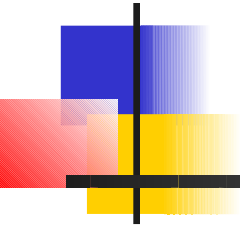


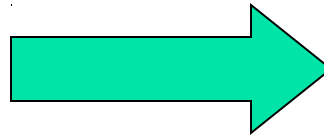
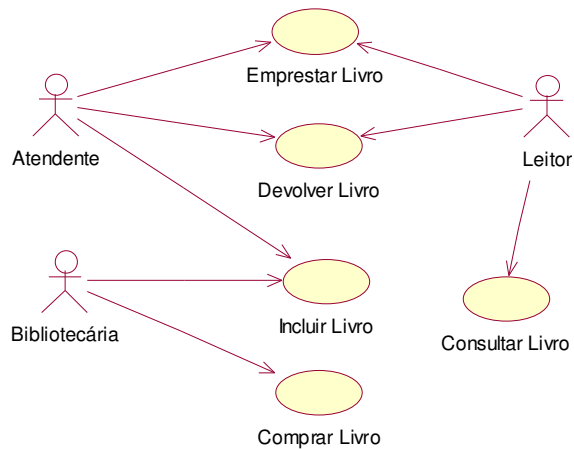
# Do Projeto para a Codificação



Profa Dra Rosana T. V. Braga  
(alguns slides adaptados do material do Prof Raul  
Wazlawick)

# O que já foi visto até agora

## Diagrama de Casos de Uso



## Casos de Uso Completo Abstrato

### Caso de Uso: Emprestar Livro

**Ator Principal:** Atendente

**Interessados e Interesses:**

- Atendente: deseja registrar que um ou mais livros estão em posse de um leitor, para controlar se a devolução será feita no tempo determinado.
- Leitor: deseja emprestar um ou mais livros, de forma rápida e segura.
- Bibliotecário: deseja controlar o uso dos livros, para que não se percam e para que sempre se saiba com que leitor estão no momento.

**Pré-Condições:** O Atendente é identificado e autenticado.

**Garantia de Sucesso (Pós-Condições):** Os dados do novo empréstimo estão armazenados no Sistema. Os livros emprestados possuem status "emprestado".

**Cenário de Sucesso Principal:**

1. O Leitor chega ao balcão de atendimento da biblioteca e diz ao atendente que deseja emprestar um ou mais livros da biblioteca.
2. O Atendente seleciona a opção para realizar um novo empréstimo.
3. O Atendente solicita ao leitor sua carteira de identificação, seja de estudante ou professor.
4. O Atendente informa ao sistema a identificação do leitor.
5. O Sistema exibe o nome do leitor e sua situação.
6. O Atendente solicita os livros a serem emprestados.
7. Para cada um deles, informa ao sistema o código de identificação do livro.
8. O Sistema informa a data de devolução de cada livro.
9. Se necessário, o Atendente desbloqueia os livros para que possam sair da biblioteca.
10. O Leitor sai com os livros.

**Fluxos Alternativos:**

- (1-8). A qualquer momento o Leitor informa ao Atendente que desistiu do empréstimo.
3. O Leitor informa ao Atendente que esqueceu a carteira de identificação.
  1. O Atendente faz uma busca pelo cadastro do Leitor e pede a ele alguma informação pessoal para garantir que ele é mesmo quem diz ser.
4. O Leitor está impedido de fazer empréstimo, por ter não estar apto.
  1. Cancelar a operação.
- 7a. O Livro não pode ser emprestado, pois está reservado para outro leitor.
  1. O Atendente informa ao Leitor que não poderá emprestar o livro e pergunta se deseja reservá-lo.
  2. Cancelar a operação (se for o único livro)
- 7b. O Livro não pode ser emprestado, pois é um livro reservado somente para consulta.
  1. Cancelar a operação (se for o único livro)

# O que já foi visto até agora

## Casos de Uso com substantivos e verbos sublinhados

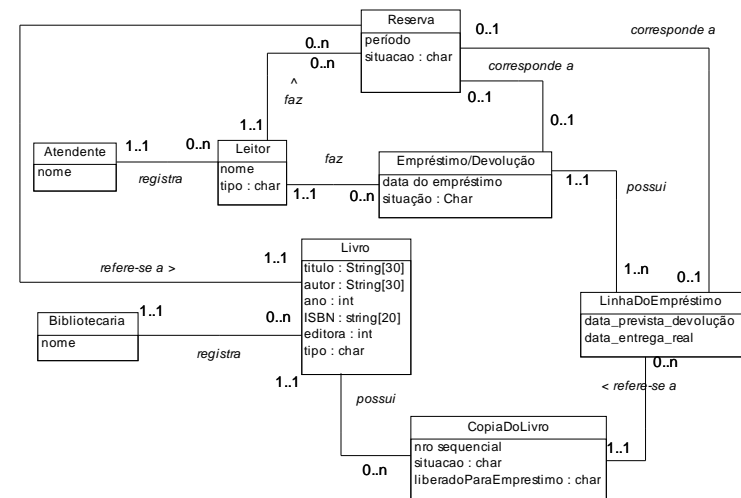
### Caso de Uso 1

1. O Leitor chega ao balcão de atendimento da biblioteca e diz ao atendente que deseja emprestar um ou mais livros da biblioteca.
2. O Atendente seleciona a opção para adicionar um novo empréstimo.
3. O Atendente solicita ao leitor sua carteirinha, seja de estudante ou professor.
4. O Atendente informa ao sistema a identificação do leitor.
5. O Sistema exibe o nome do leitor e sua situação.
6. O Atendente solicita os livros a serem emprestados.
7. Para cada um deles, informa ao sistema o código de identificação do livro.
8. O Sistema informa a data de devolução de cada livro.
9. O Atendente desbloqueia os livros para que possam sair da biblioteca.
10. O Leitor sai com os livros.

### Caso de Uso n

1. O Leitor chega ao balcão de atendimento da biblioteca e diz ao atendente que deseja emprestar um ou mais livros da biblioteca.
2. O Atendente seleciona a opção para adicionar um novo empréstimo.
3. O Atendente solicita ao leitor sua carteirinha, seja de estudante ou professor.
4. O Atendente informa ao sistema a identificação do leitor.
5. O Sistema exibe o nome do leitor e sua situação.
6. O Atendente solicita os livros a serem emprestados.
7. Para cada um deles, informa ao sistema o código de identificação do livro.
8. O Sistema informa a data de devolução de cada livro.
9. O Atendente desbloqueia os livros para que possam sair da biblioteca.
10. O Leitor sai com os livros.

## Modelo Conceitual



# O que já foi visto até agora

Modelo Conceitual  
+  
Casos de Uso

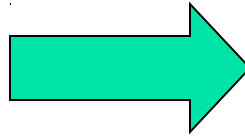
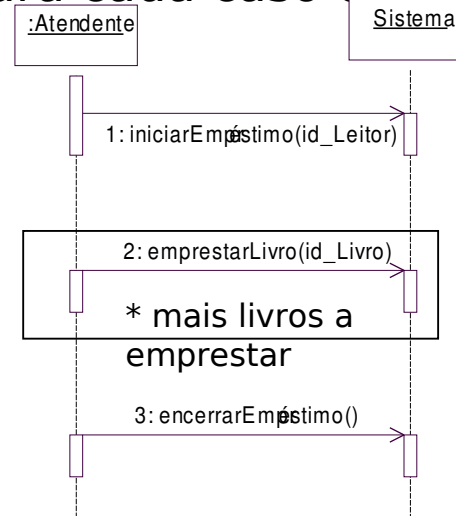
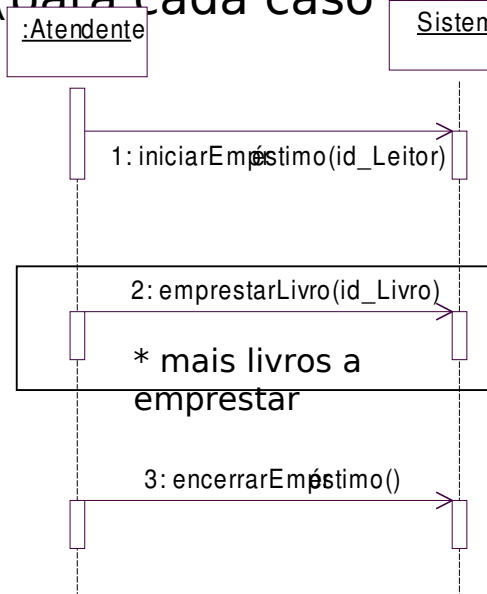


Diagrama de Seqüência do  
Sistema  
(para cada caso de uso)



# O que já foi visto até agora

Diagrama de Seqüência do Sistema  
(para cada caso de uso)



Contrato da Operação  
(para cada operação)

**Operação:** encerrarEmpréstimo()

**Referências Cruzadas:** Caso de uso: “Emprestar Livro”

**Pré-Condições:** Um leitor apto a emprestar livros já foi identificado; pelo menos um livro já foi identificado e está disponível para ser emprestado.

**Pós-Condições:** um novo empréstimo foi registrado; o novo empréstimo foi relacionado ao leitor já identificado na operação “iniciar o empréstimo”; a situação dos livros emprestados foi alterada para “emprestado”.

# O que já foi visto até agora

## Contrato da Operação (para cada operação)

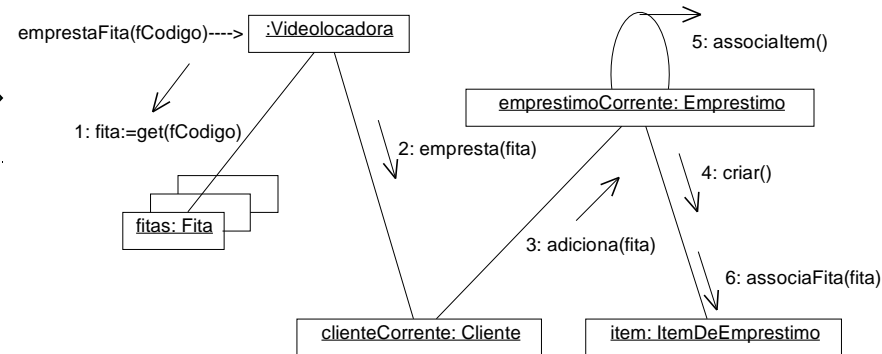
**Operação:** encerrarEmpréstimo()

**Referências Cruzadas:** Caso de uso: “Emprestar Livro”

**Pré-Condições:** Um leitor apto a emprestar livros já foi identificado; pelo menos um livro já foi identificado e está disponível para ser emprestado.

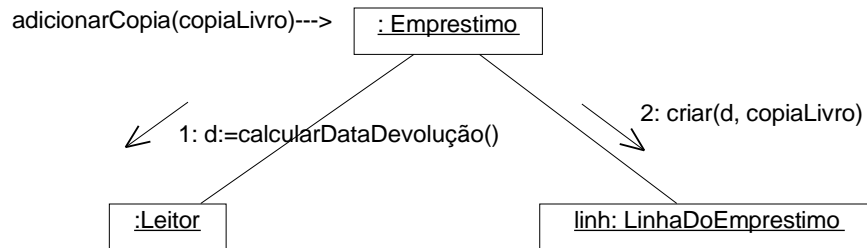
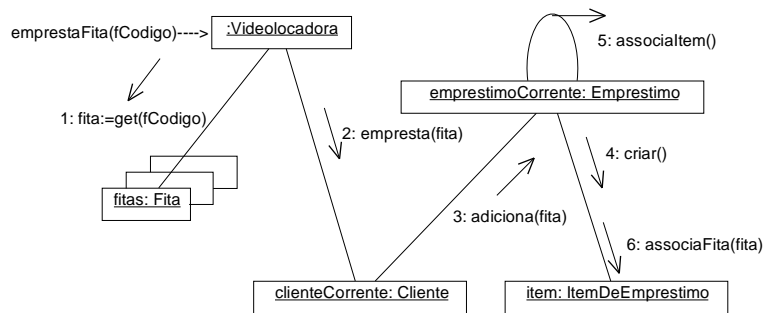
**Pós-Condições:** um novo empréstimo foi registrado; o novo empréstimo foi relacionado ao leitor já identificado na operação “iniciar o empréstimo”; a situação dos livros emprestados foi alterada para “emprestado”.

## Diagrama de Colaboração (para cada operação)

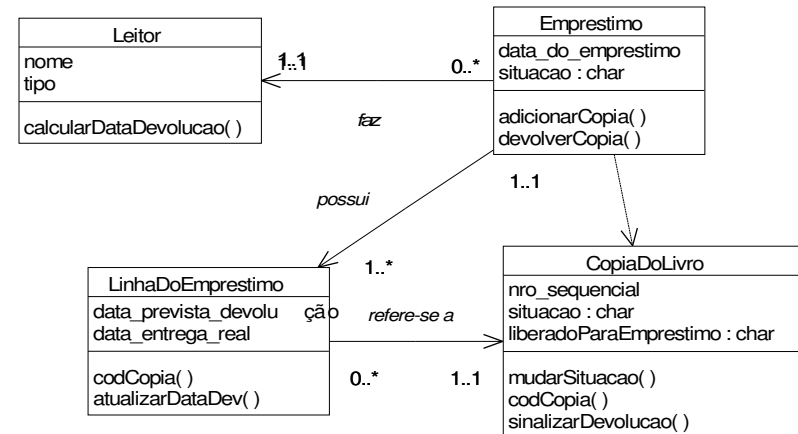


# O que já foi visto até agora

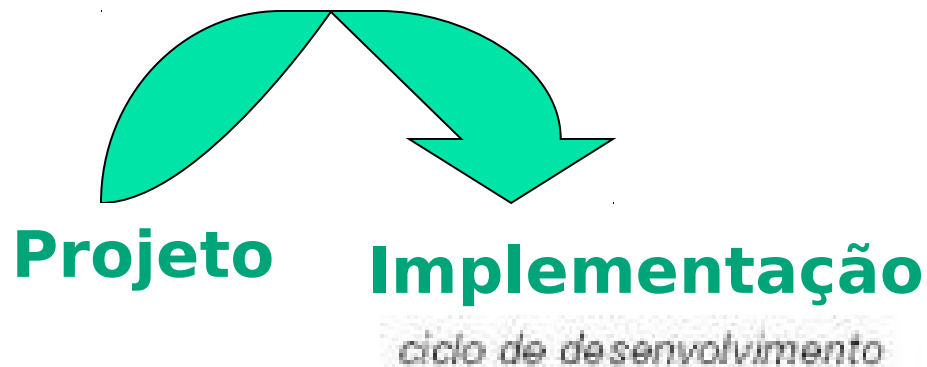
## Diagramas de Colaboração (para todas as operações)



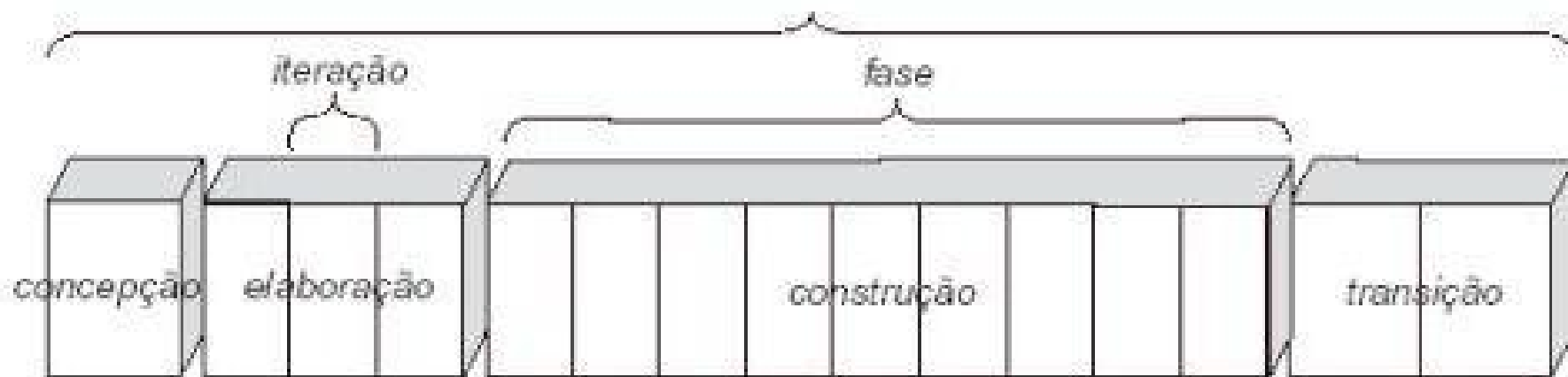
## Diagrama de Classes de Projeto



# As Fases do PU



?







# Como mapear o projeto para a implementação?

---

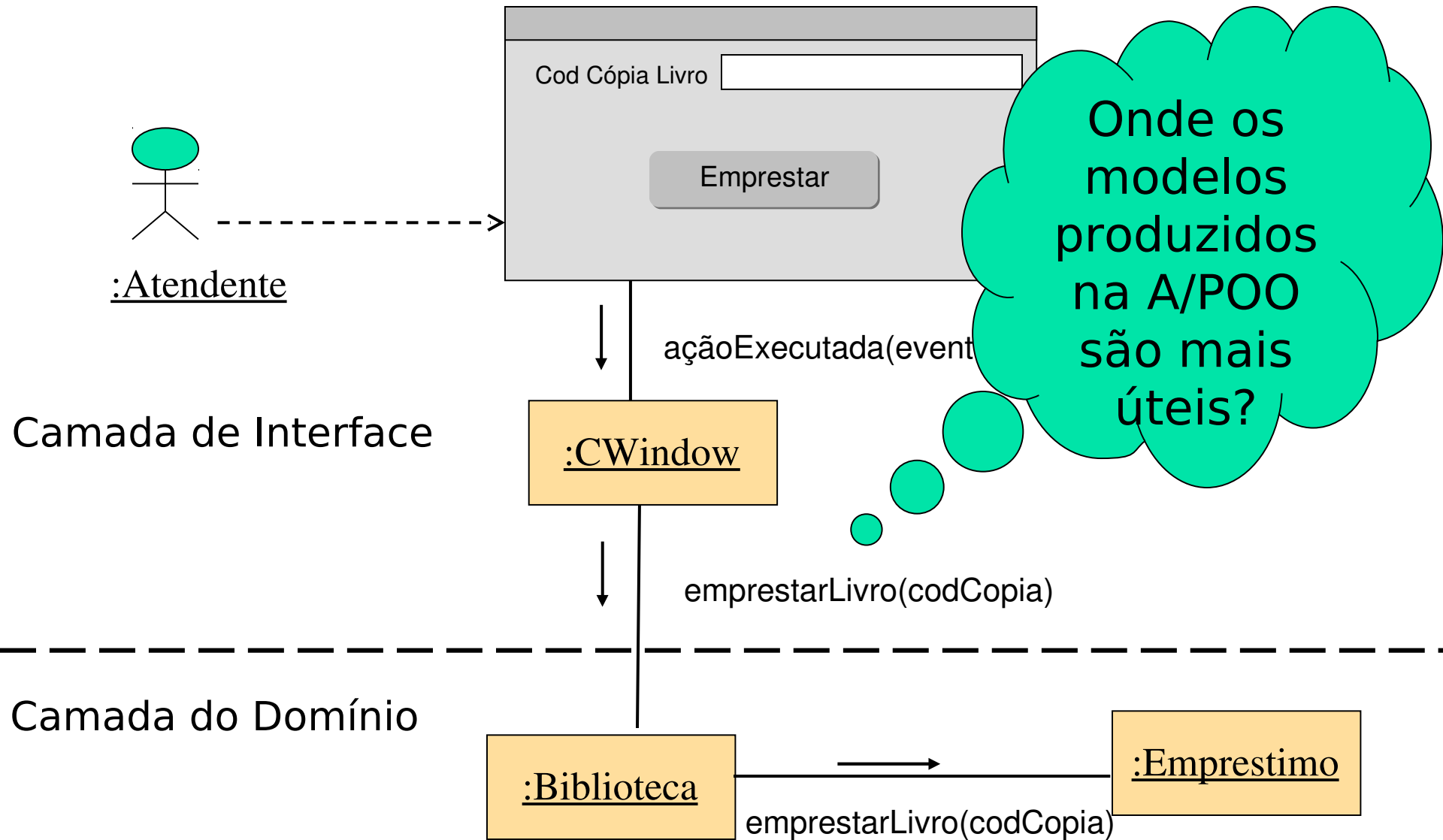
- Resultados obtidos no projeto são o ponto de partida, mas muito trabalho ainda tem que ser feito
- Muitas alterações podem ocorrer e problemas podem surgir e precisam ser solucionados
- Prepare-se: pode haver mudanças e desvios de projeto!!!



# Como mapear o projeto para a implementação?

---

- Protótipos e código exploratório pode ter sido produzido na fase de projeto
- Ferramentas CASE (geração semi-automática do código) podem ajudar
- Código a ser escrito
  - Classes
  - Interfaces
  - Métodos
- Linguagem a ser usada como exemplo: JAVA



**Resposta: Camada de domínio!!!**



# Definição de Classes

---

- Uma classe de programa deve ser criada para cada classe do Diagrama de Classes de Projeto.
- Método Criar → gera construtores em Java
- Tipos de atributos → podem ser adotados tipos nativos da linguagem ou serem criados tipos a partir dos tipos nativos.
- Definição e assinaturas dos métodos



# Classe em Java

---

Cliente
+ nome: String
+ debito: Moeda
+ idade: Inteiro

```
class Cliente {  
    private String nome;  
    private Float debito;  
    private Integer idade;  
}
```



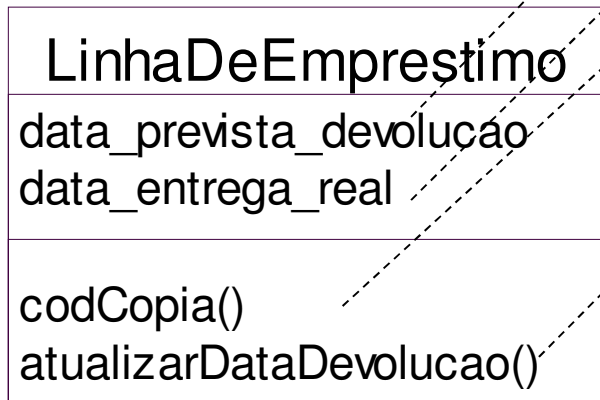
# Classe em Java

---

```
class Cliente {  
    private String nome;  
    private Float debito;  
    private Integer idade;  
  
    public void setNome(String nome) { this.nome = nome; }  
    public void setDebito(Float debito) { this.debito = debito; }  
    public void setIdade(Integer idade) { this.idade = idade; }  
  
    public String getNome() { return nome; }  
    public Float getDebito() { return debito; }  
    public Integer getIdade() { return idade; }  
}
```

# Atributos comuns e assinatura de métodos

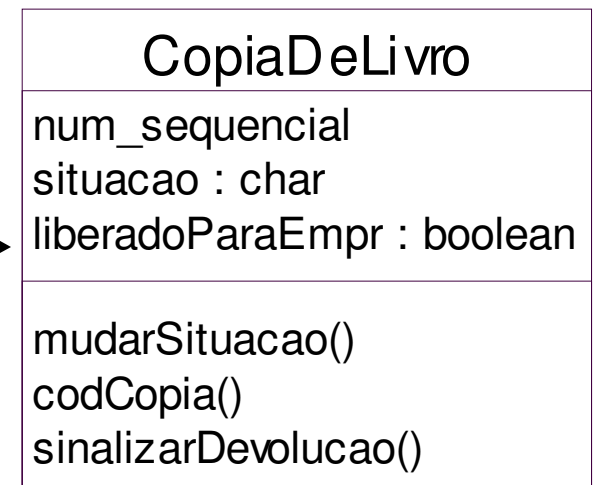
```
public class LinhaDeEmprestimo
{
    public LinhaDeEmprestimo
        (CopiaDeLivro copia, Date data_prev_dev;
    private Date data_prev_dev;
    private Date data_entrega_real;
    public int codCopia();
    public void atualizarDataDev(Date dt)
}
```



*Refere-se a*

0..\*

1





# Atributos referenciais

---

- Um atributo referencial é um atributo que referencia um outro objeto complexo e não um tipo primitivo (tal como uma string, por exemplo)
- Os atributos referenciais de uma classe são sugeridos pelas associações e pela navegabilidade em um diagrama de classes.
- No diagrama de classes, os atributos referenciais estão normalmente implícitos (ao invés de explícitos como os demais atributos).

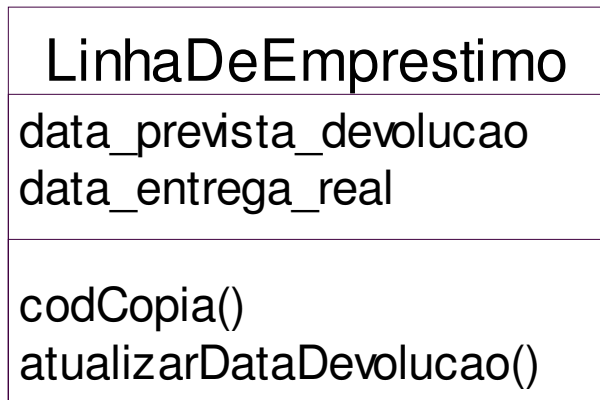


# Atributo referencial

Atributo  
Simples

Atributo  
Referencial

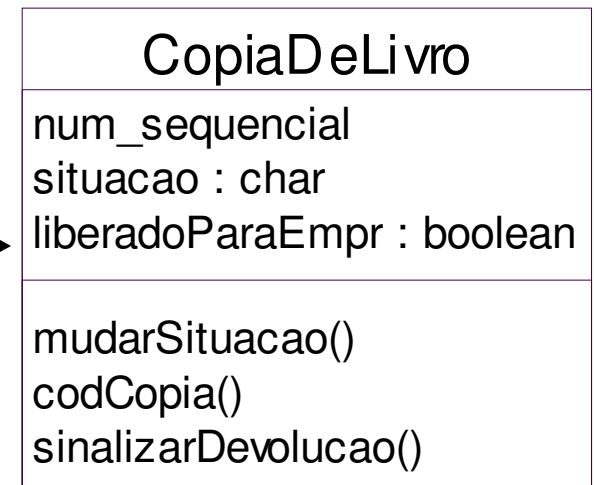
```
public class LinhaDeEmprestimo
{
    public LinhaDeEmprestimo
        (CopiaDeLivro copia, Date data_prev_dev);
    private Date data_prev_dev;
    private Date data_entrega_real;
    private CopiaDeLivro copiaLivro;
    public int codCopia();
    public void atualizarDataDev(Date dt)
}
```



*Refere-se a*

0..\*

1





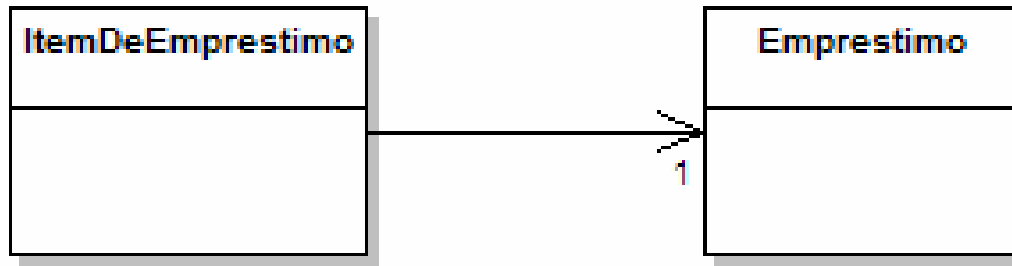
# Associação para 1

---

- Como associação é estritamente para 1, então não é possível destruir a associação, e, portanto, o método para destruir a associação não deve ser implementado.
- Como a associação para 1 é obrigatória para o objeto na origem, o método criador da classe deve ter como parâmetro o elemento a ser associado para que desde o momento da criação todas as instâncias da classe na origem da associação estejam consistentes.



# Associação para 1



```
class ItemDeEmprestimo {
    private Emprestimo emprestimo;

    public ItemDeEmprestimo(Emprestimo emprestimo) {
        this.associaEmprestimo(emprestimo )
    }
    public void associaEmprestimo(Emprestimo emprestimo) {
        this.emprestimo = emprestimo;
    }
    public Emprestimo getEmprestimo() {
        return emprestimo;
    }
}
```

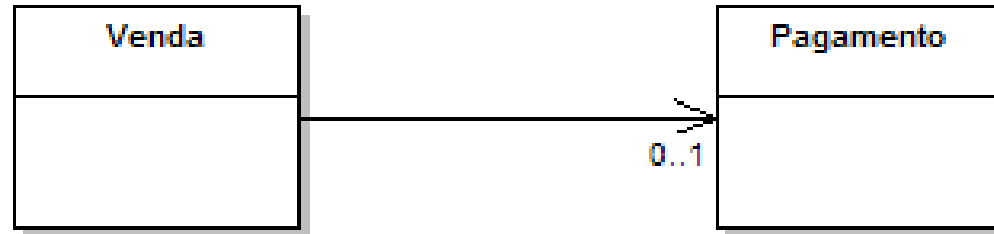


# Associação para 0..1

---

- ◆ É possível destruir a associação e, portanto deve ser implementado o método correspondente.
- ◆ Não é necessário passar um objeto como parâmetro para o método criador, pois a associação para 0..1 não é obrigatória.

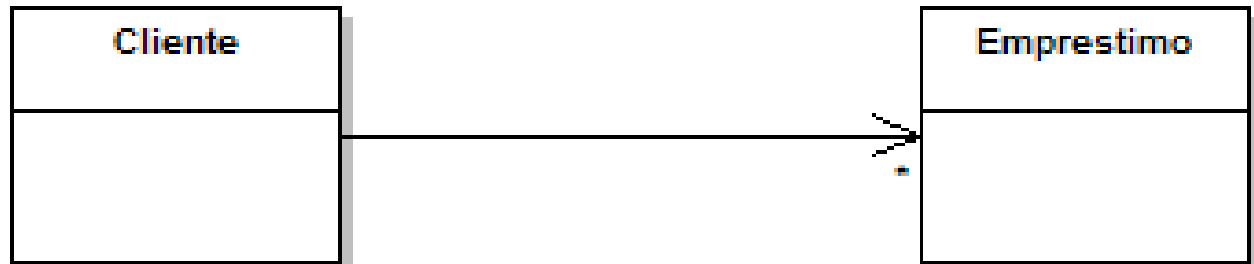
# Associação para 0..1



```
class Venda {
    private Pagamento pagamento;

    public Venda() { }
    public void associaPagamento(Pagamento pagamento) {
        this.pagamento = pagamento;
    }
    public void desassociaPagamento() {
        this.pagamento = null;
    }
    public Pagamento getPagamento() {
        return pagamento;
    }
}
```

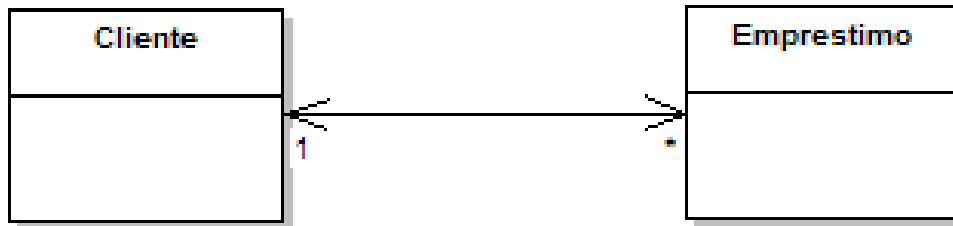
# Associação para \*



```
class Cliente {
    private Set emprestimos = new HashSet();

    public Cliente () {}
    public void adicionaEmprestimo(Emprestimo emprestimo) {
        this.emprestimos.add(emprestimo);
    }
    public void removeEmprestimo(Emprestimo emprestimo) {
        this.emprestimos.remove(emprestimo);
    }
    public Set getEmprestimos () {
        return Collections.unmodifiableSet(emprestimos);
    }
}
```


# Associação bidirecional



```
class Cliente {
    private Set empréstimos = new HashSet();

    public Cliente () {}
    public void adicionaEmpréstimoAux(Empréstimo empréstimo) {
        empréstimos.add(empréstimo);
    }
    public void removeEmpréstimoAux(Empréstimo empréstimo) {
        empréstimos.remove(empréstimo);
    }
    public void adicionaEmpréstimo(Empréstimo empréstimo) {
        if (empréstimo.getClient() != null) {
            empréstimo.getClient().removeEmpréstimoAux(empréstimo);
        };
        this.adicionaEmpréstimoAux(empréstimo);
        empréstimo.associaClienteAux(this);
    }
    public void removeEmpréstimo(Empréstimo empréstimo) {
        this.removeEmpréstimoAux(empréstimo);
        empréstimo.destroi();
    }
    public Set getEmpréstimos() {
        return empréstimos;
    }
}
```

# Associação bidirecional



```
class Emprestimo {
    private Cliente cliente;

    public Emprestimo(Cliente cliente) {
        this.associaCliente(cliente);
    }
    public void associaClienteAux(Cliente cliente) {
        this.cliente = cliente;
    }
    public void associaCliente(Cliente cliente) {
        if (this.cliente != null) {
            this.cliente.removeEmprestimoAux(this);
        };
        this.associaClienteAux(cliente);
        cliente.adicionaEmprestimoAux(this);
    }
    public Cliente getCliente() {
        return cliente;
    }
}
```





# Atributos Referenciais e nomes de Papéis

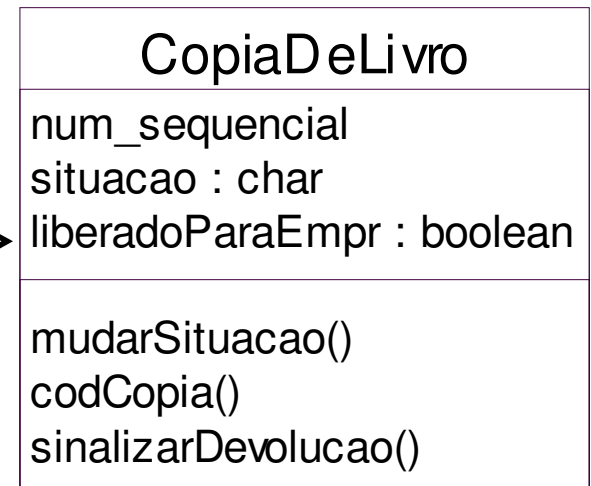
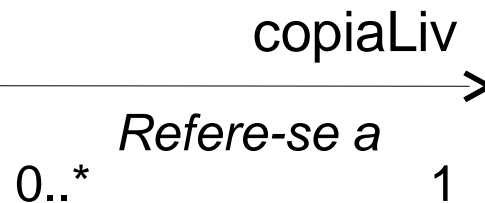
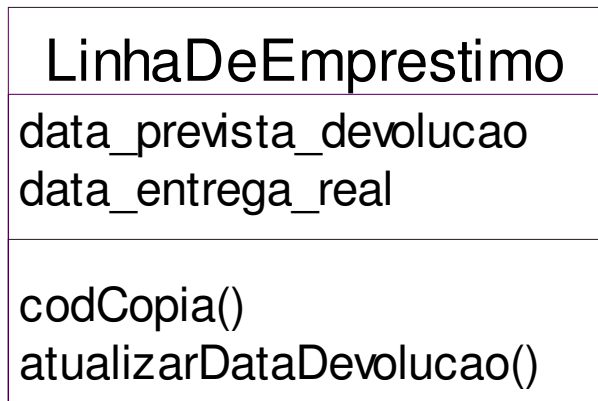
---

- O nome de papel identifica o papel da classe na associação e fornece, freqüentemente, algum contexto semântico sobre a sua natureza.
- Se houver um nome de papel no diagrama de classes, utilize-o como base para o nome do atributo referencial durante a geração de código.

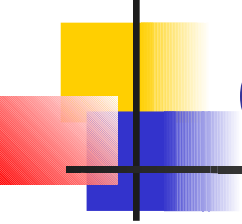
```

public class LinhaDeEmprestimo
{
    ...
    private Date data_prev_dev;
    private CopiaDeLivro copiaLiv;
}

```



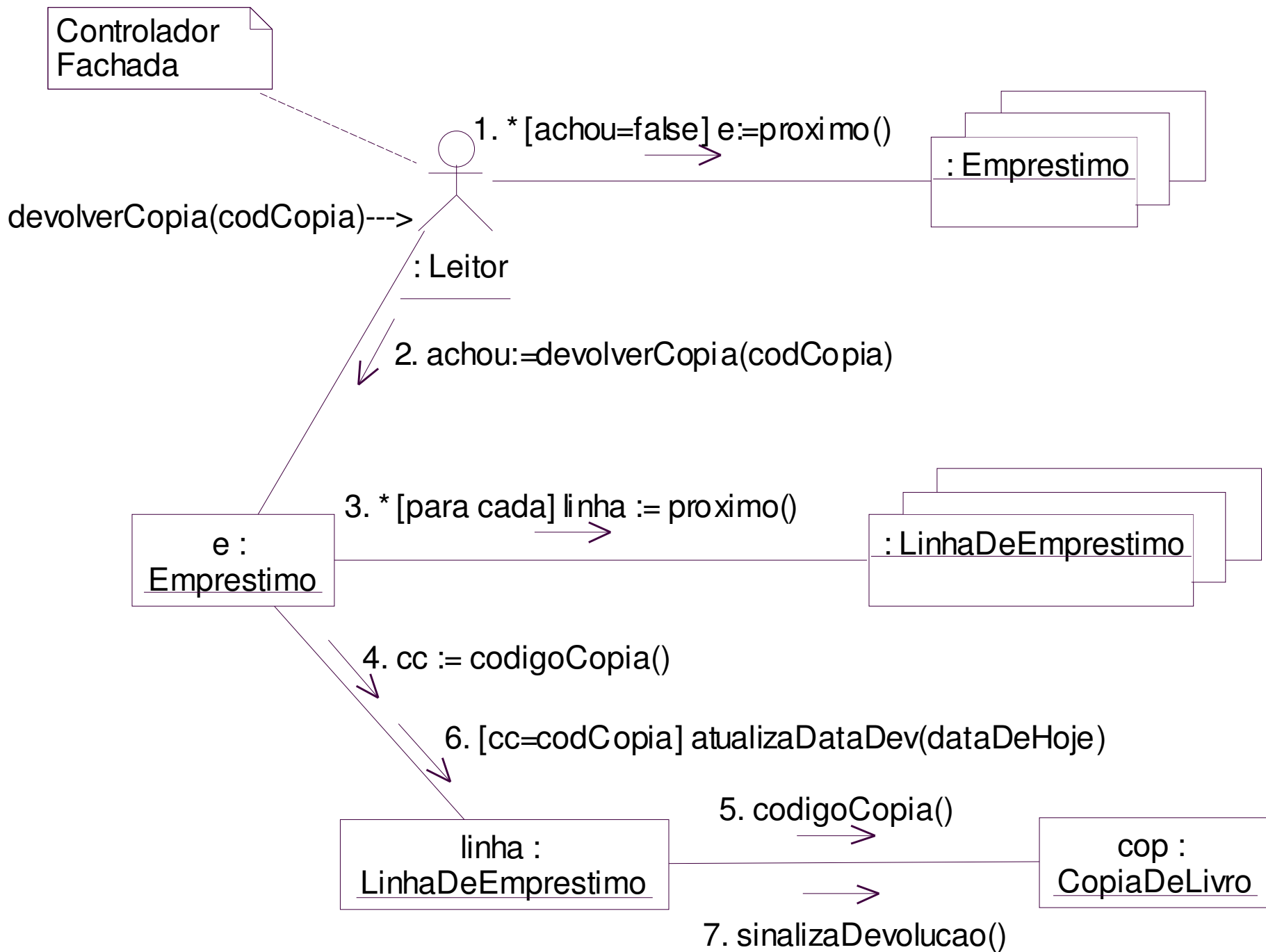
Nome do papel usado  
como nome de atributo



# Criar métodos a partir dos diagramas de colaboração

---

- A seqüência de mensagens de um diagrama de colaboração é traduzida para uma série de comandos de programação na definição do método.
- Os métodos de acesso (set e get) não serão ilustrados, por simplicidade, mas devem ser criados.
- Exemplo: A classe Emprestimo no diagrama de colaboração do método (operação) *devolverCopia*.





# Implementação de devolverCopia()

---

- A mensagem devolverCopia é enviada a Leitor → que o método devolverCopia é definido em Leitor:
  - `public void devolverCopia ( int codCopia)`
- A mensagem devolverCopia é delegada a cada empréstimo feito por leitor → que o método devolverCopia é definido também em Emprestimo:
  - `public boolean devolverCopia ( int codCopia)`



# Implementação de devolverCopia(codCopia)

```
public class Leitor
```

```
{
```

```
    private String nome;
```

```
    private Char[] tipo;
```

```
    private Boolean achou=false;
```

```
    private List emprestimos = new ArrayList();
```

```
    public void devolverCopia(int codCopia)
```

```
    {
```

```
        Iterator i = emprestimos.iterator();
```

```
        while (i.hasNext()) && (!achou) {
```

```
            Emprestimo e = (Emprestimo) i.next();
```

```
            achou=e.devolverCopia(codCopia)}
```

```
        }
```

```
    }
```

■ Na classe  
Leitor



# Implementação de devolverCopia(codCopia)

---

- Na classe Emprestimo

```
public class Emprestimo
{
    private Date data_de_emprestimo;
    private Char[] situacao;
    private int cc=0;
    private List linhas = new ArrayList();
```

```
...
```



# Implementação de devolverCopia(codCopia)

---

- Na classe Emprestimo (continuação)

```
public Boolean devolverCopia(int codCopia)
```

```
    private Boolean ach = false;  
    Iterator i = linhas.iterator();  
    Date dataDeHoje = new Date();  
    while (i.hasNext()) && (!ach) {  
        LinhaDeEmprestimo linha = (LinhaDeEmprestimo) i.next();  
        cc=linha.codigoCopia();  
        if (cc==codCopia) {  
            linha.atualizaDataDev(dataDeHoje);  
            ach := true;  
        }  
    }  
    return ach;  
}
```





# Exceções e Tratamento de Erros

---

- Até aqui, o tratamento de erros foi ignorado (intencionalmente), mas deve ser considerado na fase de projeto em sistemas reais.
- Por exemplo: os contratos podem ser anotados com observações sobre situações típicas de erros e o plano geral de tratamento desses erros.
- A UML não tem uma notação especial para ilustrar exceções.



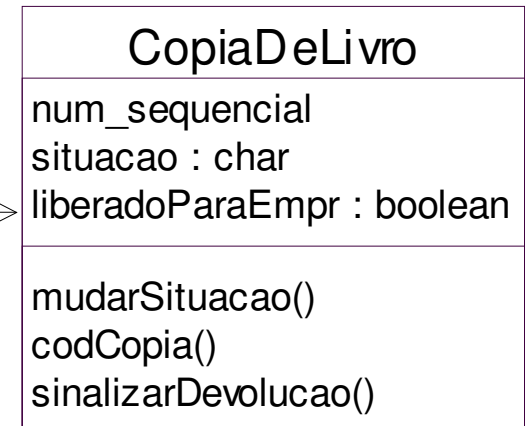
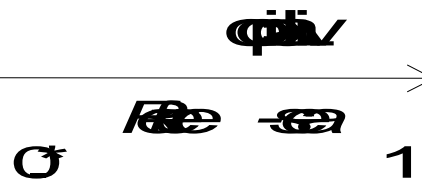
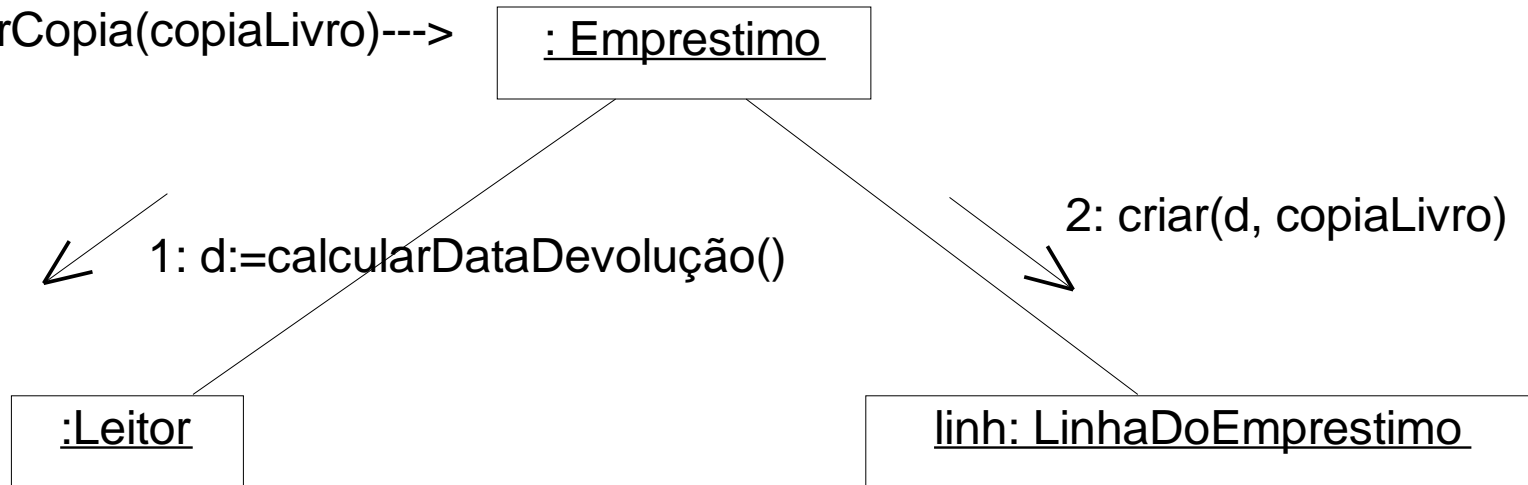
# Método criar()

---

- O método criar é utilizado para criar uma nova instância do objeto, preferencialmente aplicando-se o padrão Criador
- Em Java, o método criador é um método que possui o mesmo nome da classe e recebe como parâmetros todos os dados necessários para inicializar o novo objeto criado

# Exemplo de método criar()

adicionarCopia(copiaLivro)--->





# Código da classe LinhaDeEmprestimo

---

```
public class LinhaDeEmprestimo
{
    // atributos
    private Date data_prev_dev;
    private Date data_entrega_real;
    private CopiaDeLivro copiaLiv;
    // metodos
    public LinhaDeEmprestimo(CopiaDeLivro copia, Date
dtp)
    { this.data_prev_dev = dtp;
      this.copiaLiv=copia;
    }
    public int codCopia();
    public void atualizarDataDevolucao(Date dtdv);
```



# Ordem de Implementação

---

- Podem ser implementadas e testadas na seguinte ordem:
  - Das classes com acoplamento mais baixo para as classes com acoplamento mais alto.
- Exemplo: começar por Livro ou Leitor. Em seguida, as classes que dependem de implementações prévias: Emprestimo ou LinhaDeEmprestimo, e assim por diante.