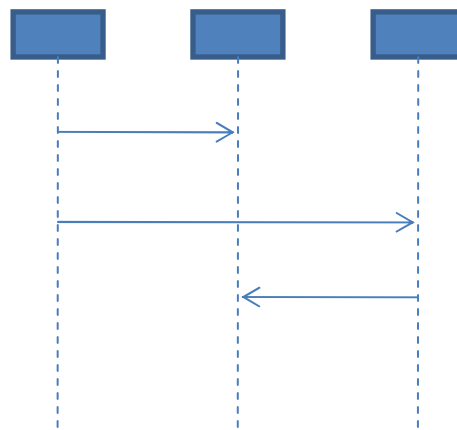


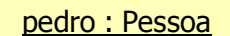
O que é?

- Diagrama criado para modelagem da interação entre objetos
 - Detalha como objetos colaboram para implementar um cenário de caso de uso
 - Útil para ajudar na identificação dos métodos das classes
- Caixas representando objetos
- Linhas verticais representando a vida do objeto
- Linhas horizontais representando troca de mensagens



Objetos

- Os objetos são de algum tipo definido no diagrama de classes
 - O nome de um objeto é da forma *nome : classe*
- Em situações onde um nome específico não pode ser identificado (ex.: pedro : Pessoa), utilize:
 - Um nome genérico (ex.: umaPessoa : Pessoa)
 - Um nome único (ex.: aPessoa : Pessoa)
 - Ou omita o nome (ex.: : Pessoa)
- Uma linha pontilhada sai do objeto (linha de vida) representando o momento da sua criação em diante
 - Quanto mais para baixo, mais tempo passou



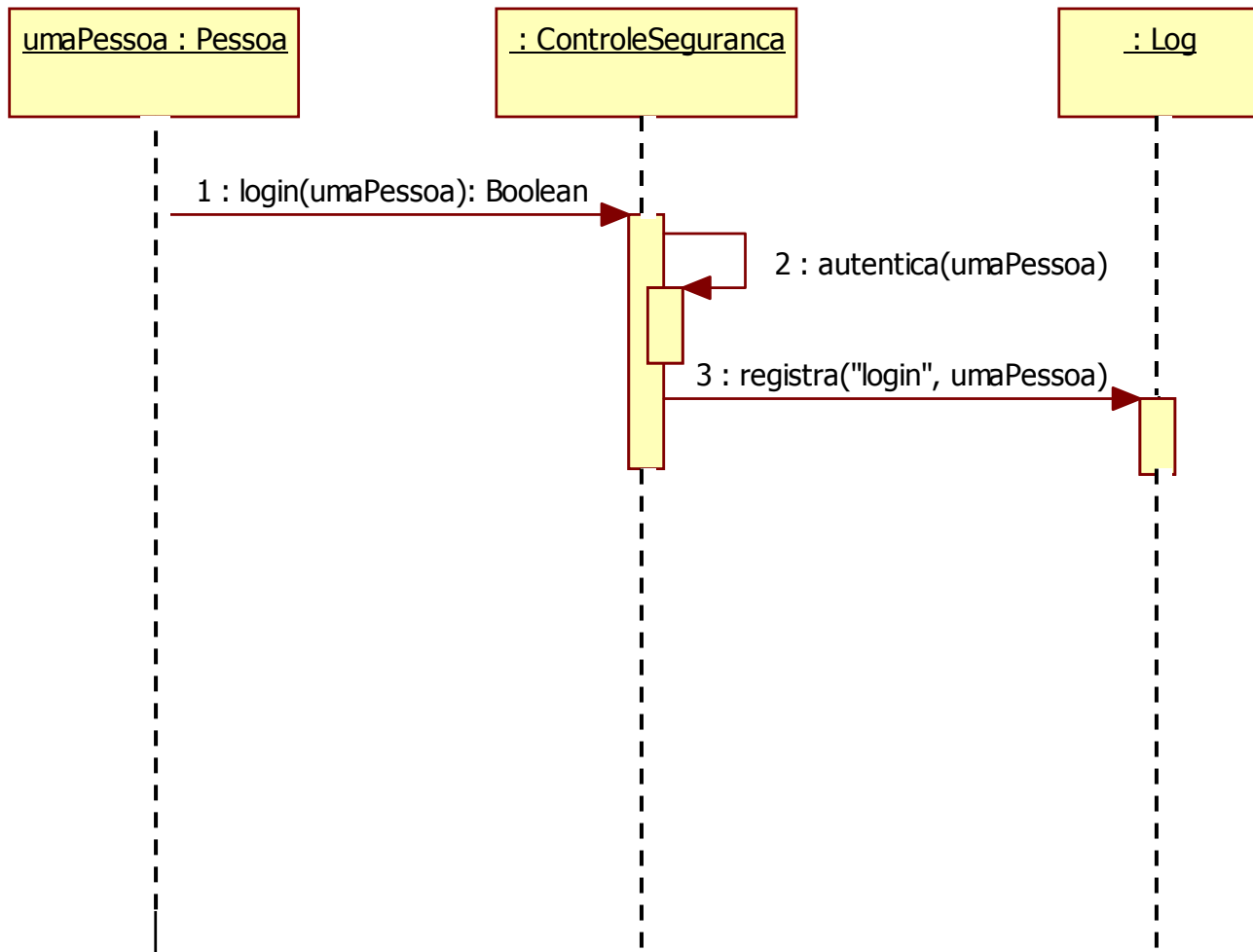
pedro : Pessoa

A diagram illustrating an object notation. It consists of a yellow rectangular box with a red border containing the text "pedro : Pessoa". A vertical dashed line extends downwards from the bottom of the box, representing the object's lifetime.

Mensagens

- A interação entre objetos é representada por mensagens
 - Para outros objetos
 - Para o mesmo objeto (auto-mensagem)
- Uma mensagem contém a assinatura do método que está sendo chamado
- Uma barra de ativação indica o escopo de execução do método

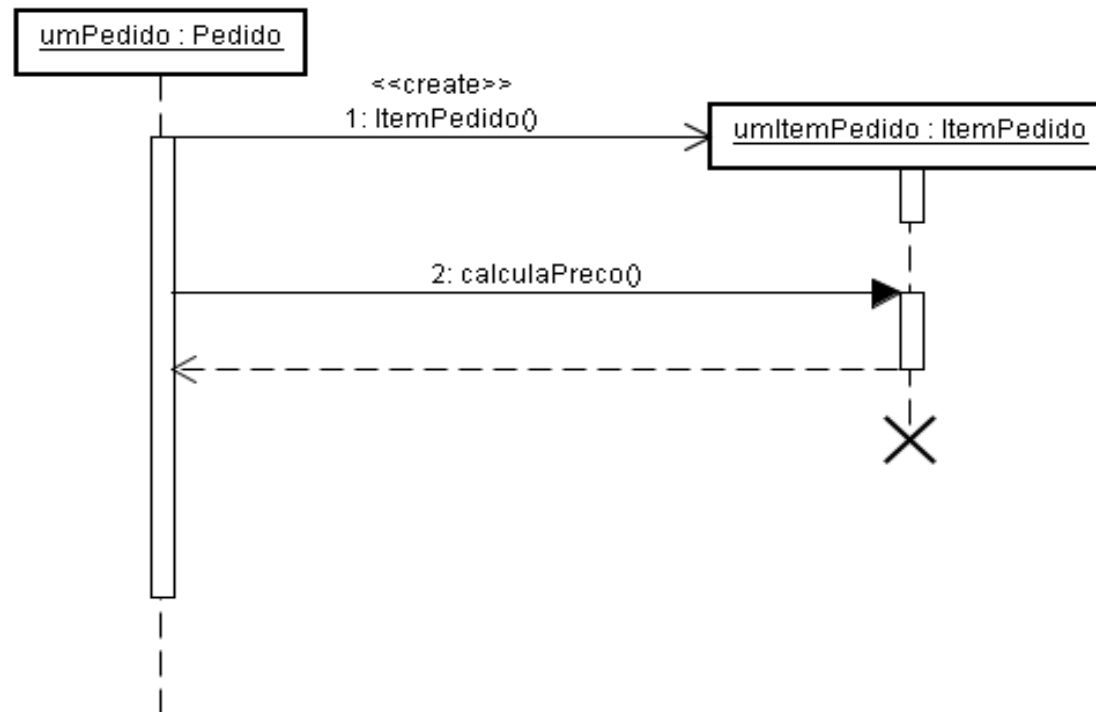
Mensagens



Mensagens

- Mensagem de criação
 - Aponta diretamente para o objeto e é marcada com <<create>>
- Mensagem de retorno
 - Opcional, e normalmente é omitida
 - Usa seta tracejada
- Marca de destruição
 - Indica o término da vida de um objeto com um “X”

Mensagens



Mas como representar um algoritmo mais complexo?

➤ Exemplo:

Para cada item de produto

 Se o valor do produto for maior que
 10000 então

 Despacha com cuidado

 Caso contrário

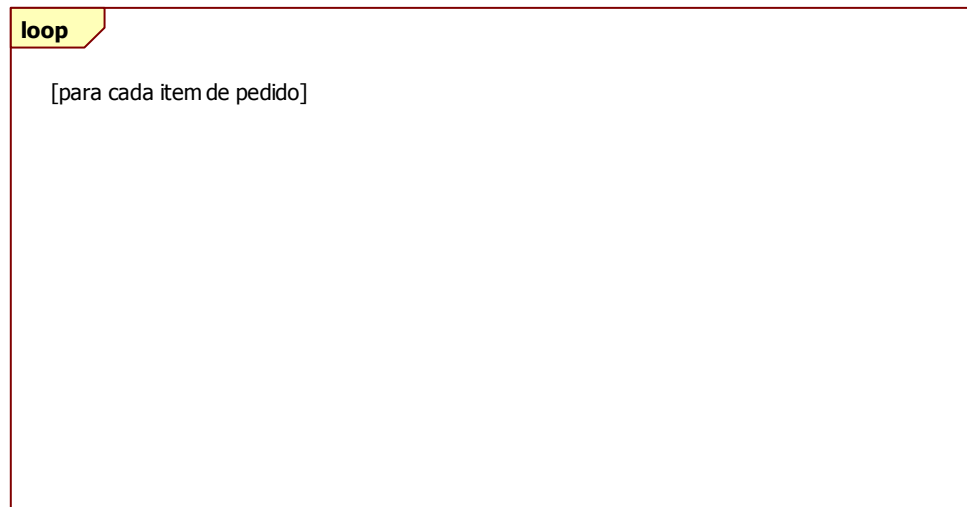
 Despacha normalmente

Se precisa de confirmação

 Envia confirmação

Repetições

- O diagrama de seqüência permite que repetições sejam feitas durante o fluxo
- Para isso são utilizados quadros (*frames*) do tipo *loop*



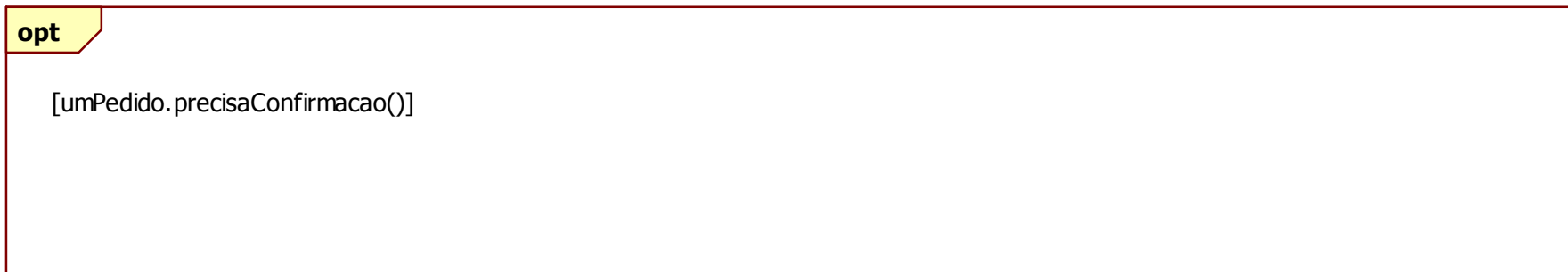
Decisões

- O diagrama de seqüência permite que decisões sejam tomadas durante o fluxo
- Para isso são utilizados quadros (*frames*) do tipo *alt* ou *opt* com condições de guarda

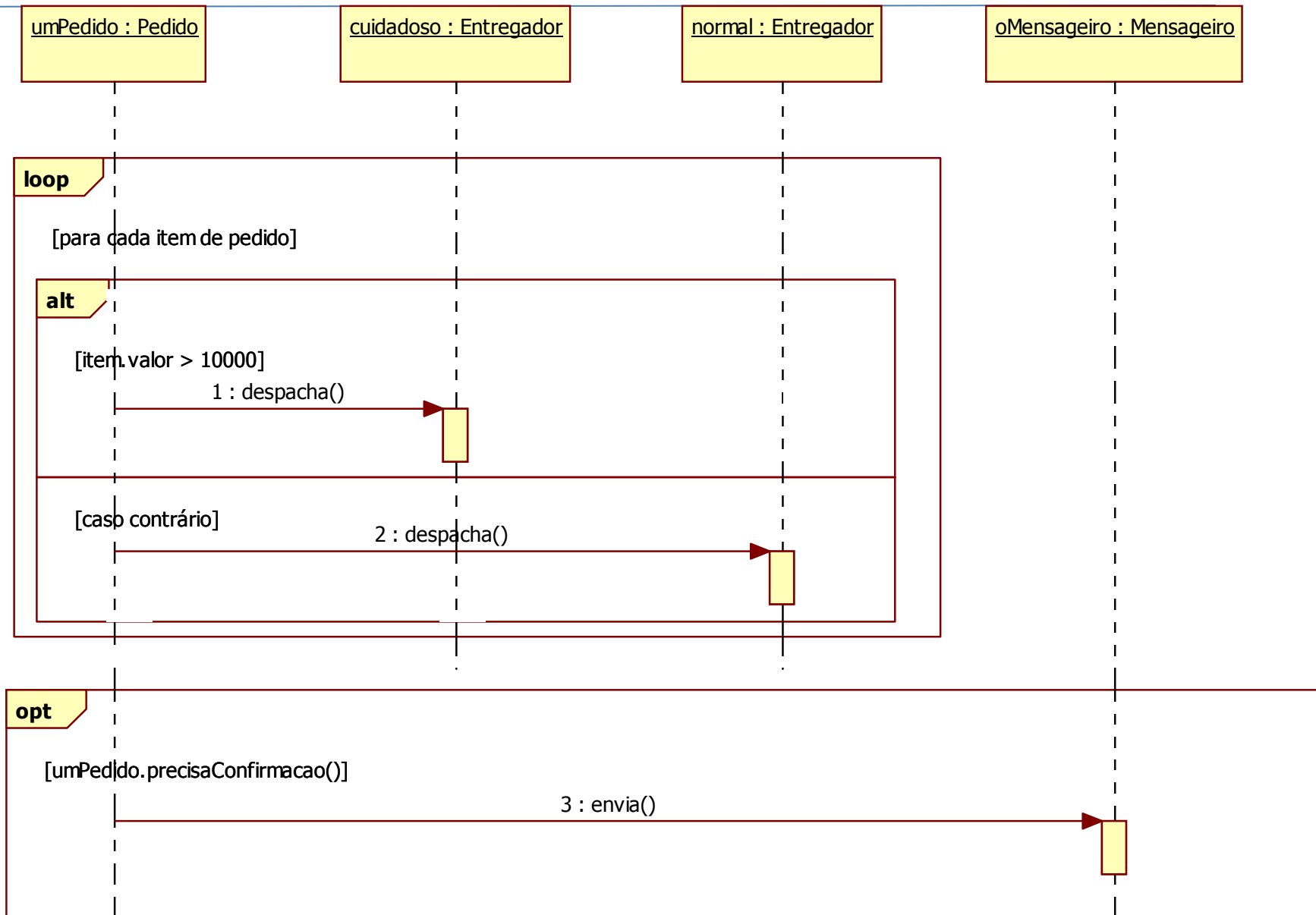
If + else



if



Exemplo

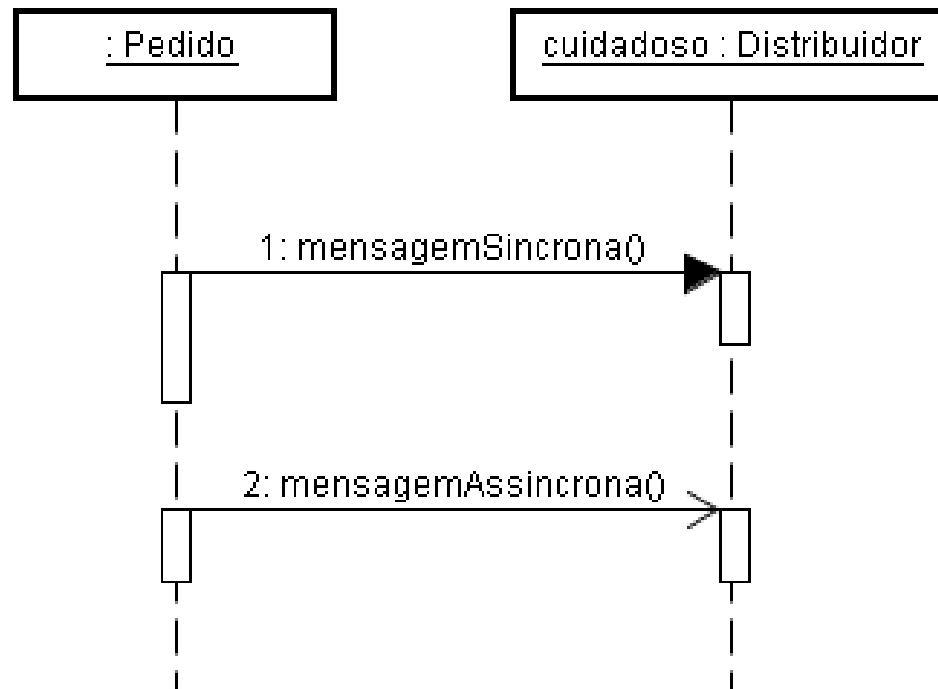


Outros quadros disponíveis

- Além dos quadros do tipo *loop*, *opt* e *alt*, existem outros tipos, entre eles:
 - *par*: Contém vários seguimentos e todos são executados em paralelo
 - *region*: Determina uma região crítica, que deve ter somente uma *thread* em execução em um dado momento

Chamada síncrona x assíncrona

- É possível utilizar dois tipos de chamada de métodos no diagrama de seqüência:
 - Chamada síncrona (seta cheia): a execução fica bloqueada até o retorno do método
 - Chamada assíncrona (seta vazia): a execução continua em paralelo ao método que foi chamado (*fork* implícito)



Quando utilizar diagrama de seqüência?

- Para representar em alto nível a interação entre diferentes objetos visando atender a um caso de uso
- Para ajudar a encontrar os métodos do diagrama de classes
- Cuidado: não use diagrama de seqüência...
 - Para métodos muito simples (ex.: get e set)
 - Para definição precisa de como será o código

Exercício

- Elabore um diagrama de seqüência para o algoritmo *Quicksort* (versão ingênua)
 - Primeiro elemento da lista de entrada é o pivô
 - Cria outras duas listas com os elementos menores e maiores que o pivô
 - Ordena recursivamente as outras duas listas
 - Concatena a lista de menores ordenada, o pivô e a lista de maiores ordenada, criando a lista de saída ordenada

Bibliografia

- Fowler, Martin. 2003. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. 3rd ed. Addison-Wesley Professional.
- Várias transparências foram produzidas por Leonardo Murta
 - <http://www.ic.uff.br/~leomurta>