



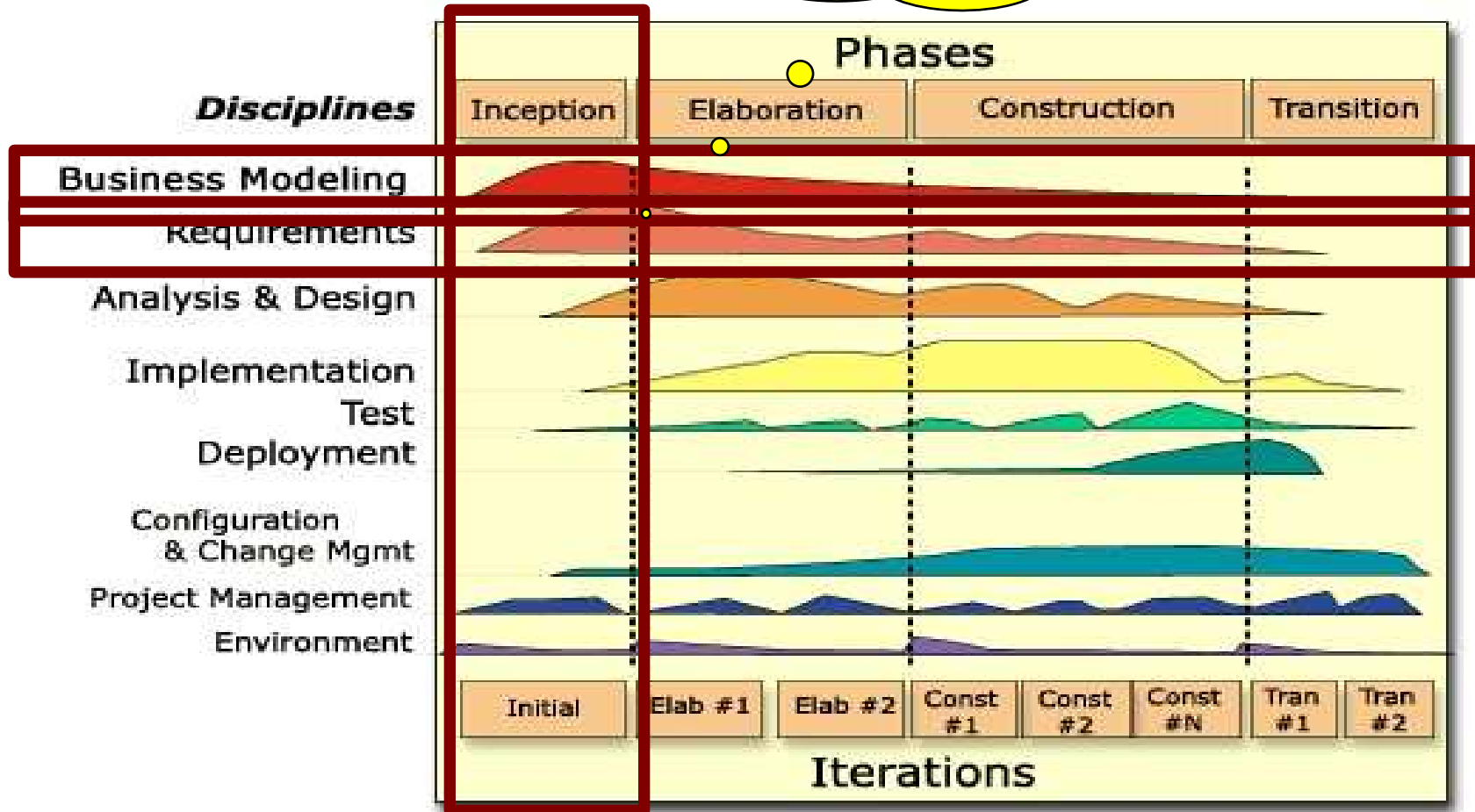
# Desenvolvimento de Sistemas Software

## Aula Teórica 11: Diagramas de Sequência I



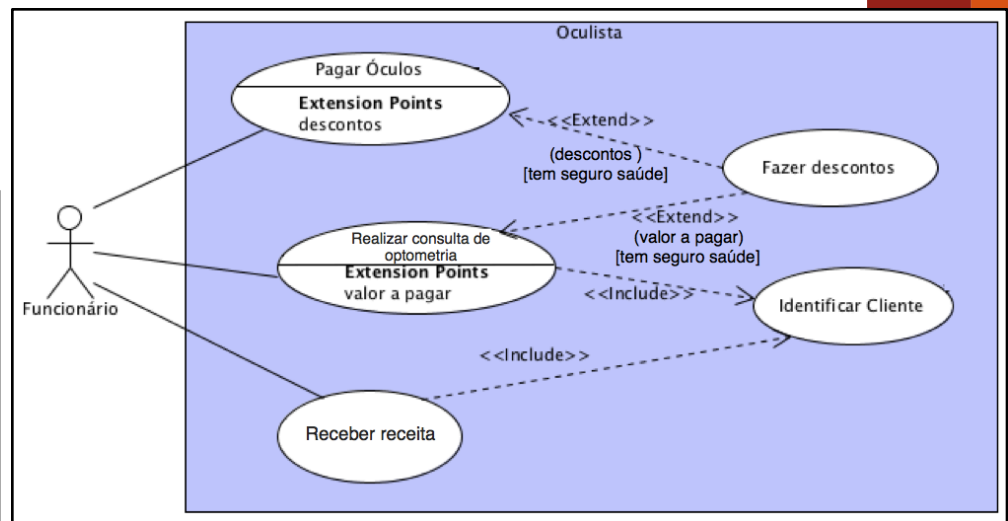
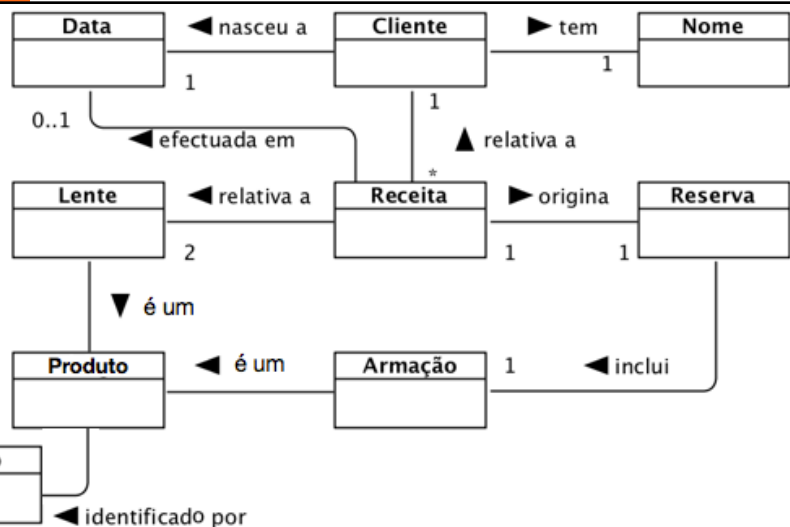
# Ponto da situação...

- Modelo de Domínio
- Modelo de Use Case



- guiado por casos de uso (use cases)
- centrado na arquitectura do sistema a desenvolver
- iterativo e incremental

# O que temos...



## Use Case: Receber receita

Descrição: Funcionário processa a receita de um cliente  
 Pré-condição: Existe papel para imprimir talões  
 Pós-condição: Pedido de óculos fica registado  
 Comportamento normal:

1. <<include>> identificar cliente
2. Funcionário indica código de armação e de lentes
3. Sistema procura produto e apresenta detalhes
4. Funcionário confirma
5. Sistema regista reserva e imprime talão

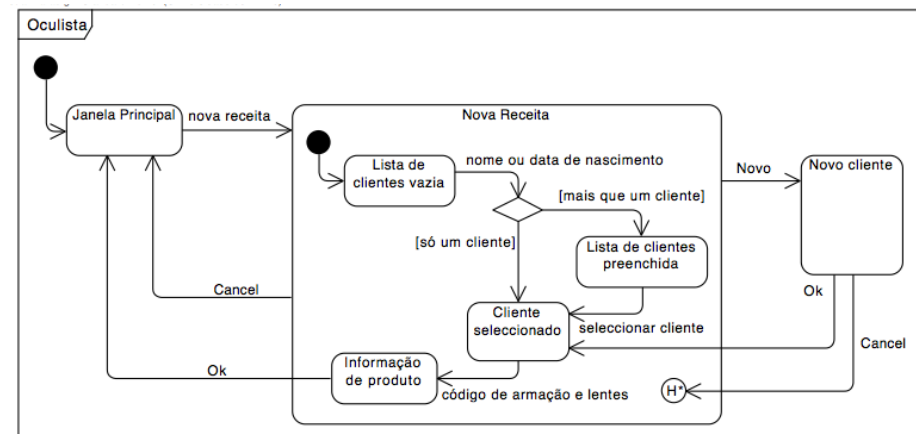
## Excepção

- 4.1. Funcionário não confirma produto
- 4.2. Sistema cancela reserva

**Registo de receitas**

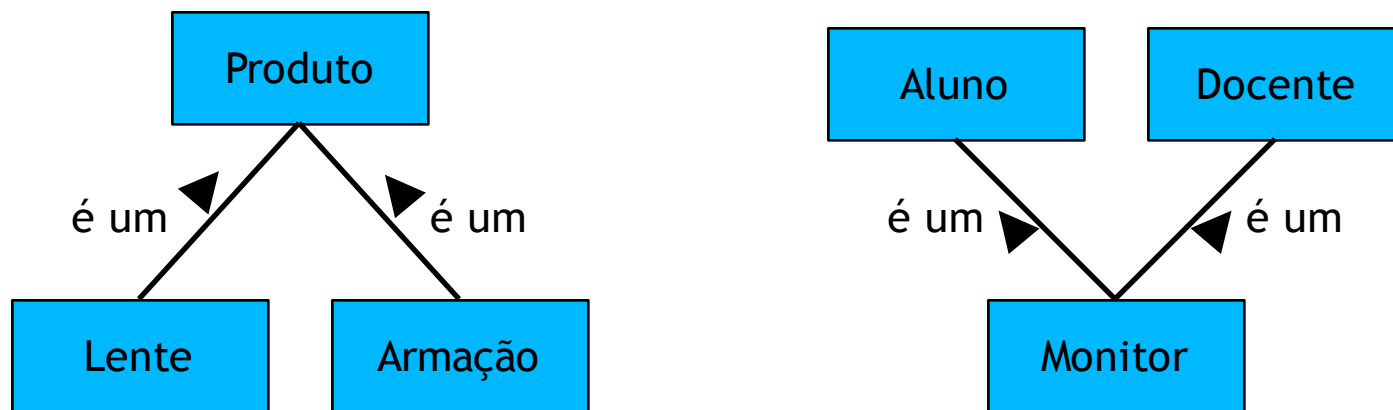
Nome Cliente:  Data Nasc.:

Armações... Lentes Esq./Dir.





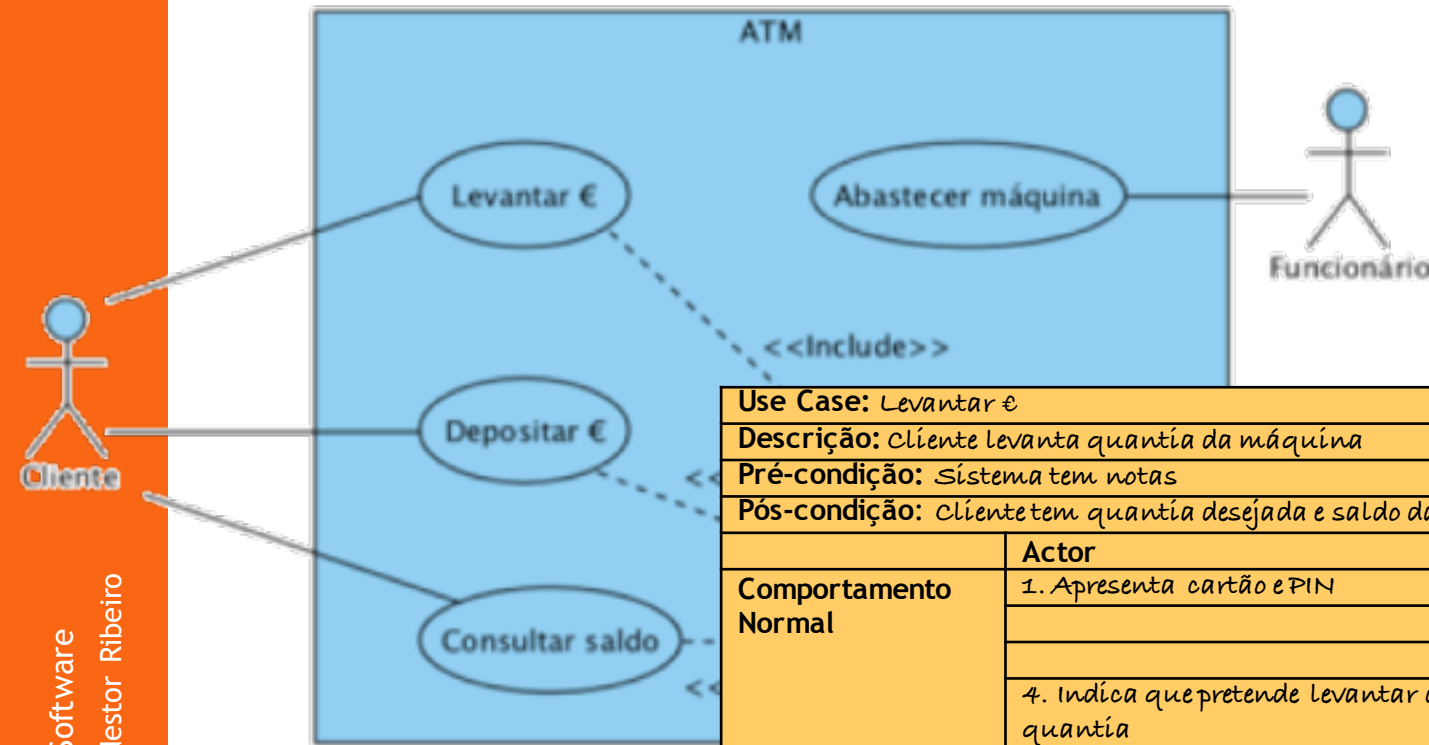
# Arquitetura da Camada de Negócio?



- Demasiado cedo para tomar decisões
- É necessário considerar o comportamento



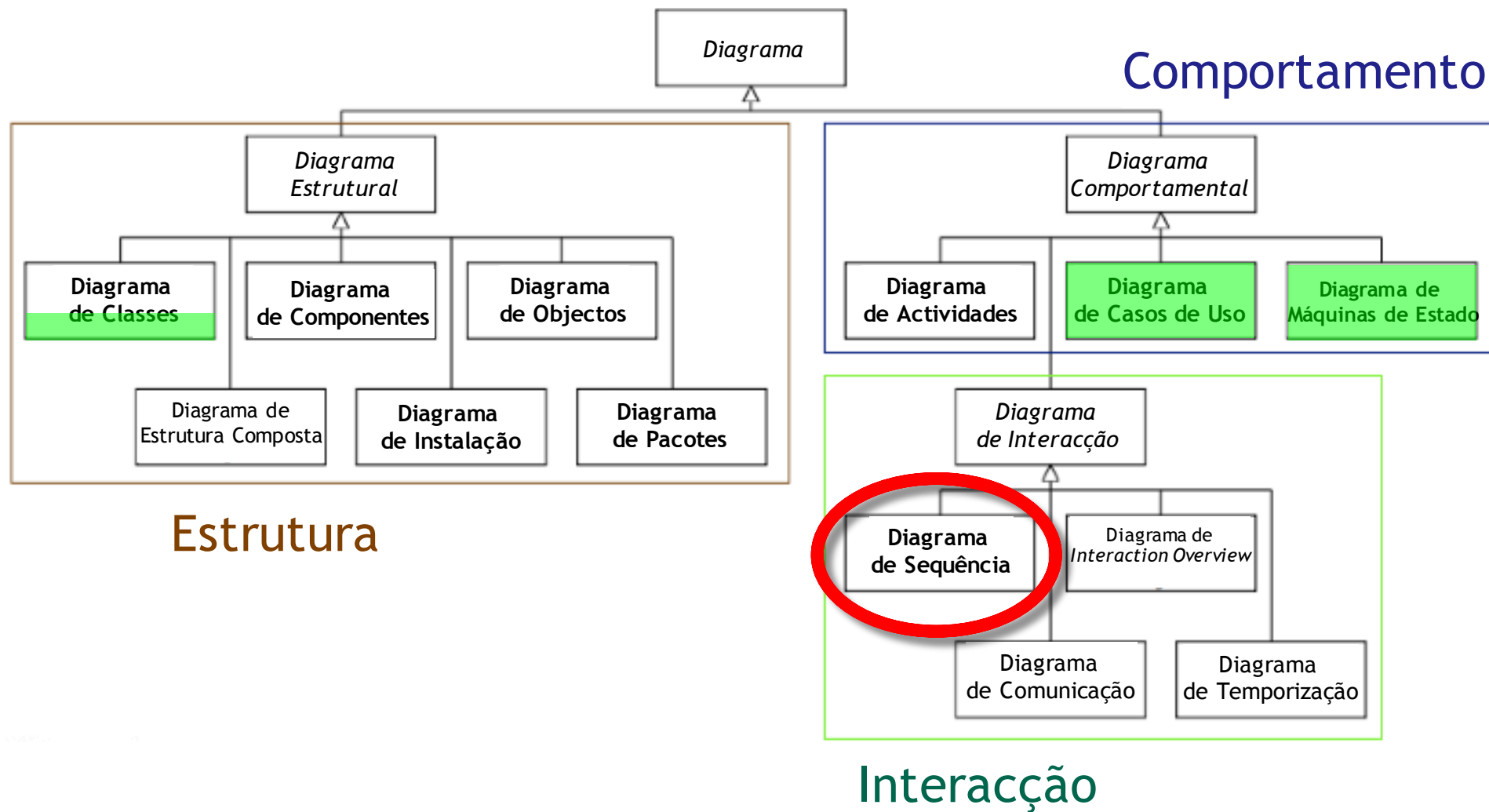
# Guiado por *Use Cases*



<b>Use Case:</b> <i>Levantar €</i>		
<b>Descrição:</b> <i>Cliente levanta quantia da máquina</i>		
<b>Pré-condição:</b> <i>Sistema tem notas</i>		
<b>Pós-condição:</b> <i>Cliente tem quantia desejada e saldo da conta foi actualizado</i>		
	Actor	Sistema
Comportamento Normal	1. Apresenta cartão e PIN	
		2. Valida acesso
		3. Apresenta opções
	4. Indica que pretende levantar dada quantia	
		5. Processa levantamento da quantia
		6. Fornece quantia, talão e devolve cartão
	7. Retira notas, talão e cartão	
Comp. Alternativo [sem papel] (passo 5)		5.1. Avisa de impossibilidade de emitir talão e pergunta se deve continuar
	5.2. diz que sim	
		5.3. Processa levantamento da quantia
		5.4. Fornece quantia e devolve cartão
	5.5. Retira notas e cartão	



# Diagramas da UML 2.x





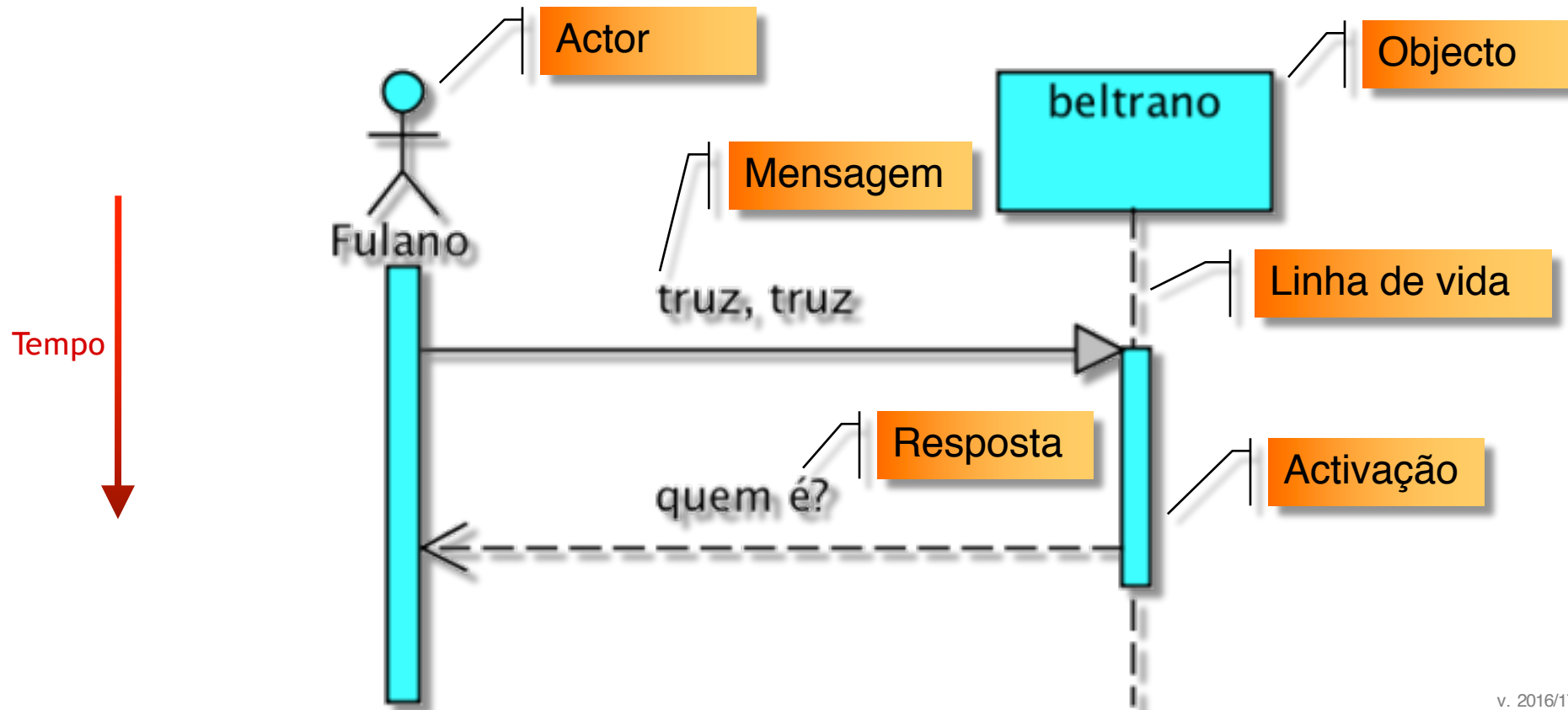
# Diagramas de Interação

- Um tipo de Diagrama Comportamental
- Descrevem como um conjunto de objectos coopera para realizar um dado comportamento
  - modelam as interações entre os objectos para atingir um objectivo (p.e. realizar um *Use Case*)
- Diagramas de sequência
  - foco no ordenamento temporal das trocas de mensagens
- Diagramas de comunicação
  - foco na arquitectura
- Diagramas de Temporização (*Timing Diagrams*)
  - foco nos aspectos temporais
- Diagramas de *Interaction Overview*
  - visão de alto nível que combina os anteriores



# Diagramas de Sequência - notação essencial

- representam as interacções entre objectos através das mensagens que são trocadas entre eles
- a ênfase é colocada na ordenação temporal das mensagens
- permitem analisar a distribuição de “responsabilidade” pelas diferentes entidades (analisar onde está a ser efectuado o processamento)

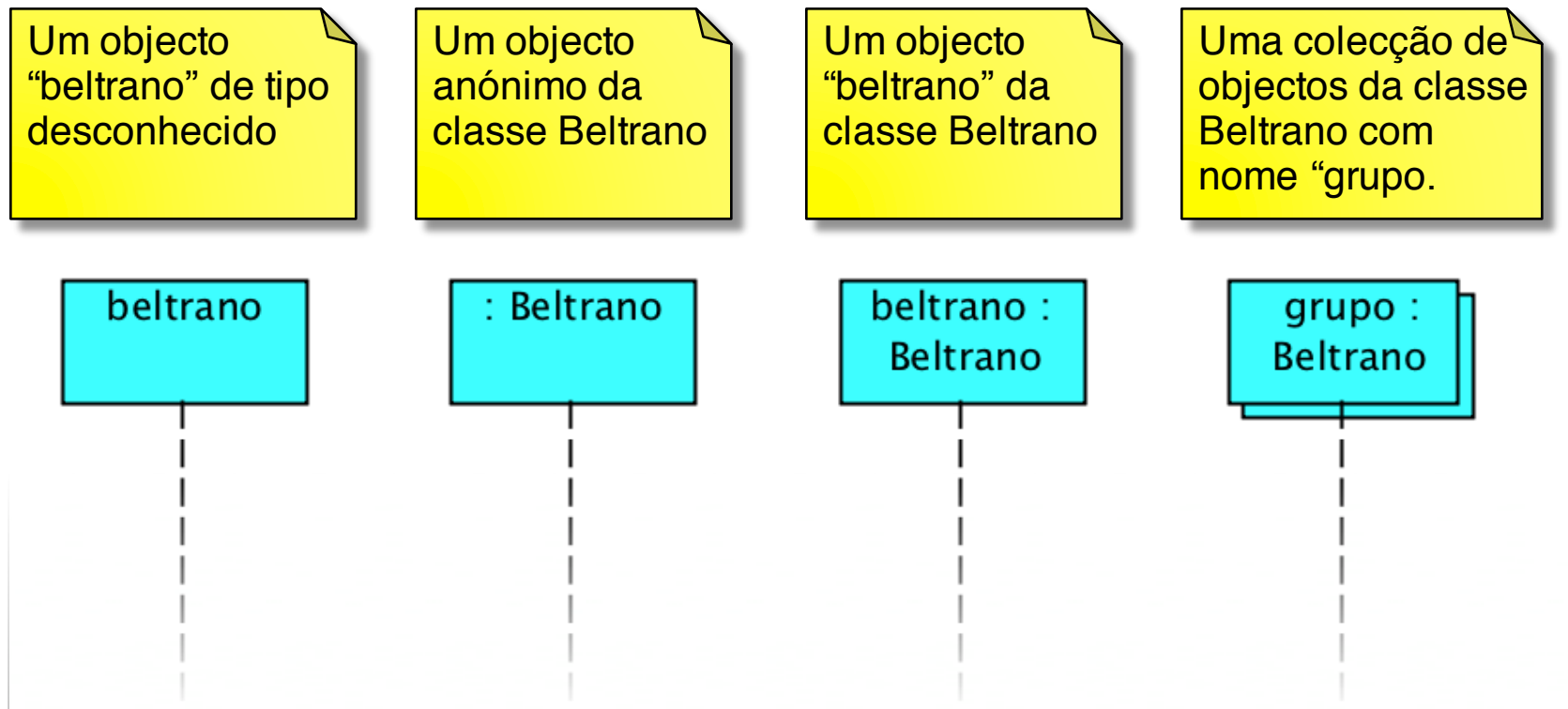






# Diagramas de Sequência - notação essencial

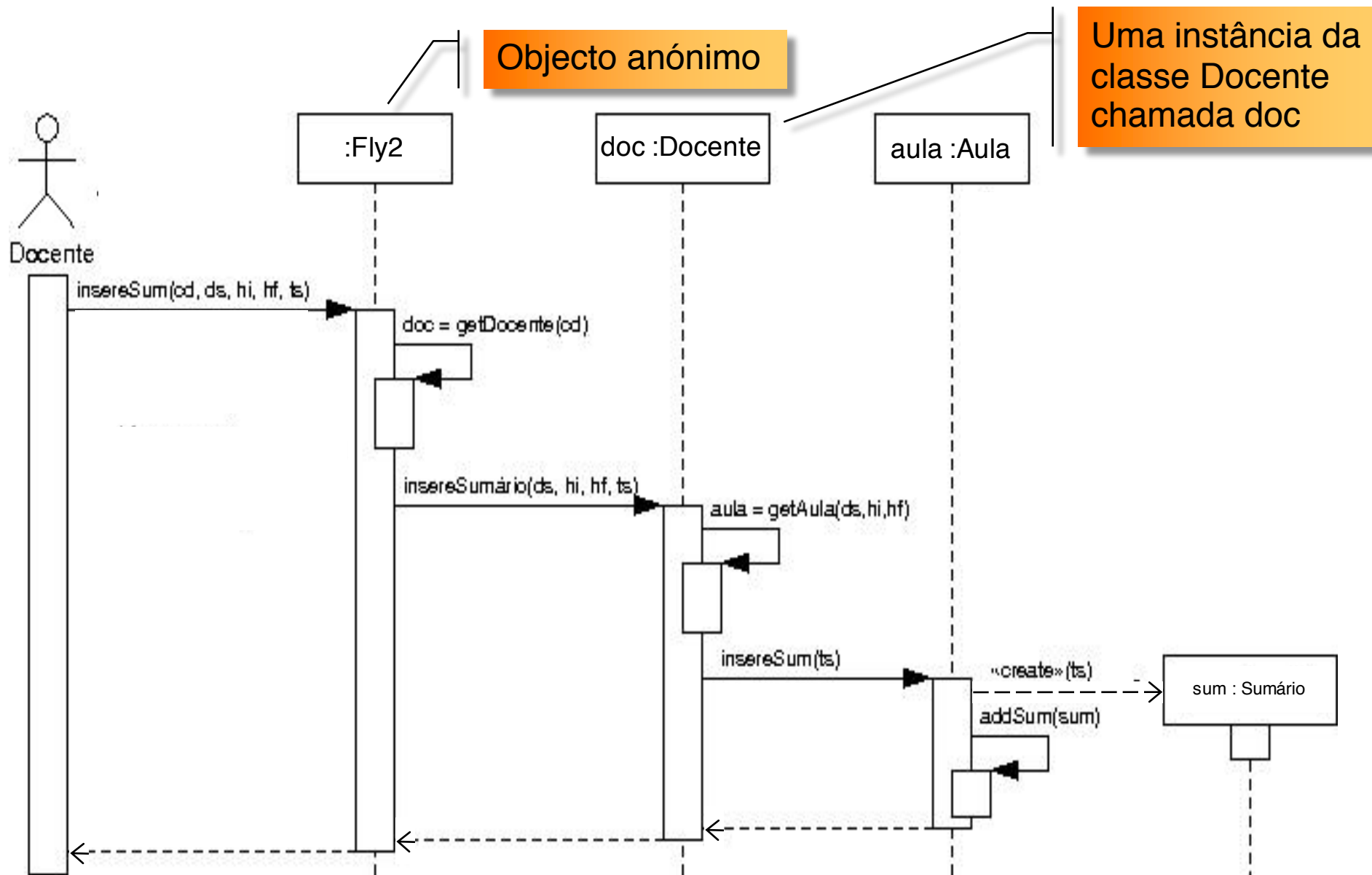
## Objectos



nome\_objecto “[“selector”]” : classe



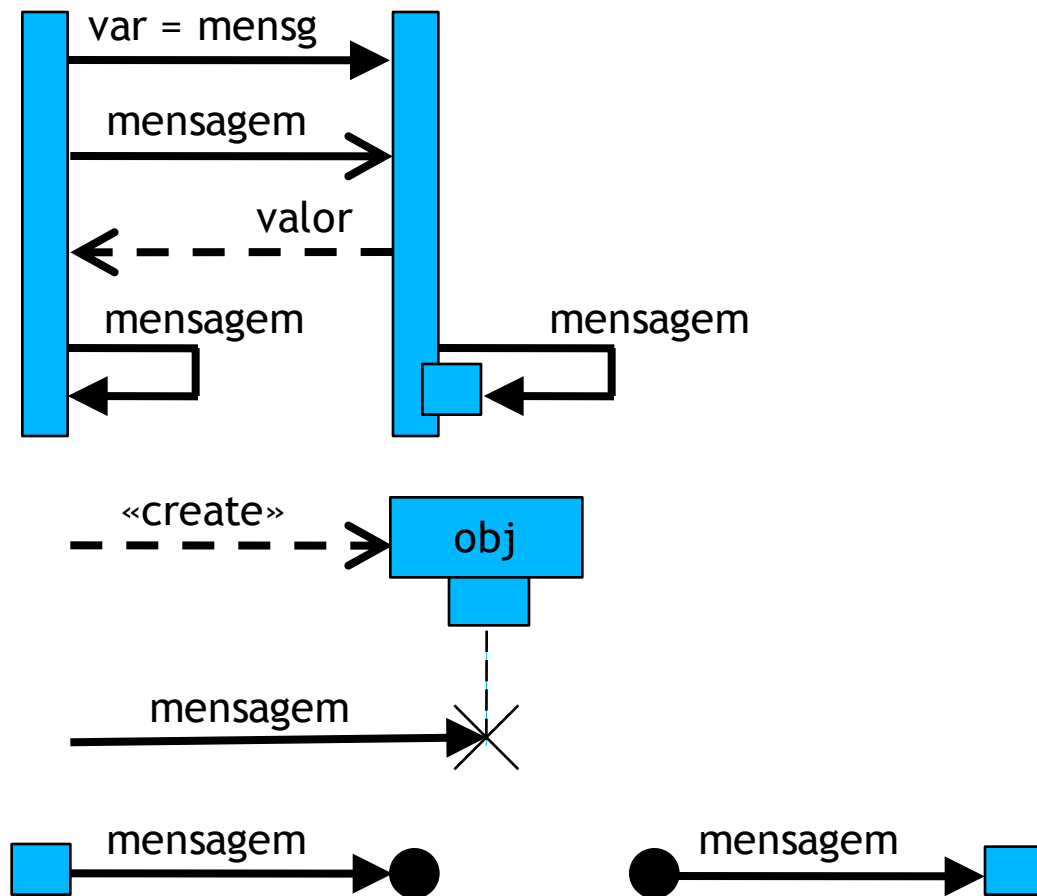
# Diagramas de Sequência - notação essencial





# Mensagens

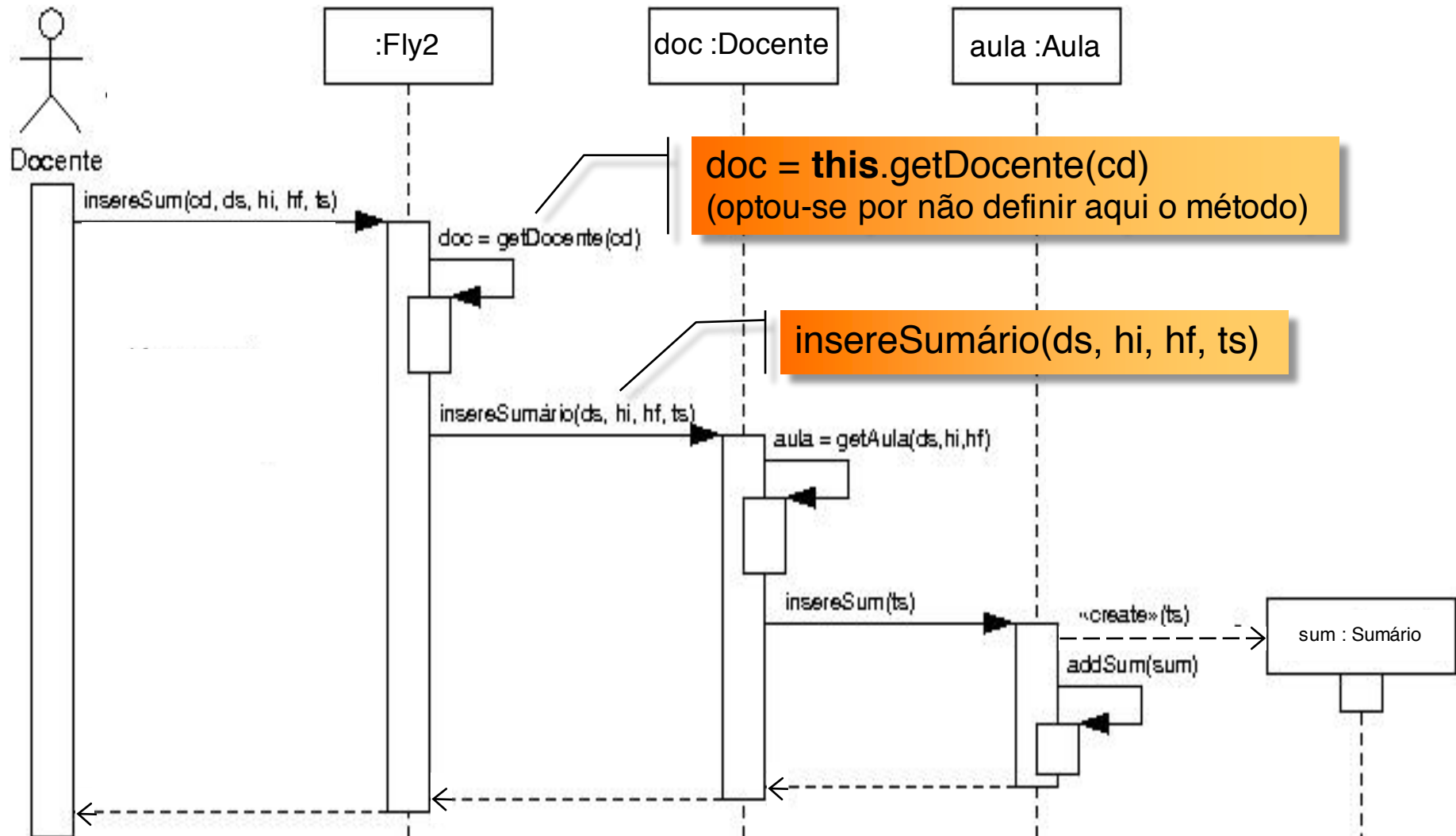
- invocação síncrona
- invocação assíncrona
- return/resultado
- self messages
- criar objectos
- destruir objectos
- lost/found messages



[ atributo '=' ] nome\_da\_operação\_sinal [ argumentos ] [ ':' tipo\_resultado ]

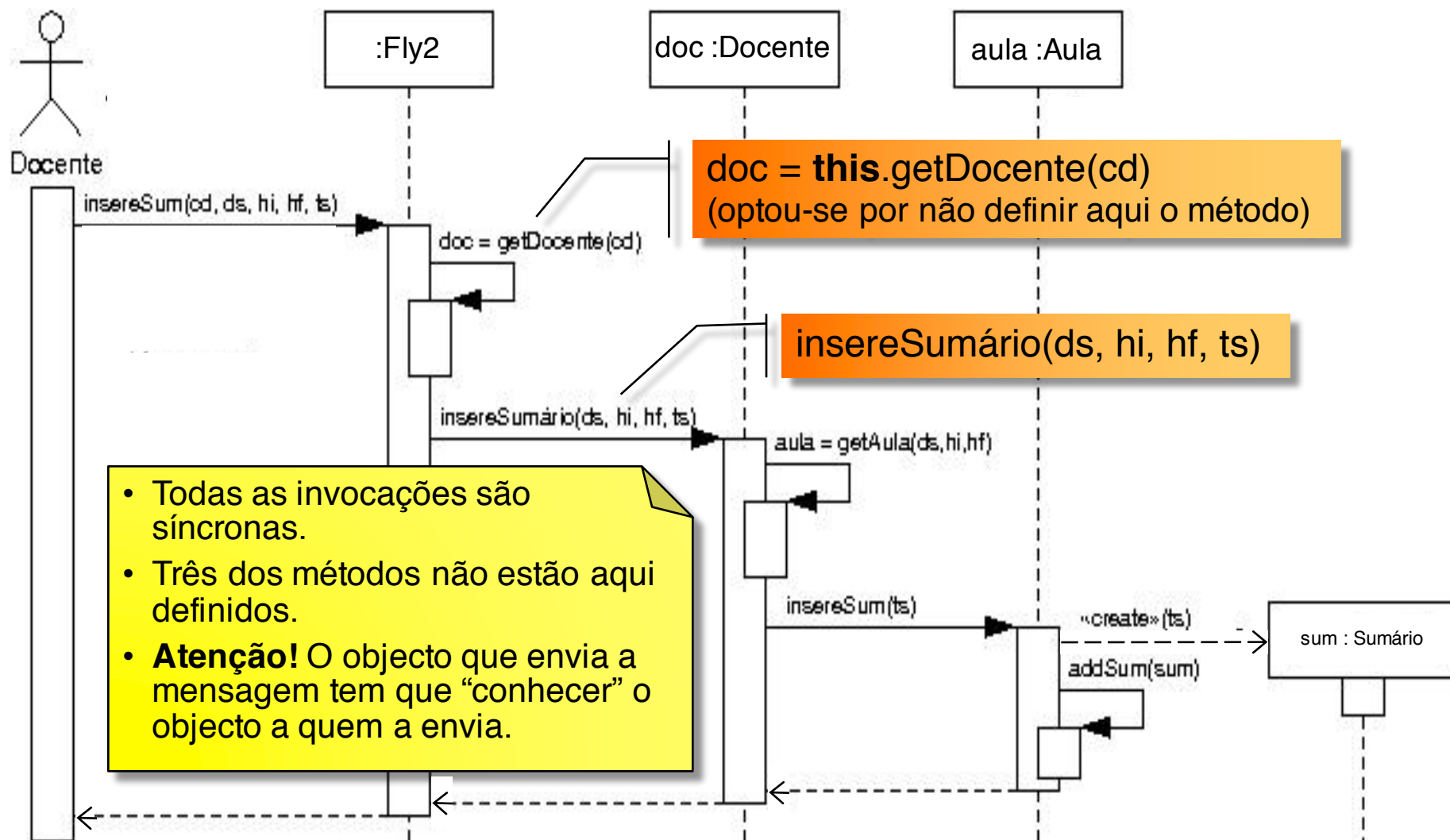


# Diagramas de Sequência - notação essencial





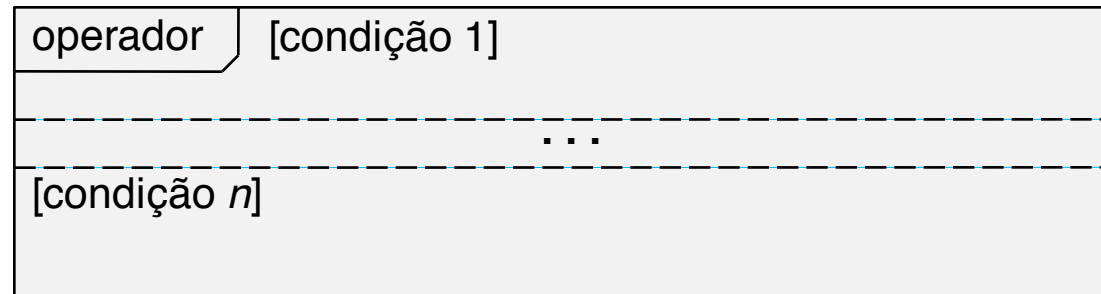
# Diagramas de Sequência - notação essencial





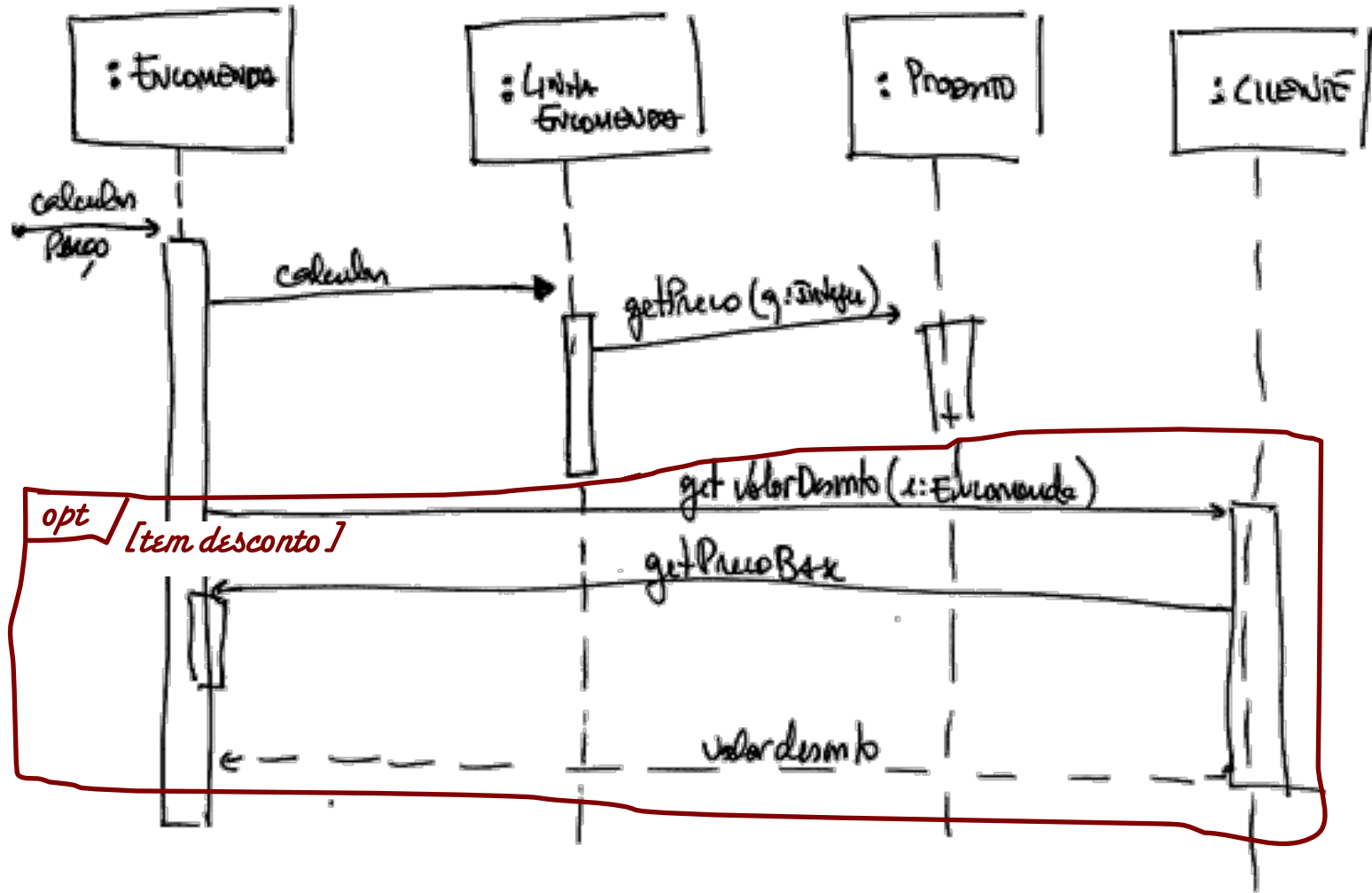
# Diagramas de Sequência - fragmentos combinados

- Um fragmento combinado agrupa conjuntos de mensagens
- Permitem expressar fluxos condicionais e estruturar os modelos



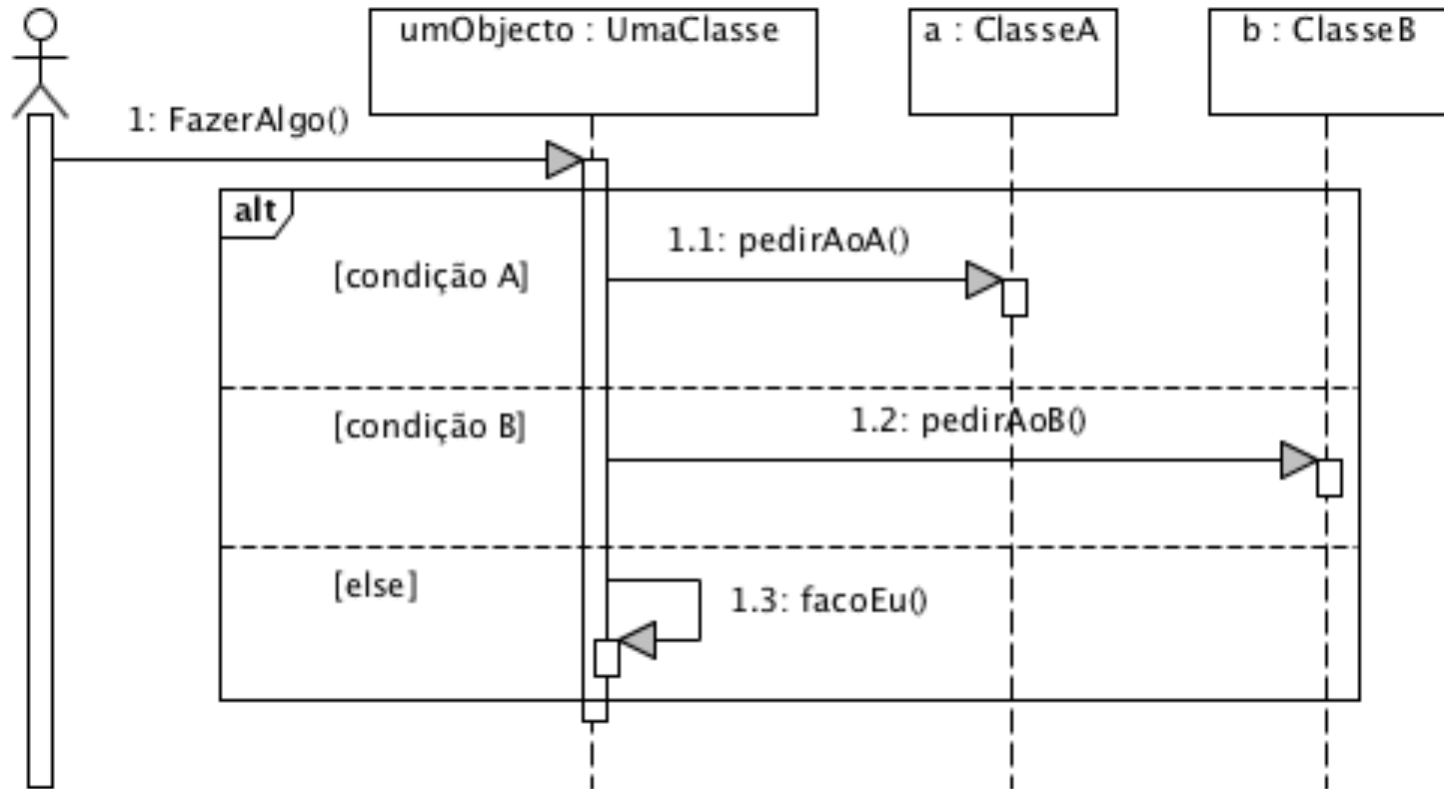
- Operadores mais comuns
  - **alt** - define fragmentos alternativos (mutuamente exclusivos)
  - **loop** / **loop(*n*)** - fragmento é repetido enquanto a guarda for verdadeira / *n* vezes
  - **opt** - fragmento opcional (ocorre se a guarda for verdadeira)
  - **par** - fragmentos ocorrem em paralelo
  - **break** - termina o fluxo
  - **ref** - referência a outro diagrama

# Operador *opt*



Cálculo do desconto só é efectuado se a guarda **tem desconto** se verificar.

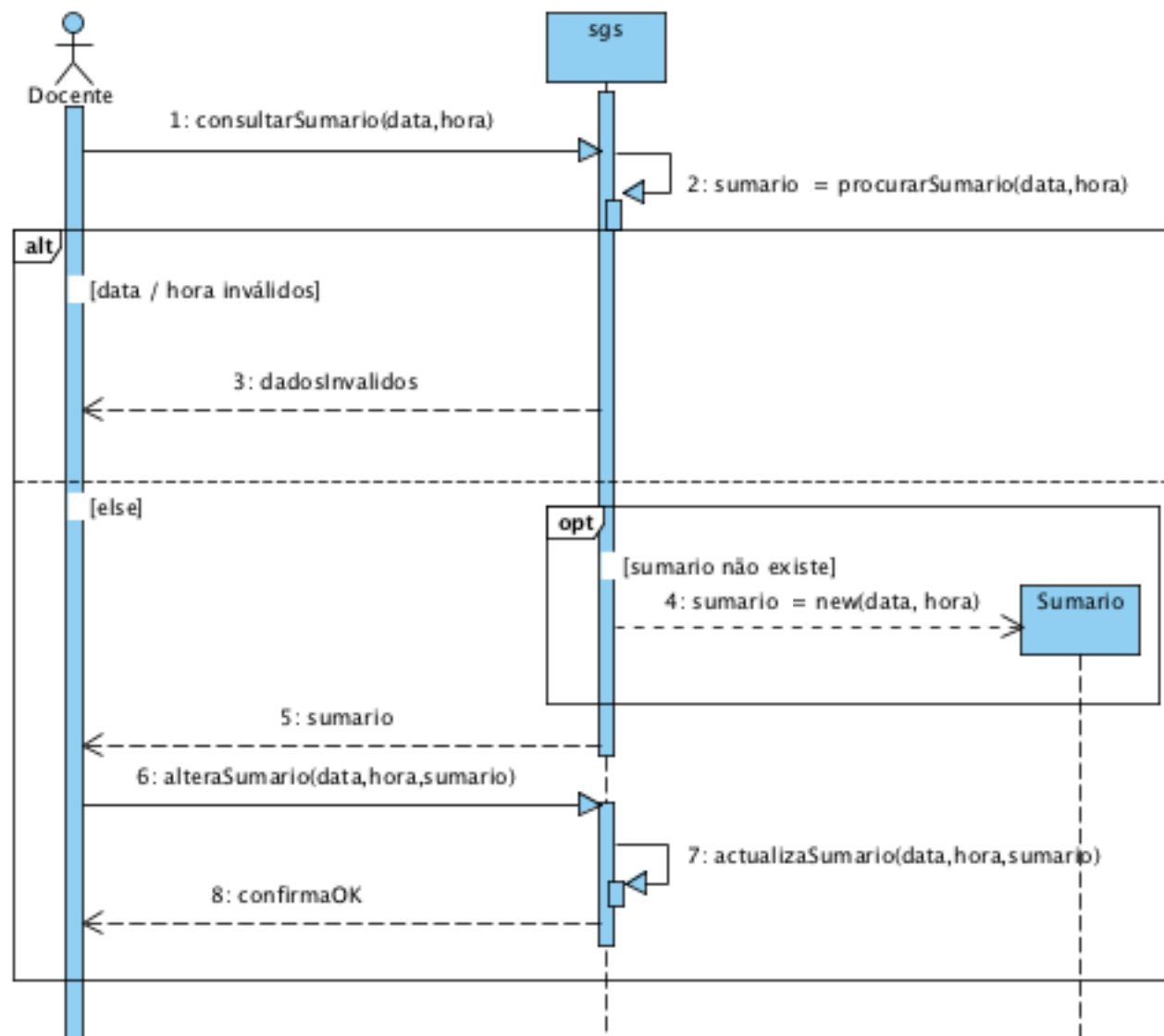
# Operador *alt*



- Os fluxos possíveis são mutuamente exclusivos, pelo que apenas um deles será seguido.
- Se mais que uma condição se verificar, não está definido qual acontece.



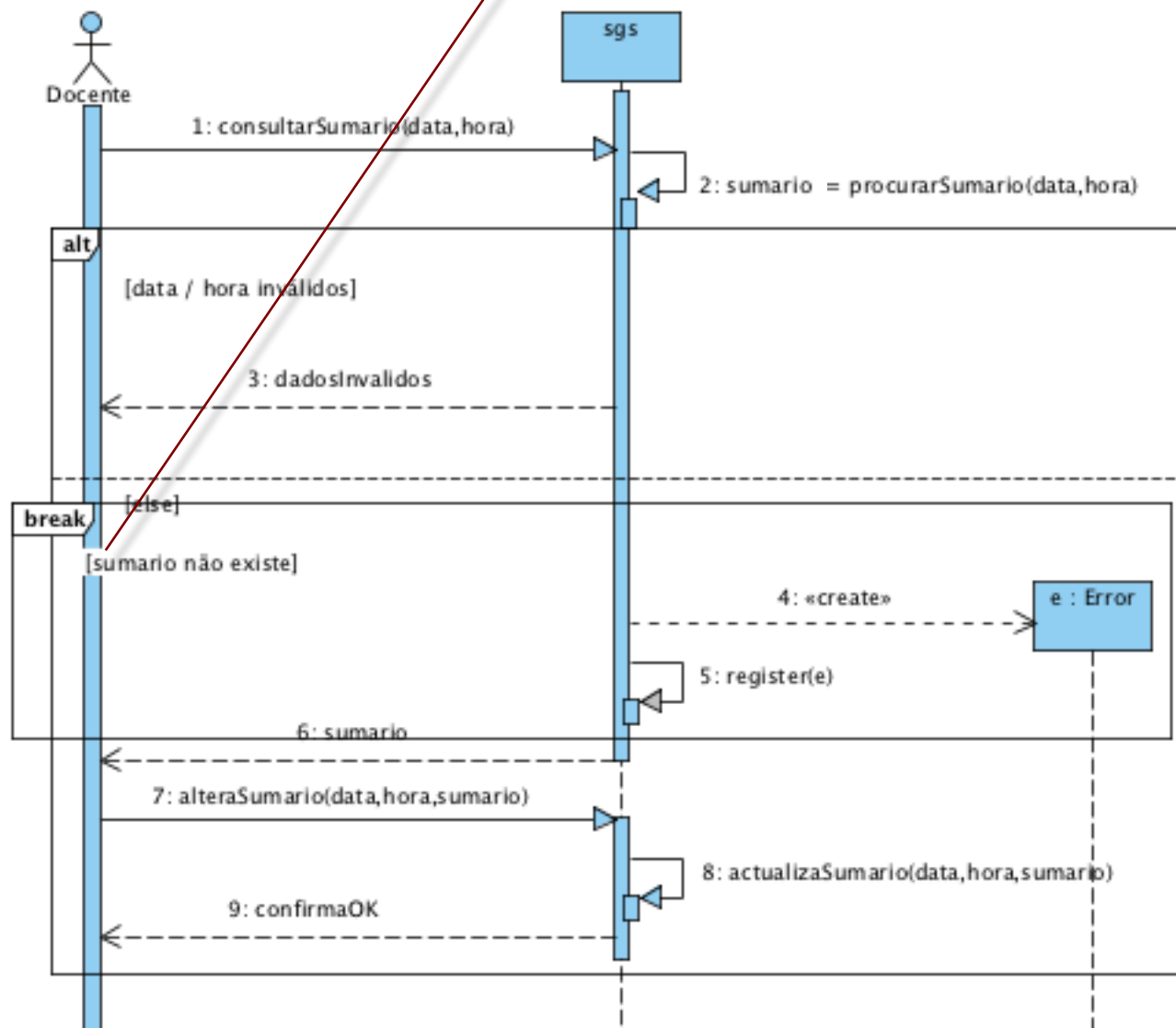
# Um exemplo...



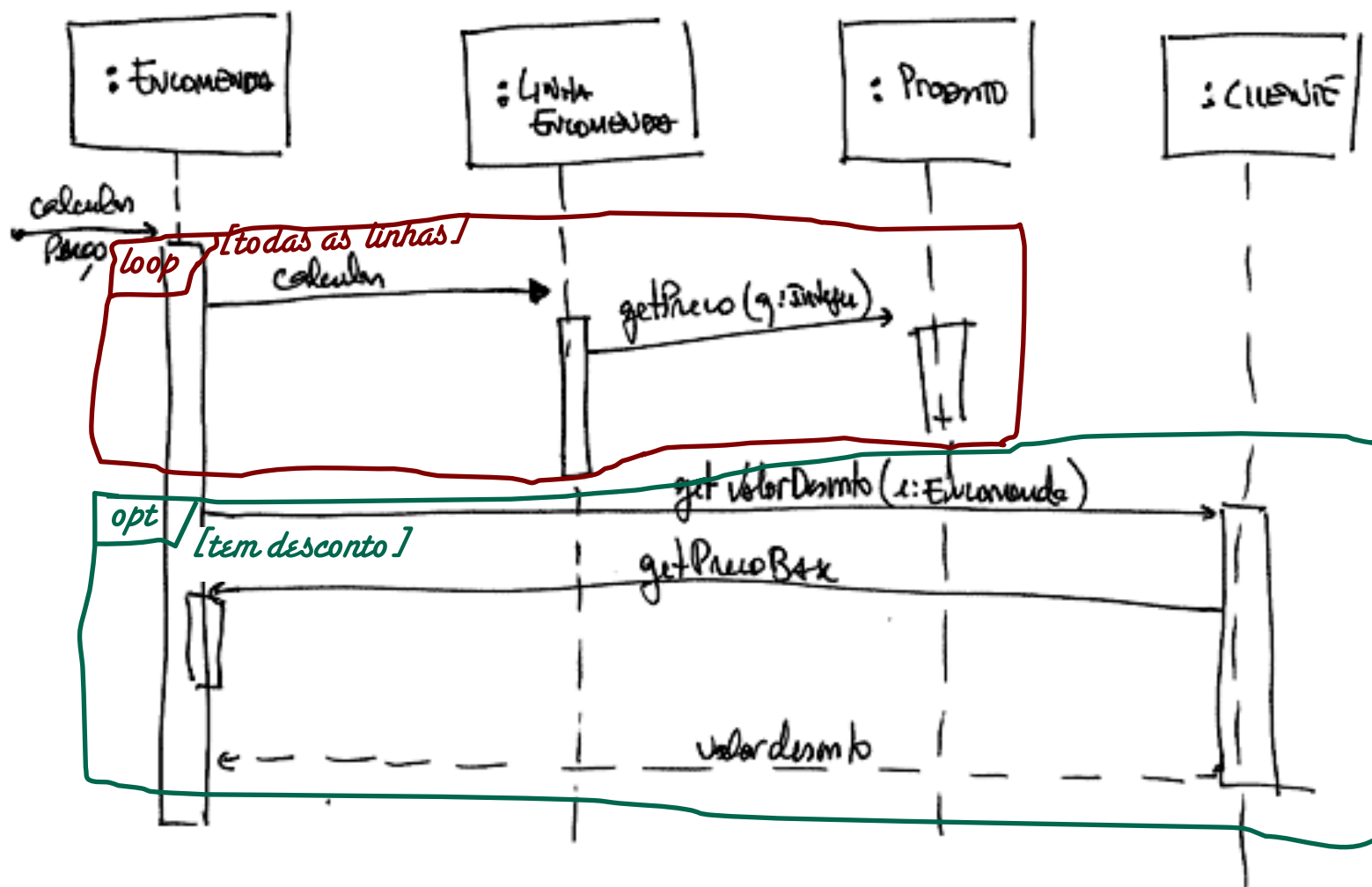


# Operador *break*

Se não existe, regista erro e termina.



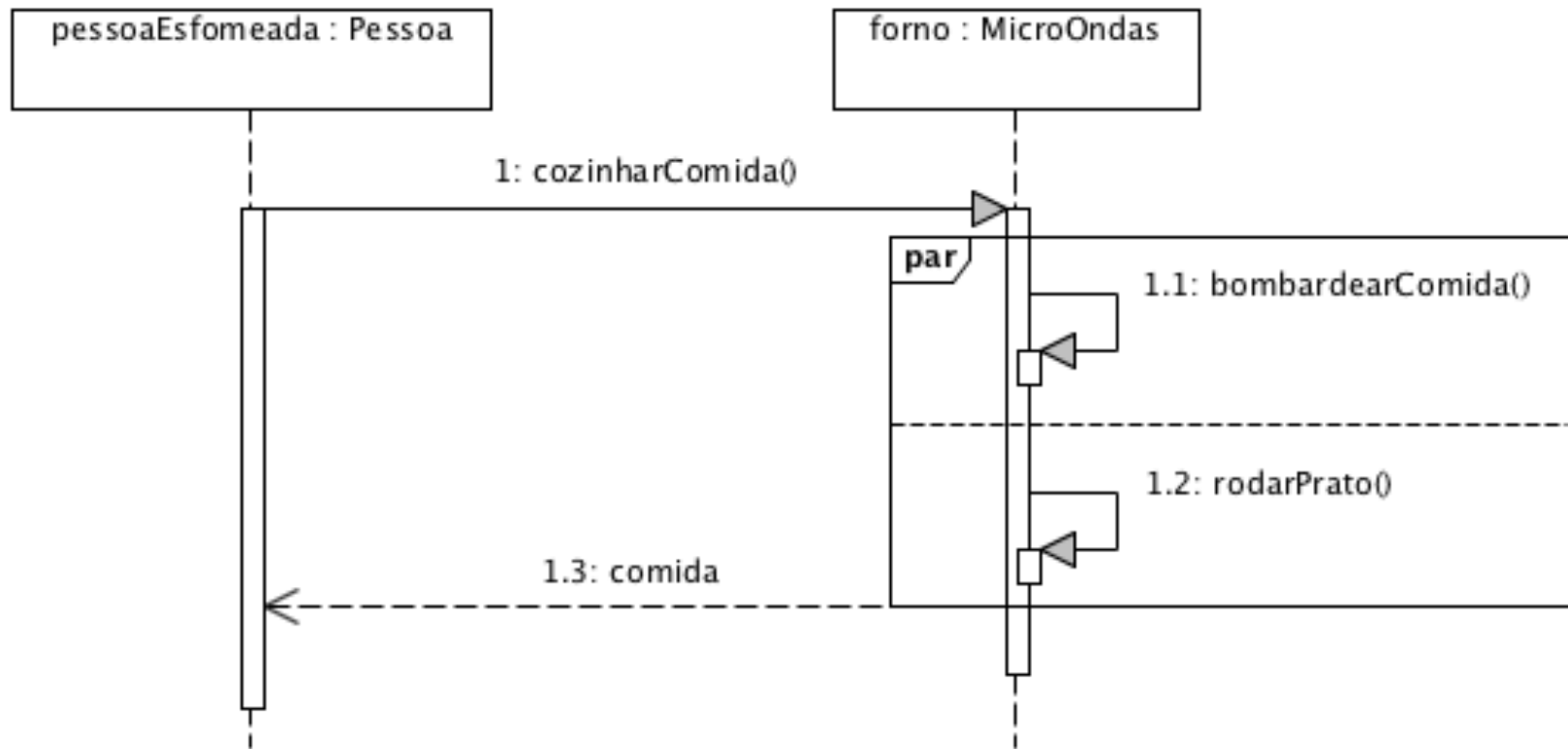
# Operador *loop*



Cálculo do preço é efectuado para todas as linhas da encomenda.

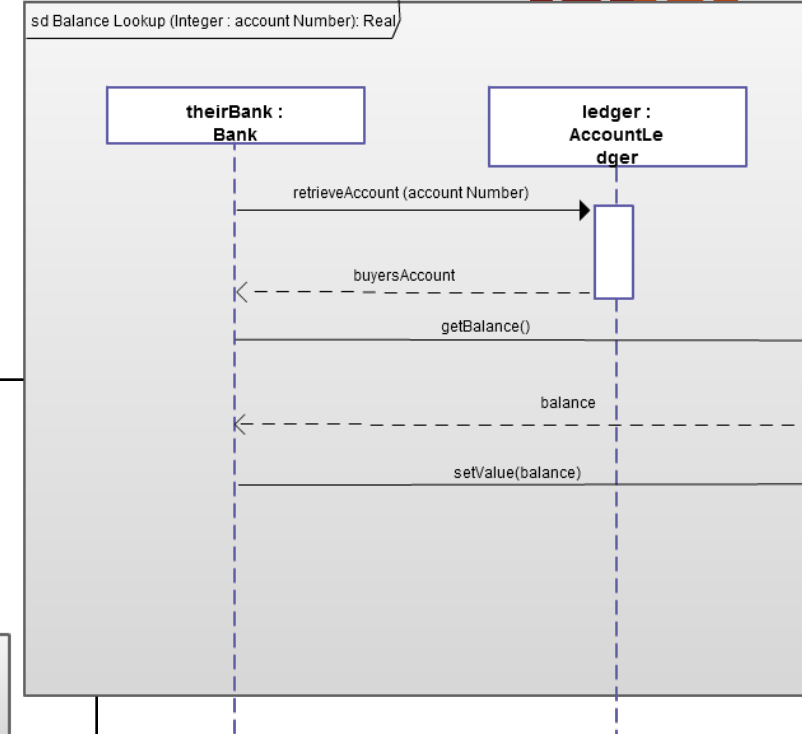
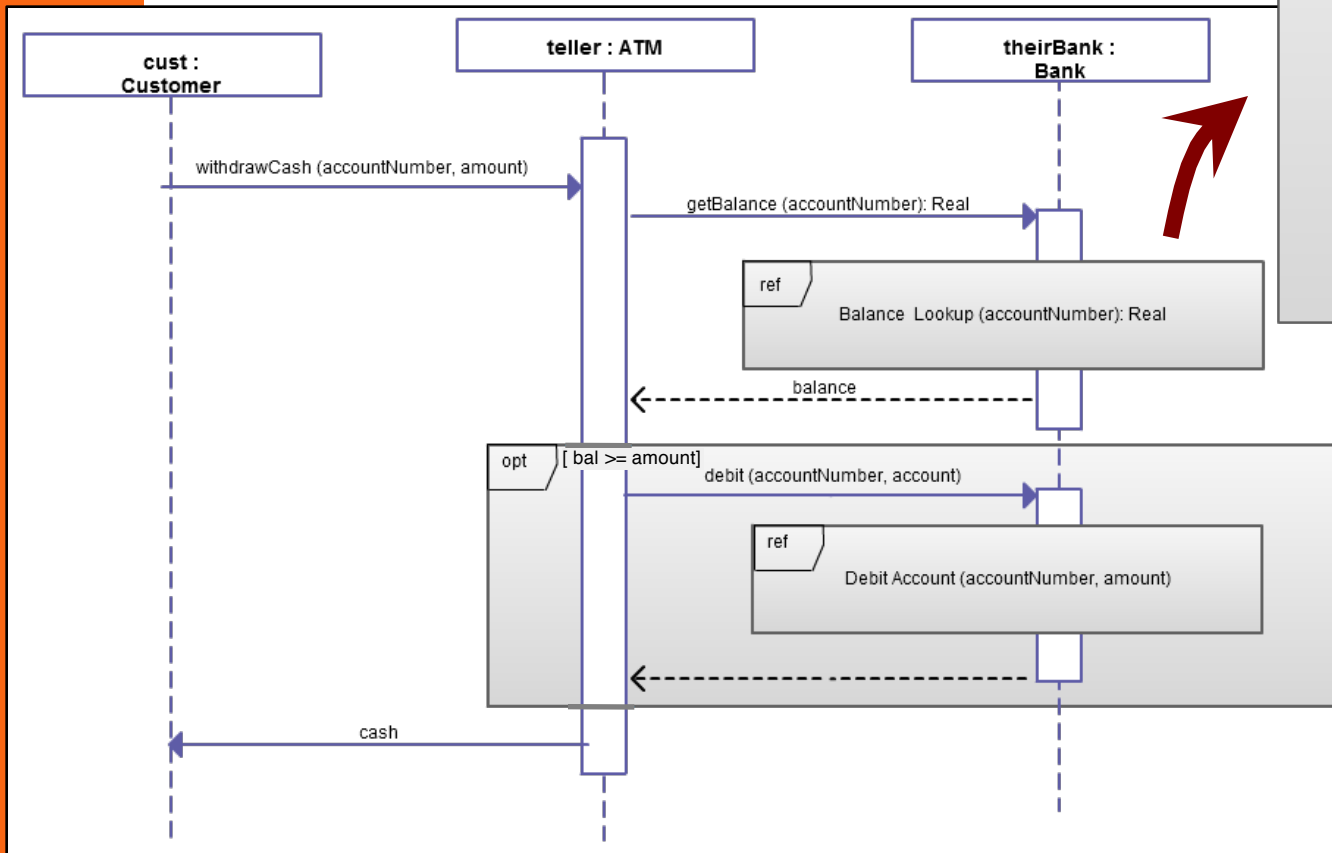


## Operador *par*



Uma pessoa esfomeada envia a um micro-ondas uma mensagem para cozinhar uma refeição. O micro-ondas envia a si próprio duas mensagens, uma para “bombardear” e outra para “rodar” a comida, tarefas que são realizadas em paralelo. Quando ambas estiverem concluídas, a esfomeada pessoa recebe como resultado comida

# Operador *ref*



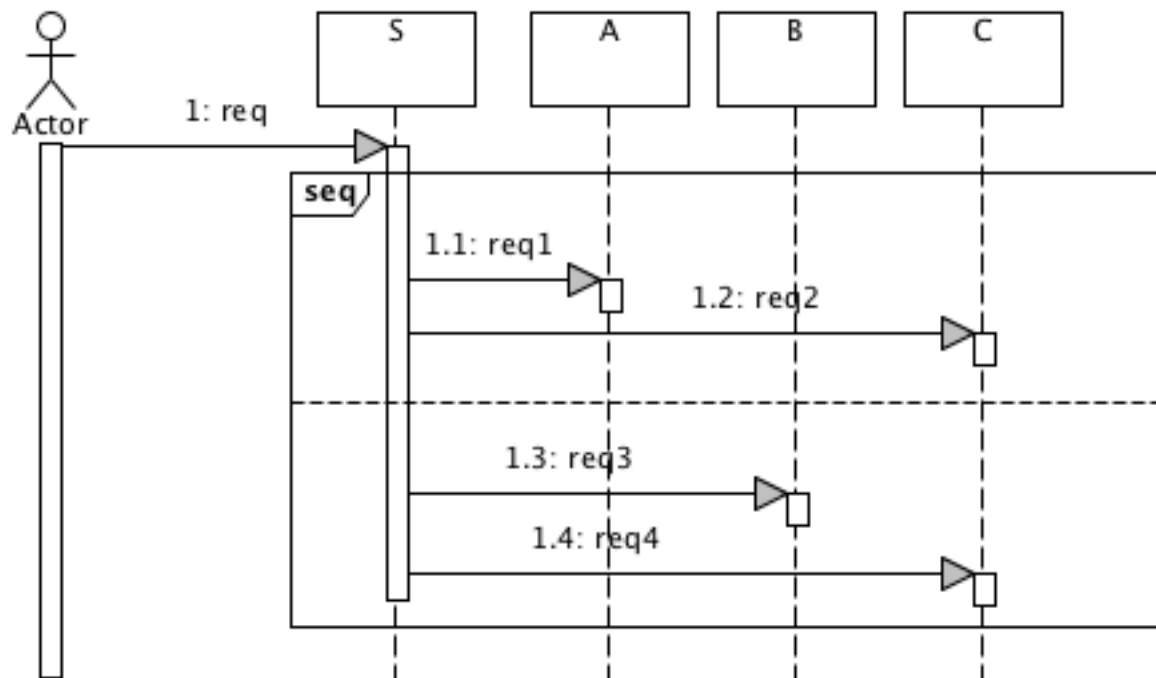
- Um SD pode reutilizar outros SD referenciando-os num fragmento com o operador *ref* – permite estruturar os modelos



## Outros operadores

- **critical** - o operando executa de forma atômica
- **seq** (sequenciação fraca) - todos os operandos executam em paralelo, mas eventos enviados a uma mesma linha de vida acontecem na mesma sequência dos operandos
- **strict** - os operandos executam em sequência
- **neg** - negação, o operando mostra uma interacção inválida
- **assert** - mostra o único comportamento válido naquele ponto
- **ignore** - indica mensagens intencionalmente omitidas da interacção (ignore {m1, m2, ...})
- **consider** - indica mensagens intencionalmente incluídas na interacção (dual de ignore)

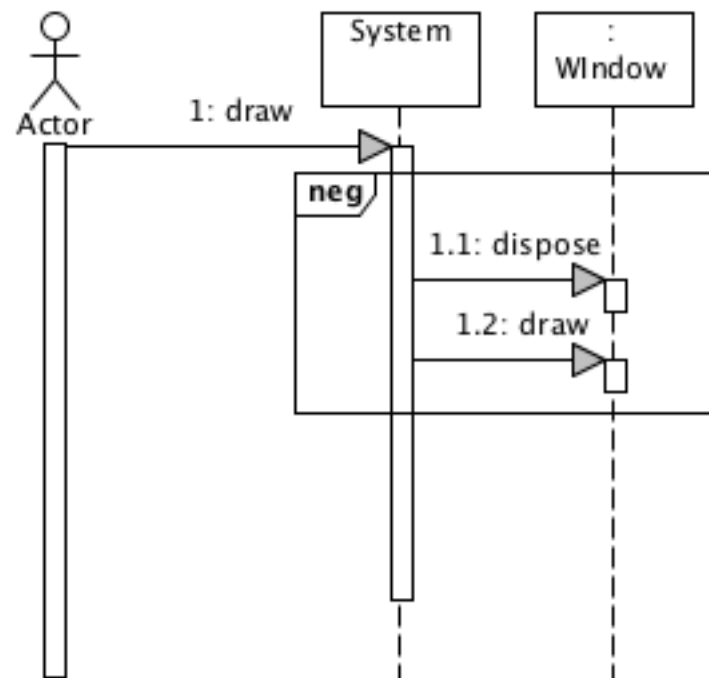
# Operador *seq*



**Eventos *req1* e *req3* podem acontecer em paralelo. Evento *req2* acontece antes de evento *req4* (porque ambos vão para C).**



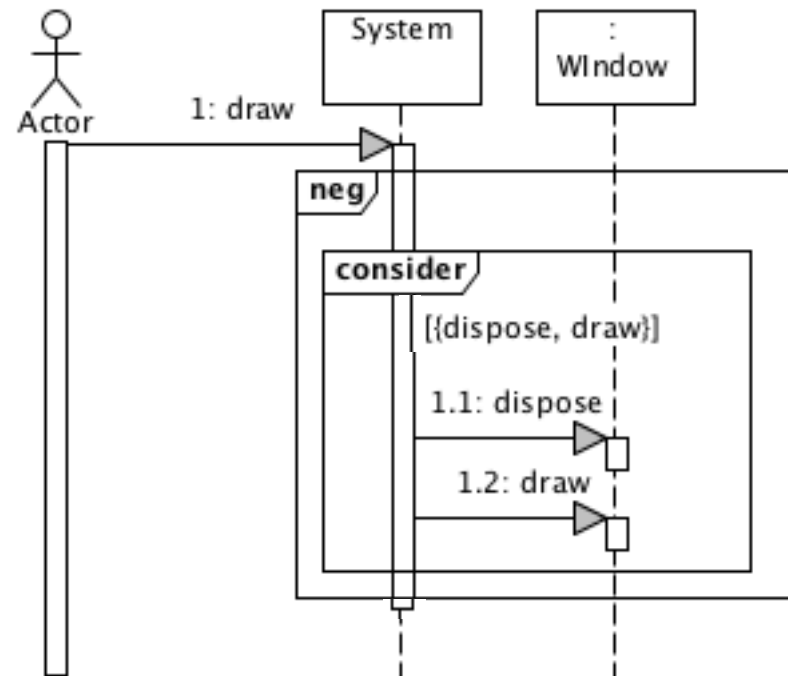
## Operadro *neg*



**Não é válido desenhar numa janela depois de ela ter sido removida.**



# Operador *consider*



Porque podem existir outros eventos pelo meio...



# Diagramas de Sequência

## Sumário

- Necessidade de modelação comportamental
- Diagramas de Sequência
  - Enquadramento
  - Notação base
  - Notação para representação de objectos
  - Notação para representação de mensagens
  - Fragmentos e operadores