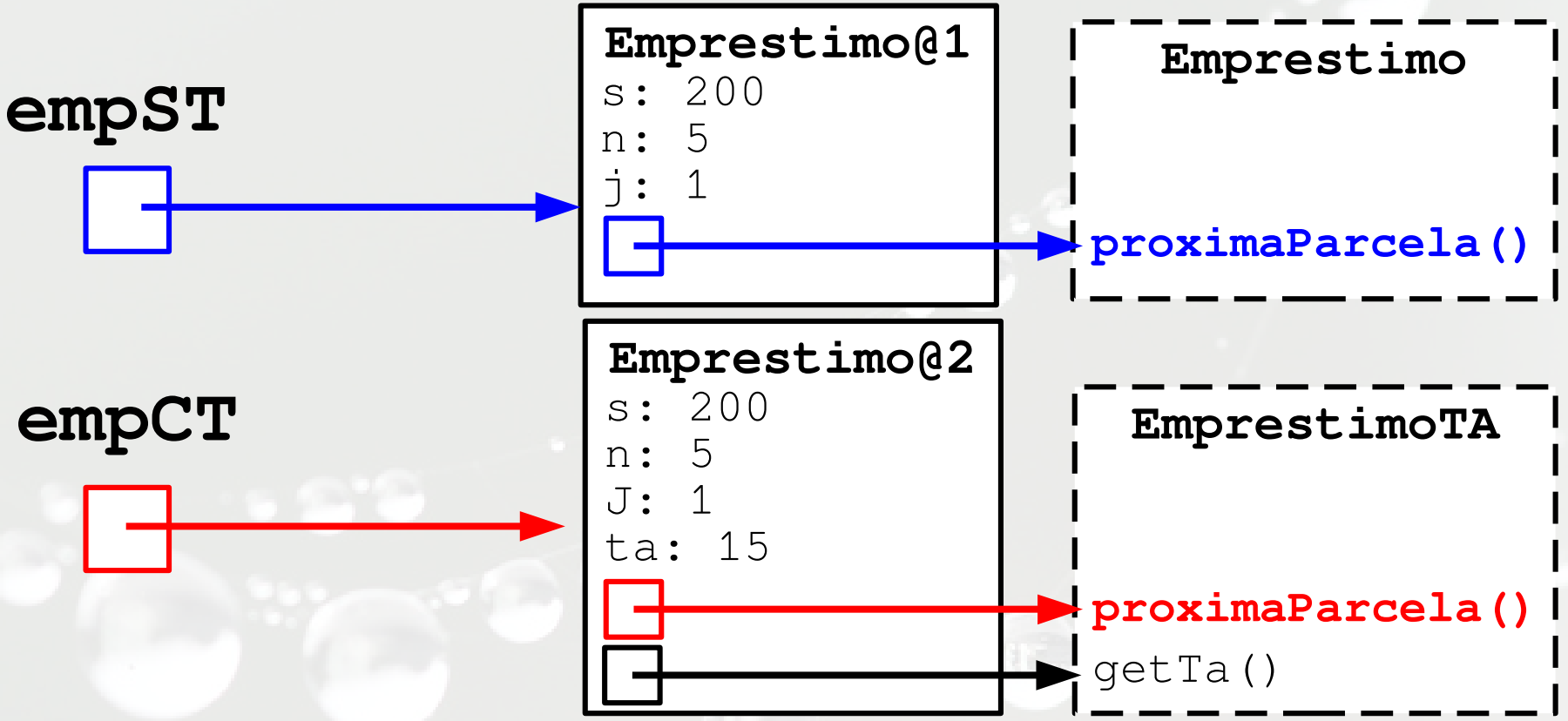

```
Emprestimo empST = new Emprestimo(200, 5, 1);
EmprestimoTA empCT = new EmprestimoTA(200, 5, 1, 15);
```

Método na Classe



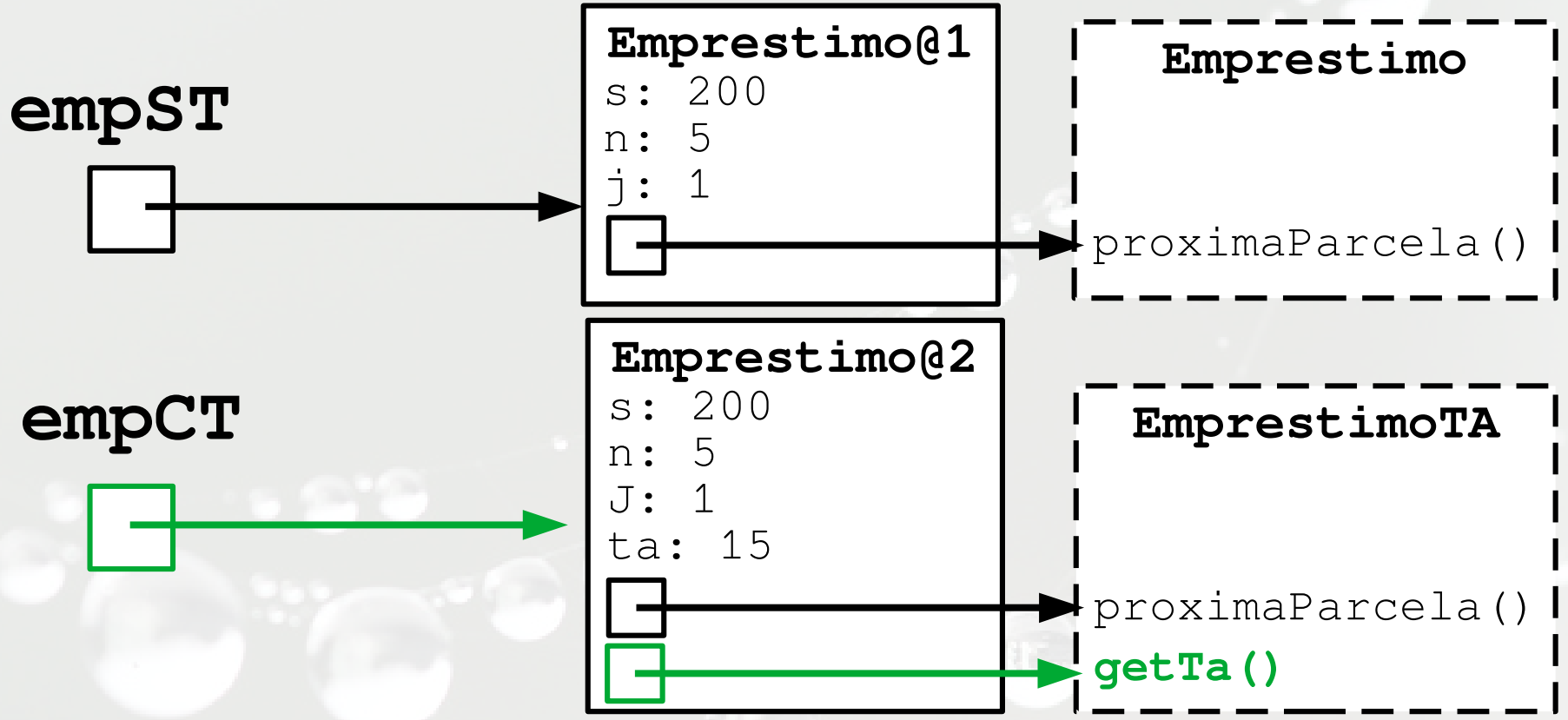

```
float pst = empST.proximaParcela();
```

Método na Classe

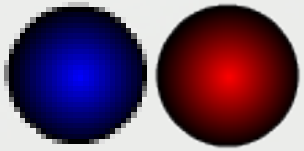



```
float pst = empST.proximaParcela();
float pct = empCT.proximaParcela();
```

Método na Classe




```
float adm = empCT.getTa();
```



Polimorfismo

- Propriedade de se apresentar ou tomar formas diversas
- "Habilidade das mais importantes dos sistemas orientados a objetos, e que consiste em as operações automaticamente se adequarem aos objetos aos quais estão sendo aplicadas." (Meyer, 1997)

Princípio para o Polimorfismo

- Uma variável declarada em uma classe pode ser instanciada em qualquer subclasse

Princípio para o Polimorfismo

- Uma variável declarada em uma classe pode ser instanciada em qualquer subclasse
- A declaração:

```
EmprestimoTA empCT = new EmprestimoTA(200, 5, 1, 15);
```

Princípio para o Polimorfismo

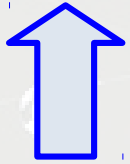
- Uma variável declarada em uma classe pode ser instanciada em qualquer classe herdeira
- A declaração:

```
EmprestimoTA empCT = new EmprestimoTA(200, 5, 1, 15);
```

- Também pode ser declarada como:

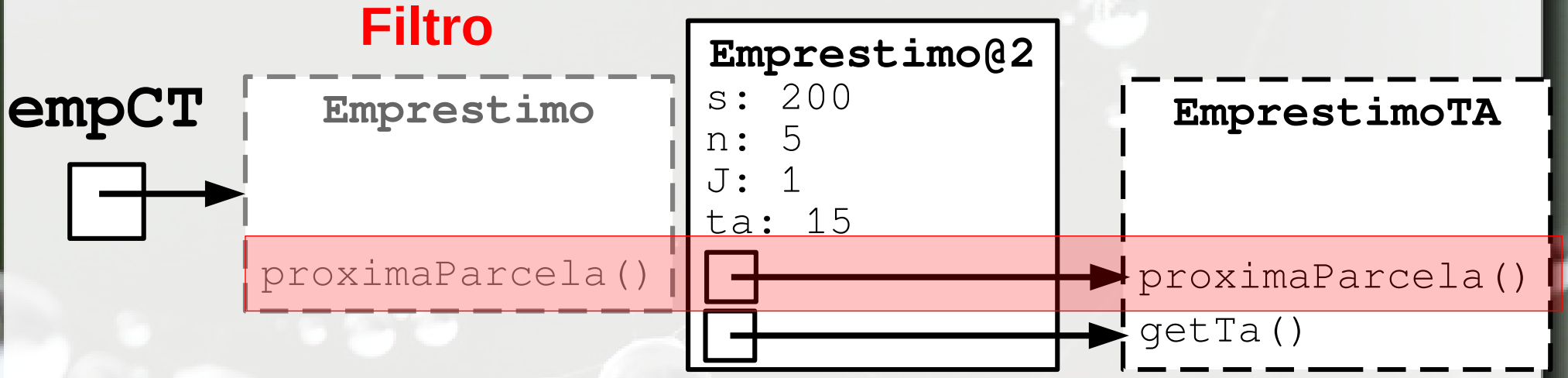
```
Emprestimo empCT = new EmprestimoTA(200, 5, 1, 15);
```

Declaração na Superclasse



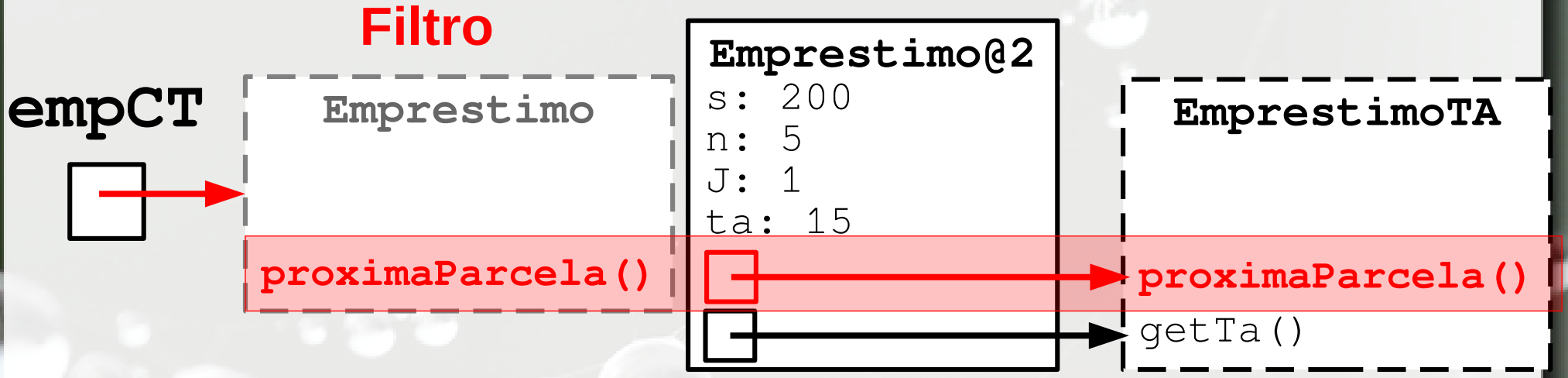
Instanciação na Herdeira

Método na Classe



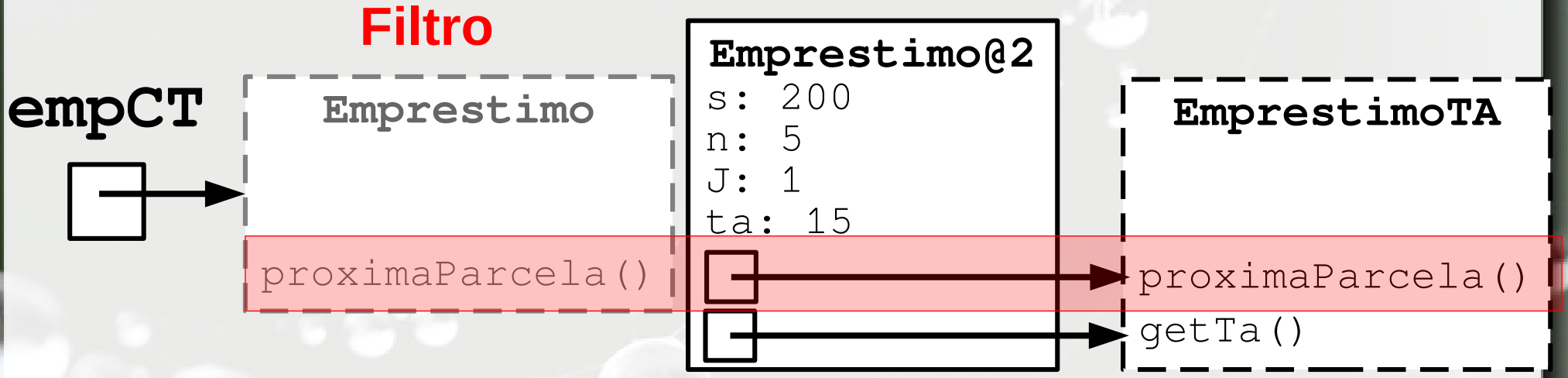

```
Emprestimo empCT = new EmprestimoTA(200, 5, 1, 15);
```

Método na Classe



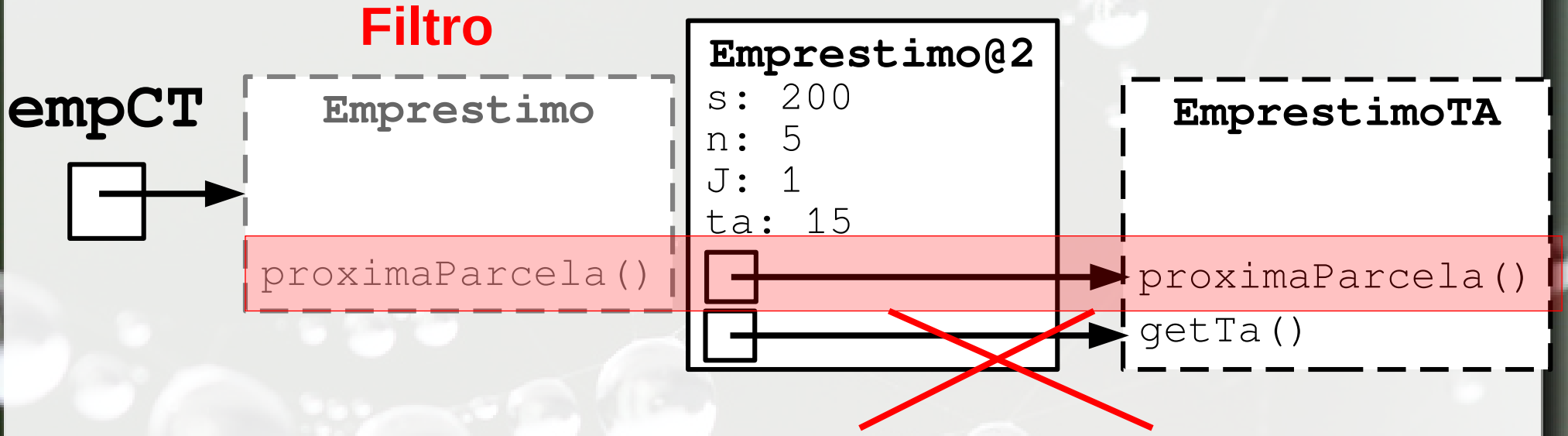

```
float pct = empCT.proximaParcela();
```

Método na Classe




```
float adm = empCT.getTa();
```

Método na Classe



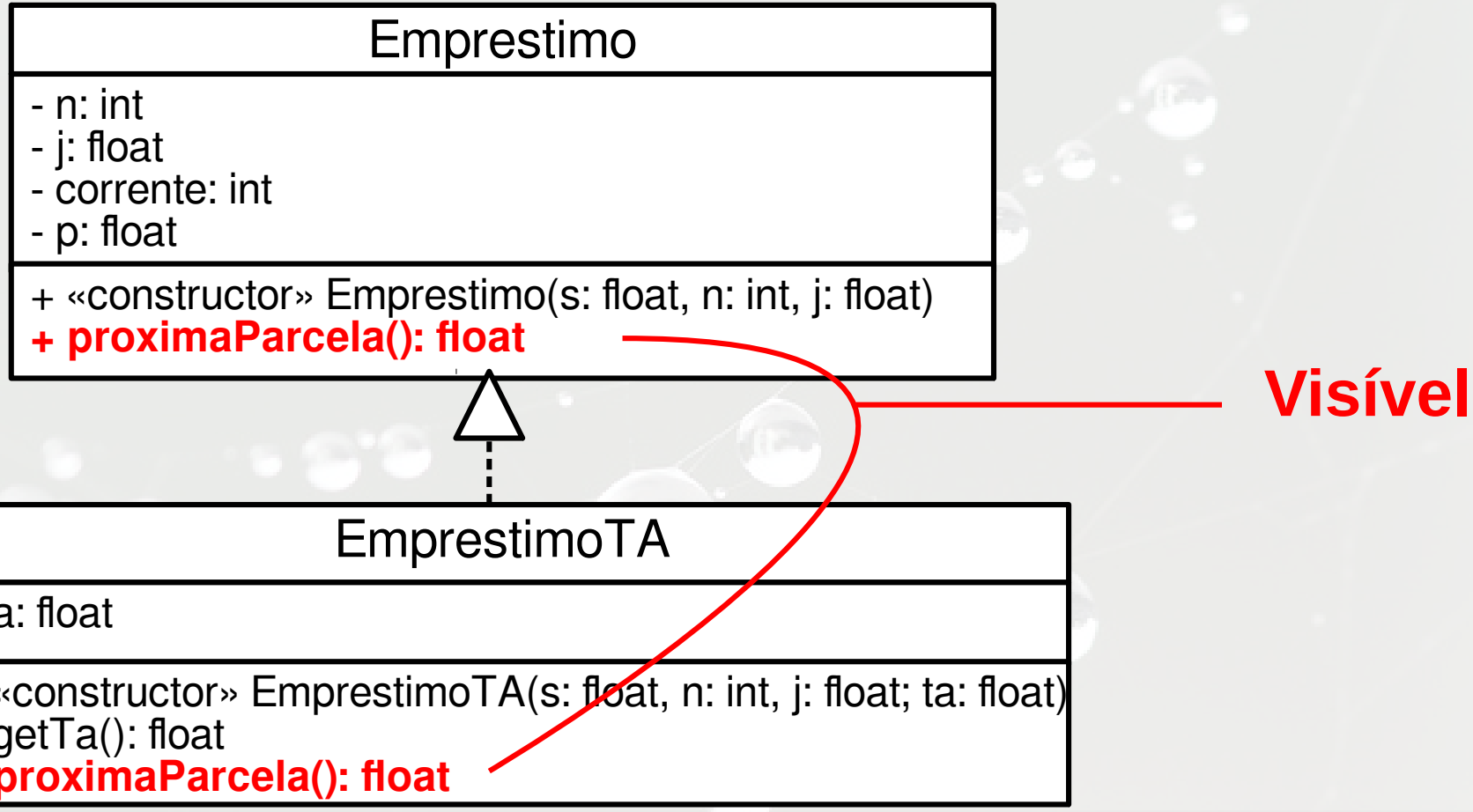

```
float adm = empCT.getTa();
```

Princípio para o Polimorfismo

- Uma variável declarada em uma classe A pode ser instanciada em qualquer subclasse B
- Nesse caso, só estarão “visíveis”(respeitadas as regras de encapsulamento):
  - Atributos e métodos da superclasse A
  - Métodos sobrescritos da subclasse B

Princípio para o Polimorfismo

```
Emprestimo empCT = new EmprestimoTA(200, 5, 1, 15);
```



O Contrário não é Possível

- Uma variável declarada em uma superclasse A pode ser instanciada em qualquer subclasse B
- Porém...
- Uma variável declarada em uma subclasse B **não pode ser instanciada** em sua superclasse A

O Contrário não é Possível

- Uma variável declarada em uma subclasse B **não pode ser instanciada** em sua superclasse A

Por quê?

O Contrário não é Possível

- Uma variável declarada em uma subclasse B **não pode ser instanciada** em sua superclasse A

Por quê?

- Se declararmos na subclasse B, a expectativa é termos todos os métdos e atributos de B, o que não acontece na superclasse A

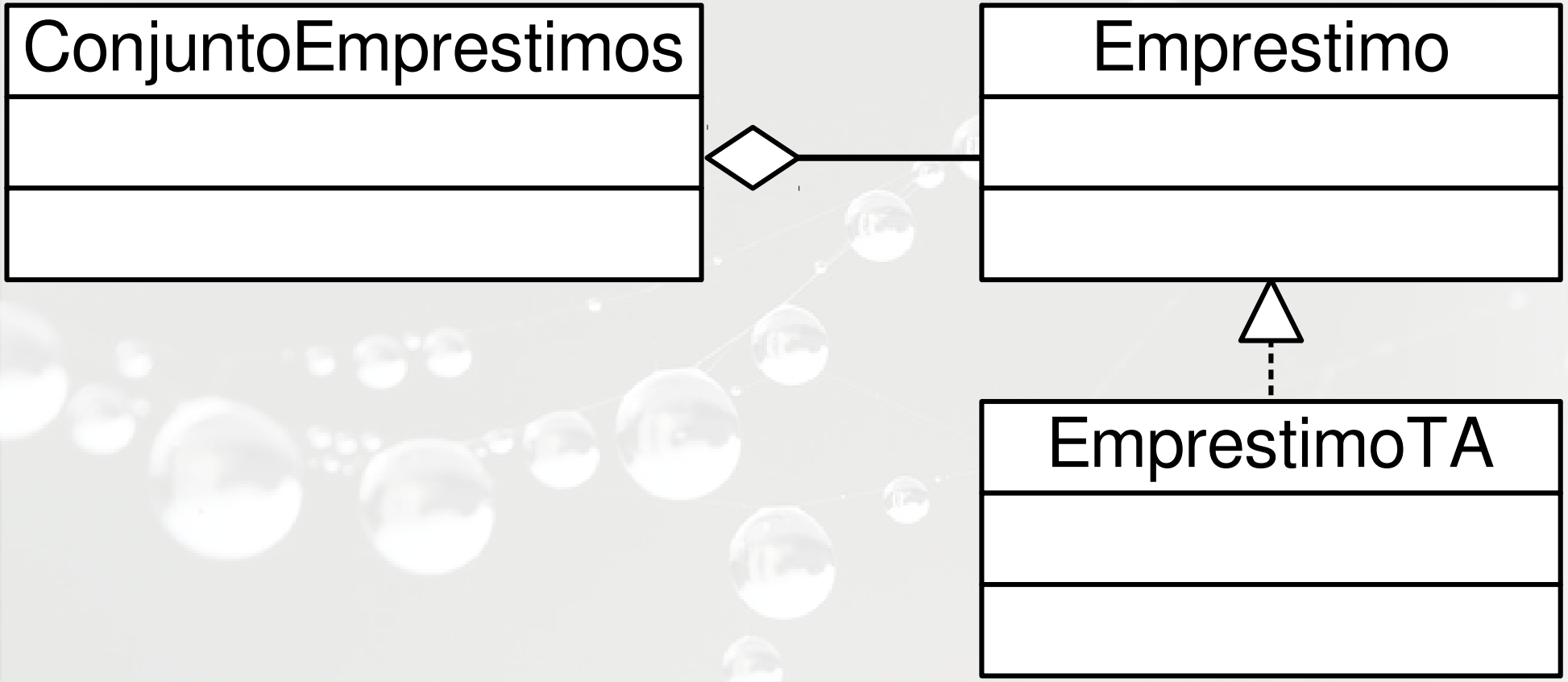
Para que serve o Polimorfismo?

Como controlar um conjunto de empréstimos heterogêneos?

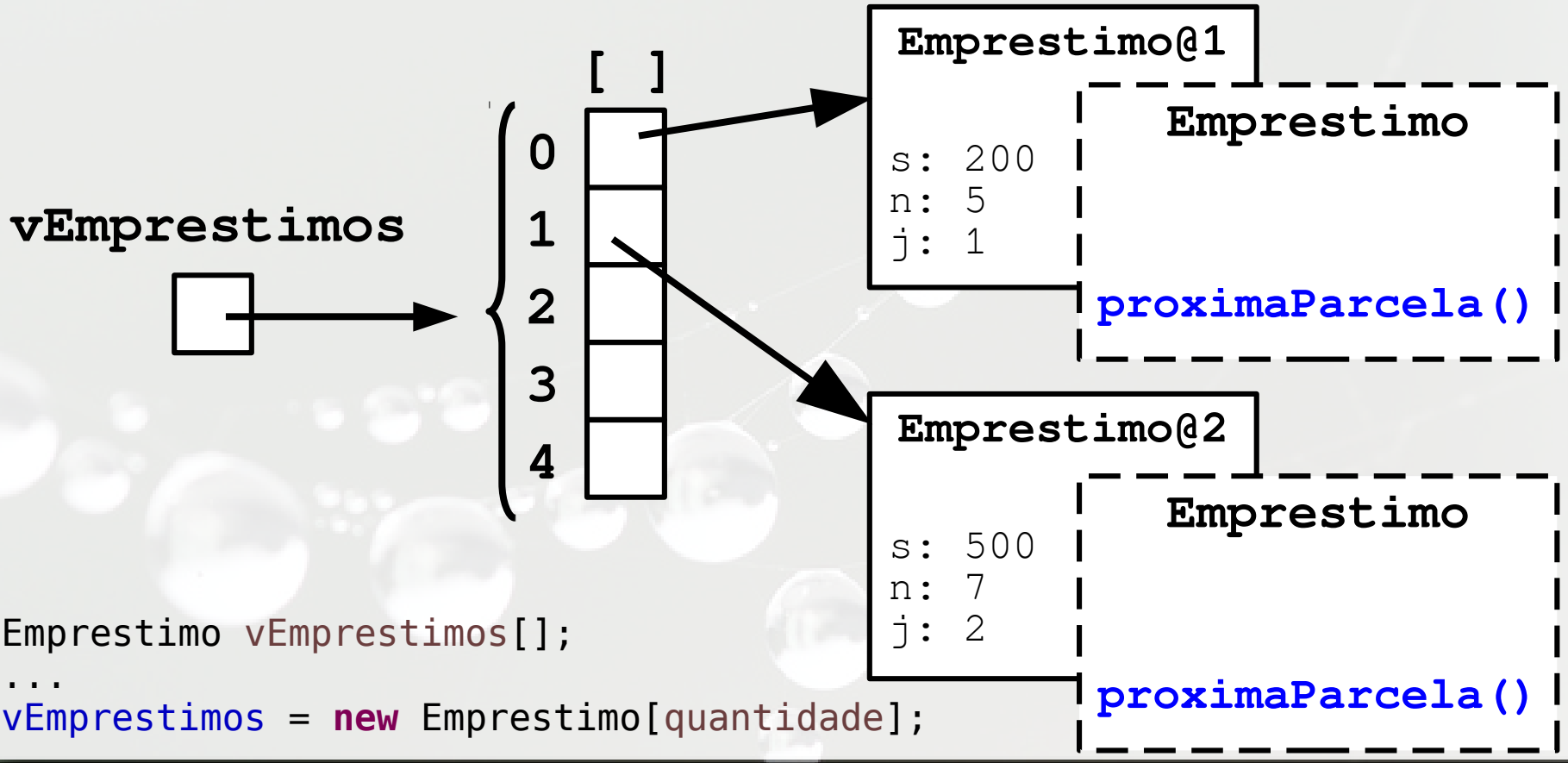
Classe `ConjuntoEmprestimos`

- Controla um conjunto de empréstimos
- Métodos:
  - **construtor** – parâmetro: número máximo de empréstimos;
  - **adicionaEmprestimo** – parâmetro: um objeto da classe empréstimo e o armazena;
  - **proximasParcelas** - mostra as próximas parcelas de todos os empréstimos cadastrados

Classe `ConjuntoEmprestimos`



Vetor de Empréstimos

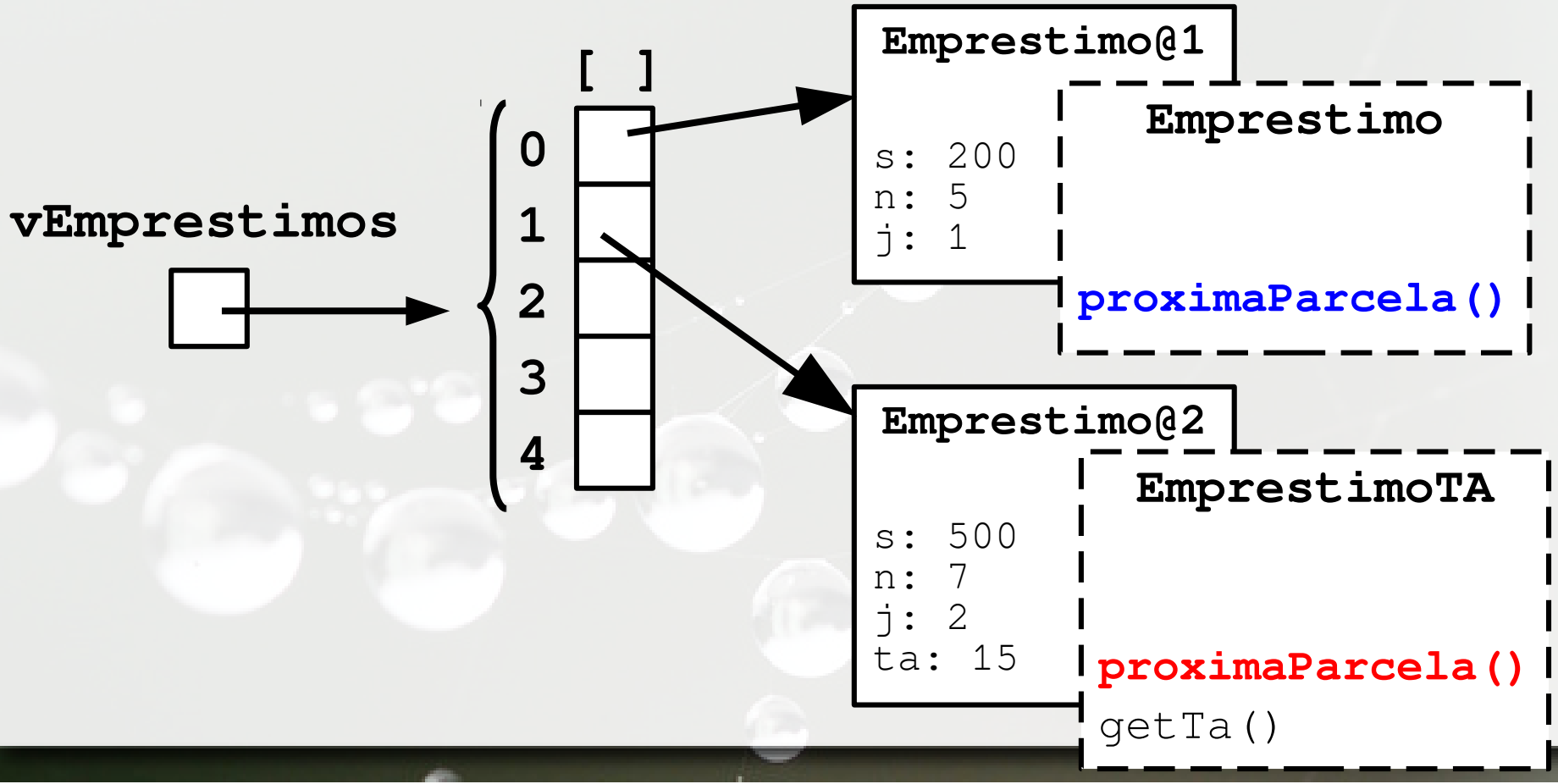



```
Emprestimo vEmprestimos[];
...
vEmprestimos = new Emprestimo[quantidade];
```

Princípio do Polimorfismo no Vetor

- Um vetor declarado em uma superclasse A pode receber instâncias de A e instâncias de quaisquer herdeiros de A
- Seguindo esse princípio, o vetor permite misturar instâncias de classes diferentes

Vetor de Empréstimos



Exercício

- Crie uma classe herdeira de de `Emprestimo` chamada `EmprestimoTAP`
- A classe acrescenta uma taxa administrativa na forma de um percentual (em vez de um valor fixo)
- Crie um exemplo no ConjuntoEmprestimos que inclua as três classes: `Emprestimo`, `EmprestimoTA`, `EmprestimoTAP`

Referências

- Meyer, Bertrand (1997) **Object-Oriented Software Construction** – Second Edition. USA, Prentice-Hall, Inc.

André Santanchè
http://www.ic.unicamp.br/~santanche

Licença

- Estes slides são concedidos sob uma Licença Creative Commons. Sob as seguintes condições: Atribuição, Uso Não-Comercial e Compartilhamento pela mesma Licença.
- Mais detalhes sobre a referida licença Creative Commons veja no link: http://creativecommons.org/licenses/by-nc-sa/3.0/
- Agradecimento a James Ratcliffe [http://www.flickr.com/photos/jamie/1762955591/] por sua fotografia “A spider web after a misty morning” usada na capa e nos fundos, disponível em [http://www.flickr.com/photos/jamie/1762955591/]

  vide licença específica da fotografia.