

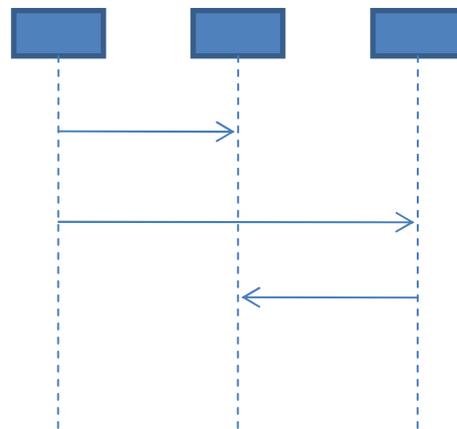
Diagrama de Seqüência

Viviane Torres da Silva
viviane.silva@ic.uff.br

<http://www.ic.uff.br/~viviane.silva/es1>

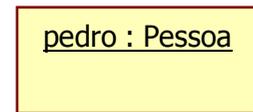
O que é?

- 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

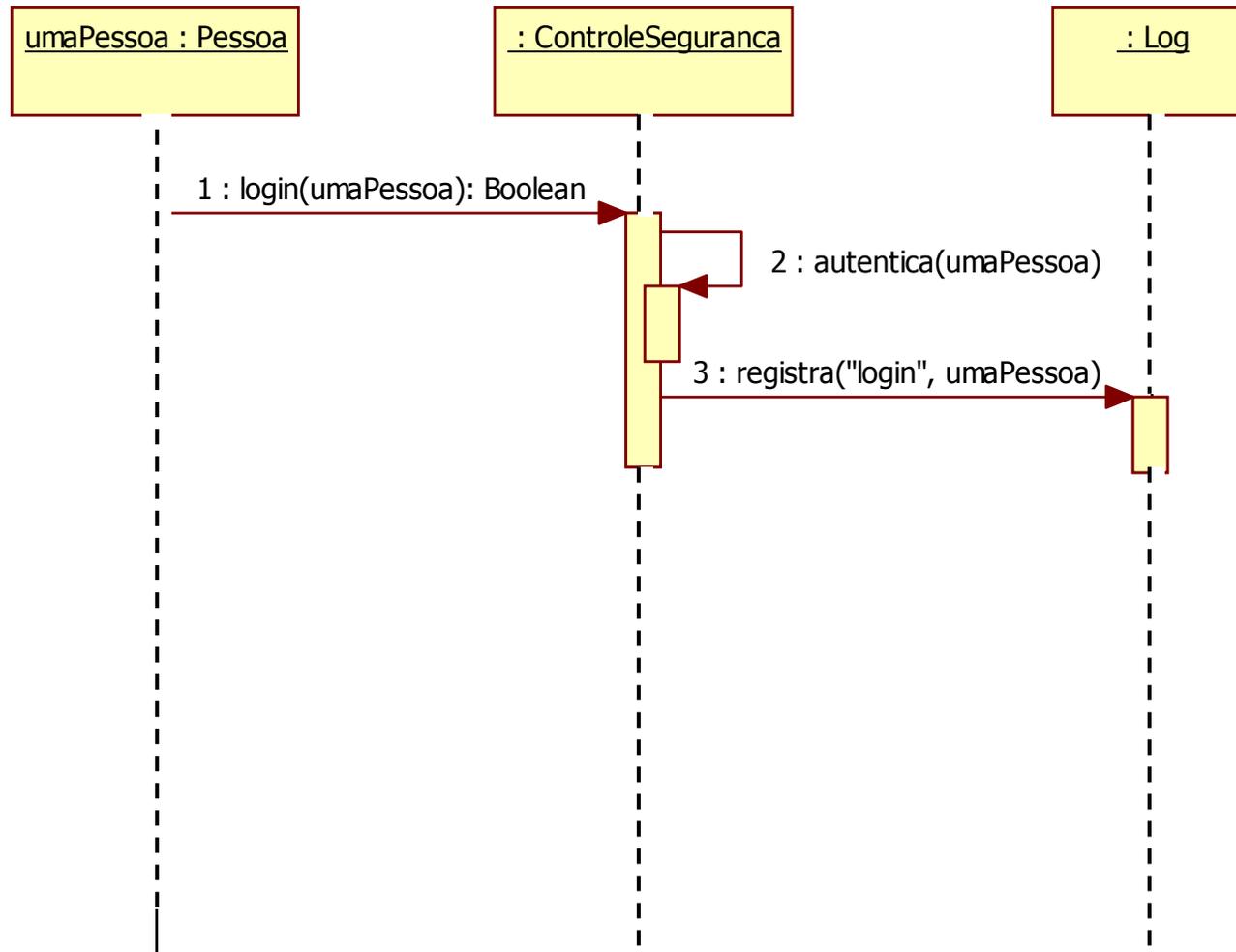
- Os objetos são de algum tipo definido no diagrama de classes
 - O nome de um objeto é da forma *nome : classe*
- Em situações onde um nome específico não pode ser identificado (ex.: pedro : Pessoa), utilize:
 - Um nome genérico (ex.: umaPessoa : Pessoa)
 - Um nome único (ex.: aPessoa : Pessoa)
 - Ou 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



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

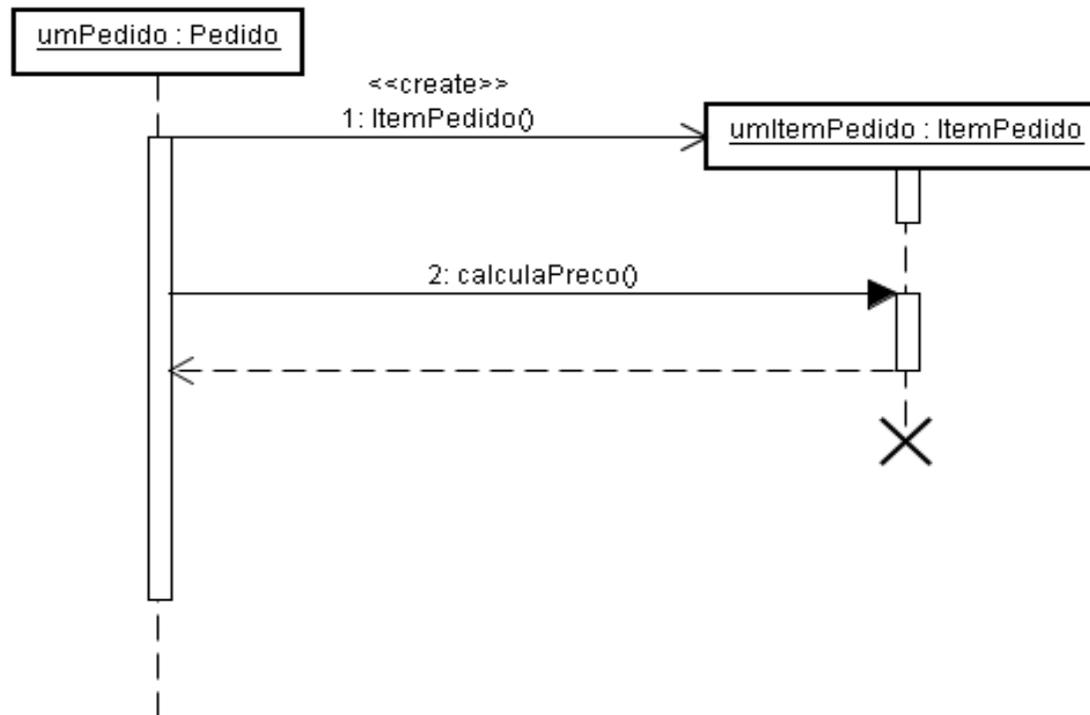
Mensagens



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

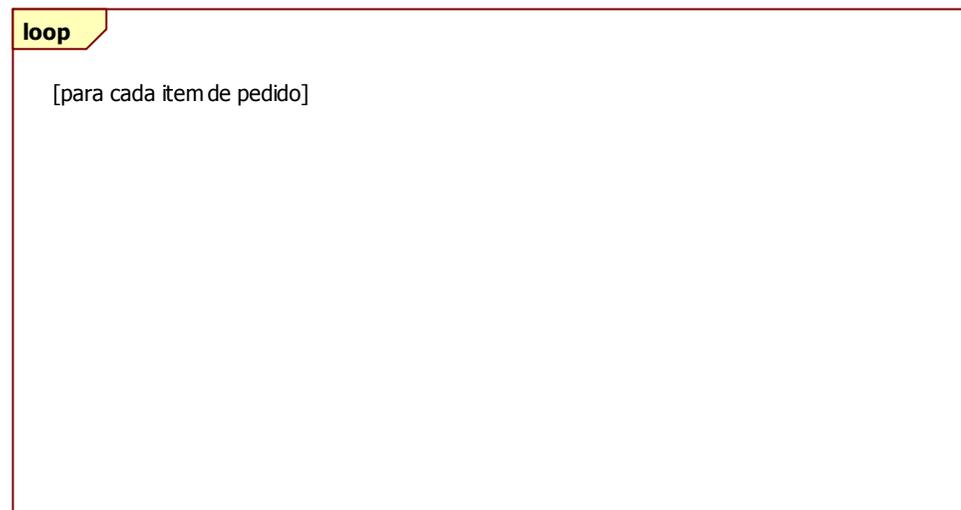
 Despacha normalmente

Se precisa de confirmação

 Envia confirmação

Repetições

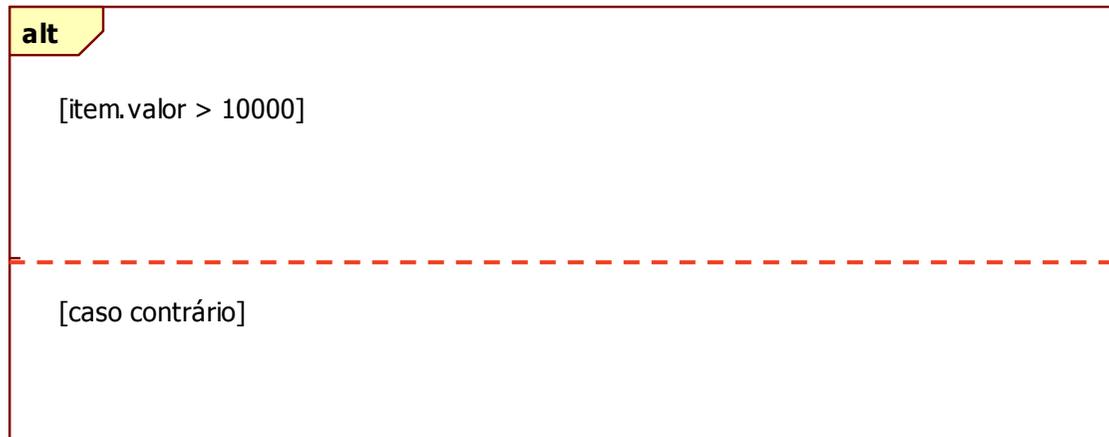
- O diagrama de seqüência permite que repetições sejam feitas durante o fluxo
- Para isso são utilizados quadros (*frames*) do tipo *loop*
- A condição entre [] é a condição de execução do *loop*



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
- A condição entre [] é a condição da decisão

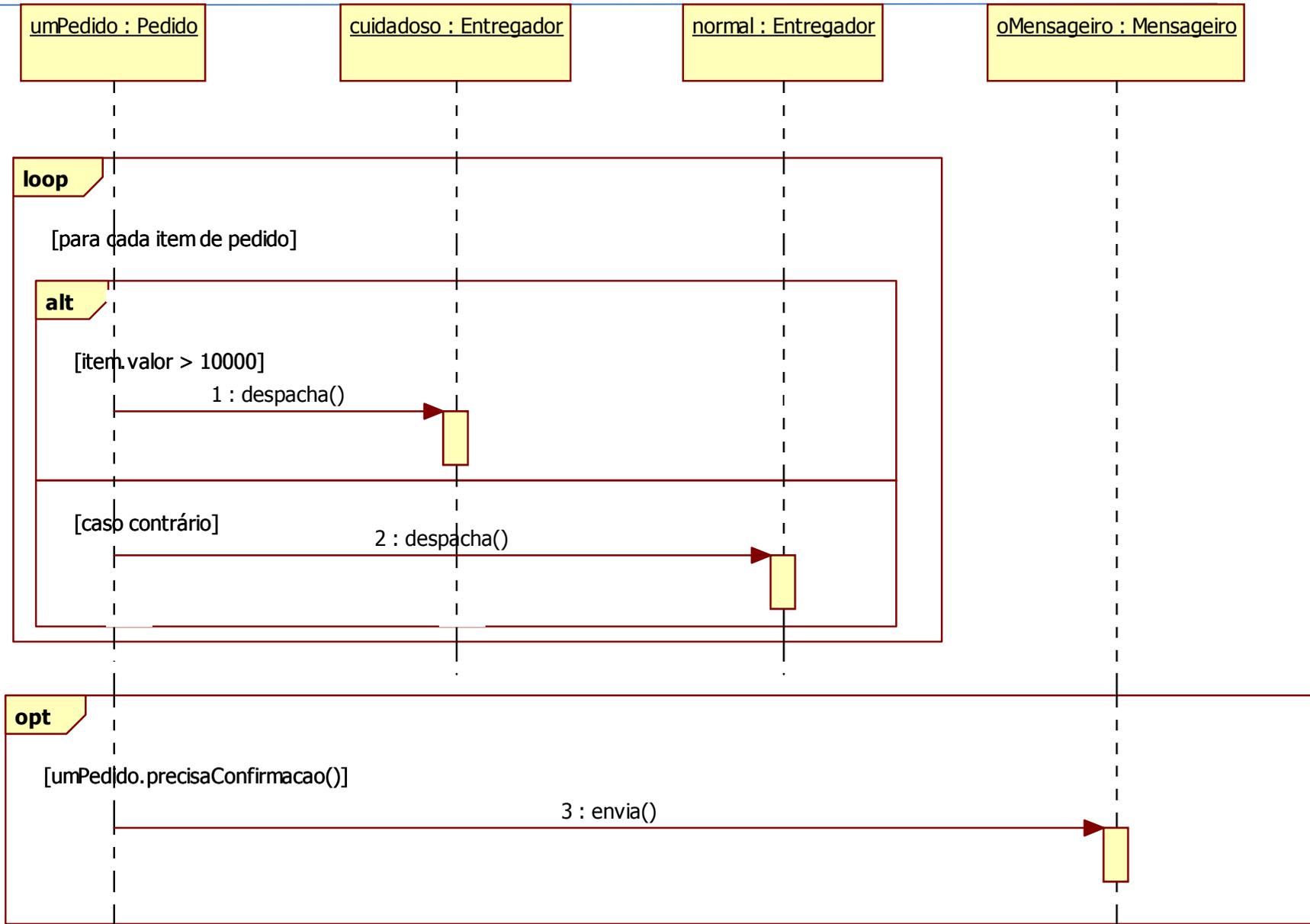
If + else



If



Exemplo

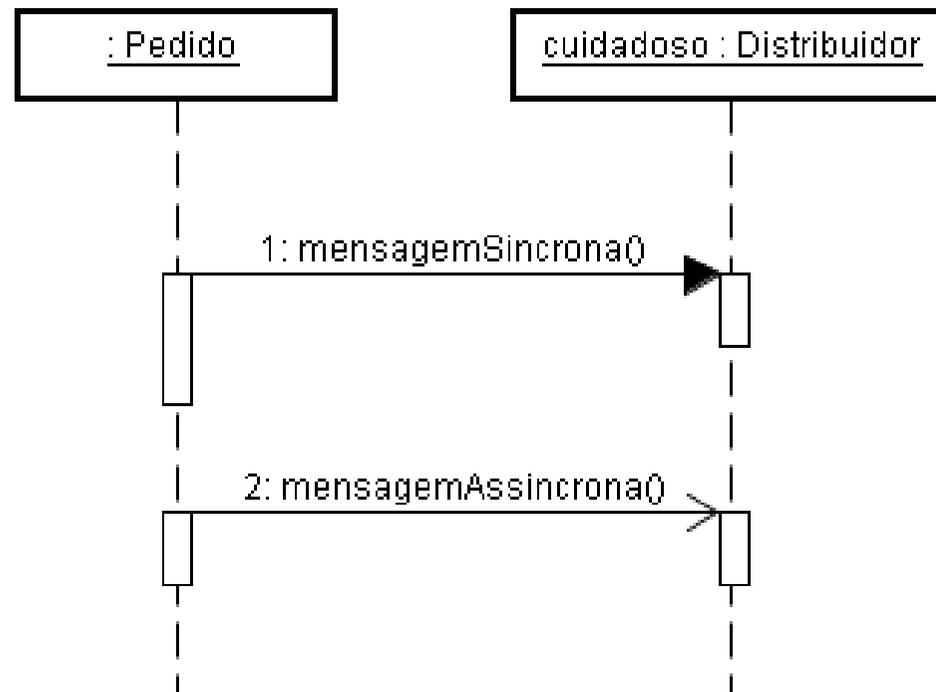


Outros quadros disponíveis

- 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)

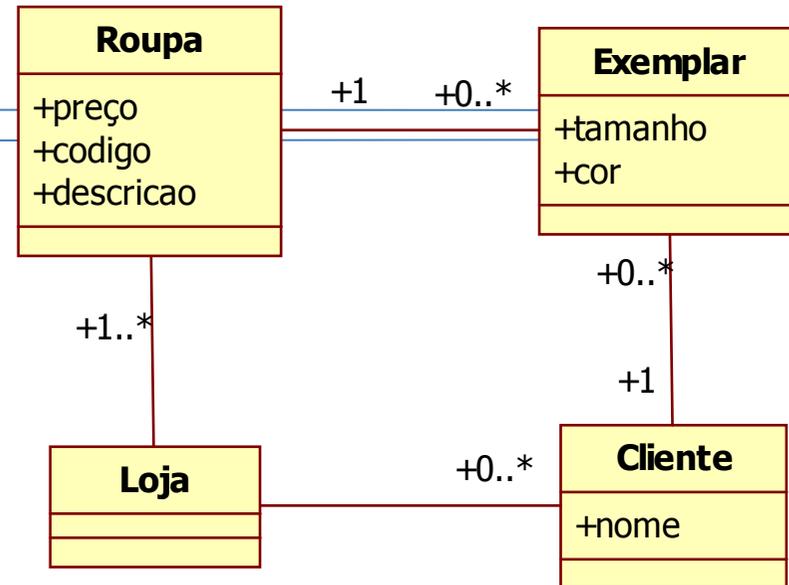


Quando utilizar diagrama de seqüência?

- 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 definição precisa de como será o código

Exercício

- Uma loja que vende roupas e cada roupa possui um preço, um código e uma descrição. Um roupa pode possuir vários exemplares. Cada exemplar tem um tamanho e uma cor associados.



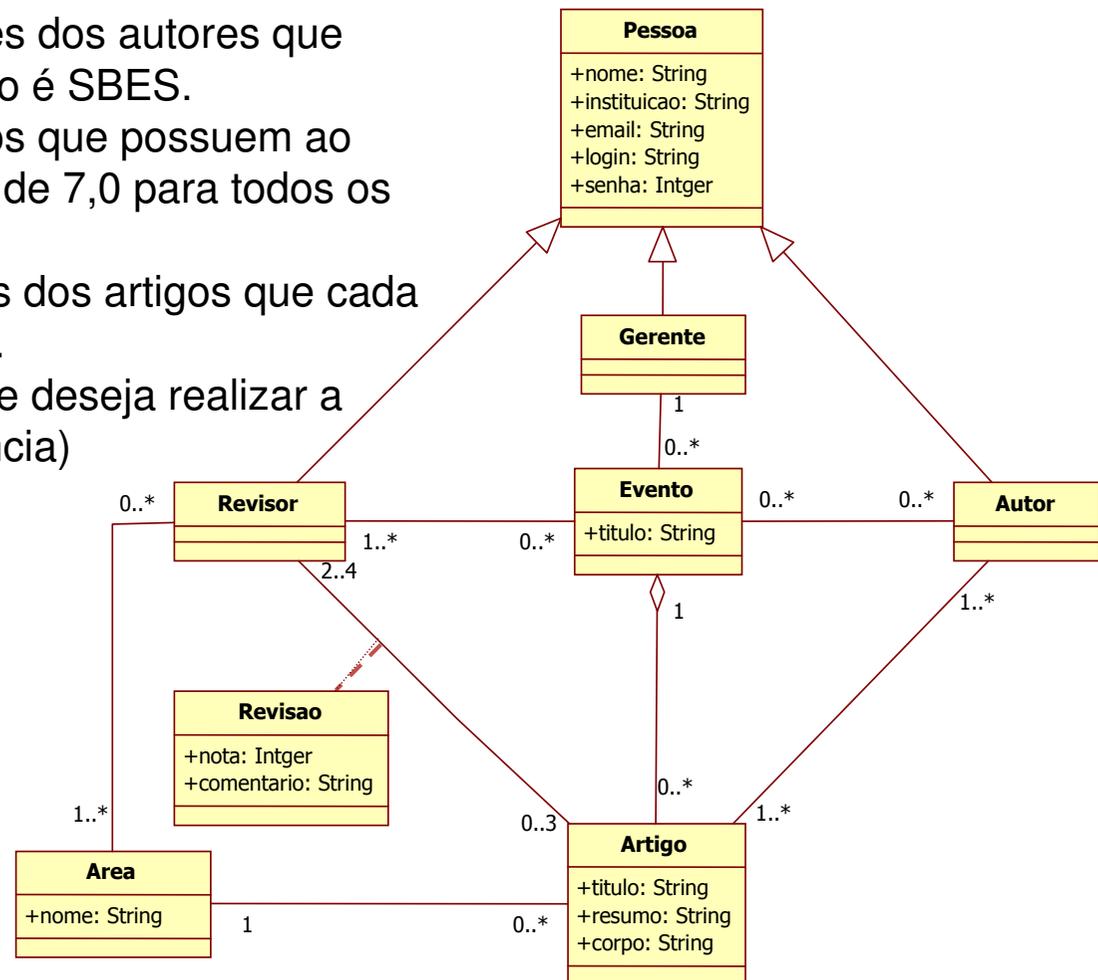
- Os clientes da loja são cadastrados pelo nome e quando a venda é realizada, o sistema guarda informação de qual(is) foi(foram) o(s) exemplar(es) vendido(s) para o cliente.
- Faça um diagrama de sequência que modele um sistema capaz de respondendo as perguntas abaixo:
 - Quantos exemplares a roupa com código 123 possui?
 - Quais são os códigos das roupas compradas pela cliente Ana?
 - Quais são os nomes dos clientes que já compraram um exemplar da roupa com código de barra 123?
 - Qual o tamanho da roupa com código 123 comprada pela cliente Ana?(obs: utilize uma instancia da classe Loja para iniciar a modelagem)

Exercício 2 (parte I/II)

- Um sistema de gerenciamento de submissões de artigos para um evento automatiza o processo de envio de artigos para o evento, de distribuição de artigos para os revisores, de envio das avaliações sobre os artigos para os autores e de envio da versão final do artigo modificada de acordo com as avaliações feitas pelos revisores.
- Vários eventos podem ser gerenciados ao mesmo tempo. Cada evento possui um título, um único gerente, vários revisores, vários autores e é composto por um conjunto de artigos submetidos para o evento. Um mesmo artigo só pode ser submetido para um único evento.
- Todas as pessoas que acessam o sistema de gerenciamento de submissões de artigos possuem um nome, uma instituição à qual pertencem, um email, um login e uma senha. Um artigo pode ser escrito por vários autores, possui um título, um resumo, um corpo de texto e uma área.
- Todo artigo tem no mínimo 2 e no máximo 4 revisores e cada revisor pode fazer a revisão de até 3 artigos. Um revisor possui áreas de expertise e só pode revisar os artigos destas áreas. A revisão de um artigo feita por um revisor contém uma nota dada pelo revisor e um comentário.

Exercício 2 (parte II/II)

- Seguindo este diagrama e adicionando os métodos necessários, faça um diagrama de seqüência para cada um dos itens abaixo representando todos os gets e sets necessários.
 - Revisor deseja alterar a nota da revisão do artigo cujo título é “ES1”.
 - Gerente quer saber quais são os nomes dos autores que enviaram artigos para o evento cujo título é SBES.
 - Gerente quer saber quais são os artigos que possuem ao menos uma das notas de revisão acima de 7,0 para todos os eventos dos quais é gerente.
 - Gerente quer saber quais são os títulos dos artigos que cada um dos revisores do SBES pode revisar.
- (Dica: utilize uma instancia da classe que deseja realizar a tarefa para iniciar o diagrama de seqüência)



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>