Infnet
Projeto de Bloco

Projeto de Bloco

Engenharia Disciplinada de Softwares

Graduação em Engenharia de Software - 2020

Tarefas Solicitadas na Aula Passada - 20/08/2020

1. Revisar os diagramas de Casos de Uso e Classes (modelo de domínio), incluindo os atributos no modelo de domínio.
2. Elaborar um esboço do Diagrama de Pacotes do VenturaHR.
3. Iniciar a elaboração do TP3 a ser entregue no dia 31/08 (o enunciado será publicado amanhã 21/08).

Infnet
Projeto de Bloco

Etapa 3 Aula 1

Disciplined Agile Delivery - Inception

Infnet
Projeto de Bloco

Projeto Orientado a Objetos

Projeto Orientado a Objetos

Depois de identificar seus requisitos e criar um modelo de domínio, adicione métodos às classes apropriadas e defina a mensagem entre os objetos para atender aos requisitos.

Projeto Orientado a Objetos

Estamos prontos para tirar nossos chapéus de analista e colocar nossos chapéus de designer-modelador.

Dadas uma ou mais das **entradas** de análise, os desenvolvedores:

1. começam a codificar imediatamente (de preferência com o desenvolvimento test-first),
2. começam alguma modelagem UML para o design do objeto ou
3. começam com outra técnica de modelagem.

Projeto Orientado a Objetos

Uma forma popular de pensar sobre o projeto de objetos de software e também de componentes em larga escala é em termos de responsabilidades, funções e colaborações.

Isso faz parte de uma abordagem mais ampla chamada design orientado por responsabilidade ou RDD.

No RDD , pensamos nos objetos de software como tendo responsabilidades - uma abstração do que eles fazem.

Projeto Orientado a Objetos

A UML define uma responsabilidade como “um contrato ou obrigação de um classificador”.

As responsabilidades estão relacionadas às obrigações ou comportamento de um objeto em termos de sua função.

Basicamente, essas responsabilidades são dos seguintes tipos: fazer e saber.

Projeto Orientado a Objetos

As responsabilidades de execução de um objeto incluem:

- fazer algo por si mesmo, como criar um objeto ou fazer um cálculo.
- iniciando ação em outros objetos.
- controlar e coordenar atividades em outros objetos.

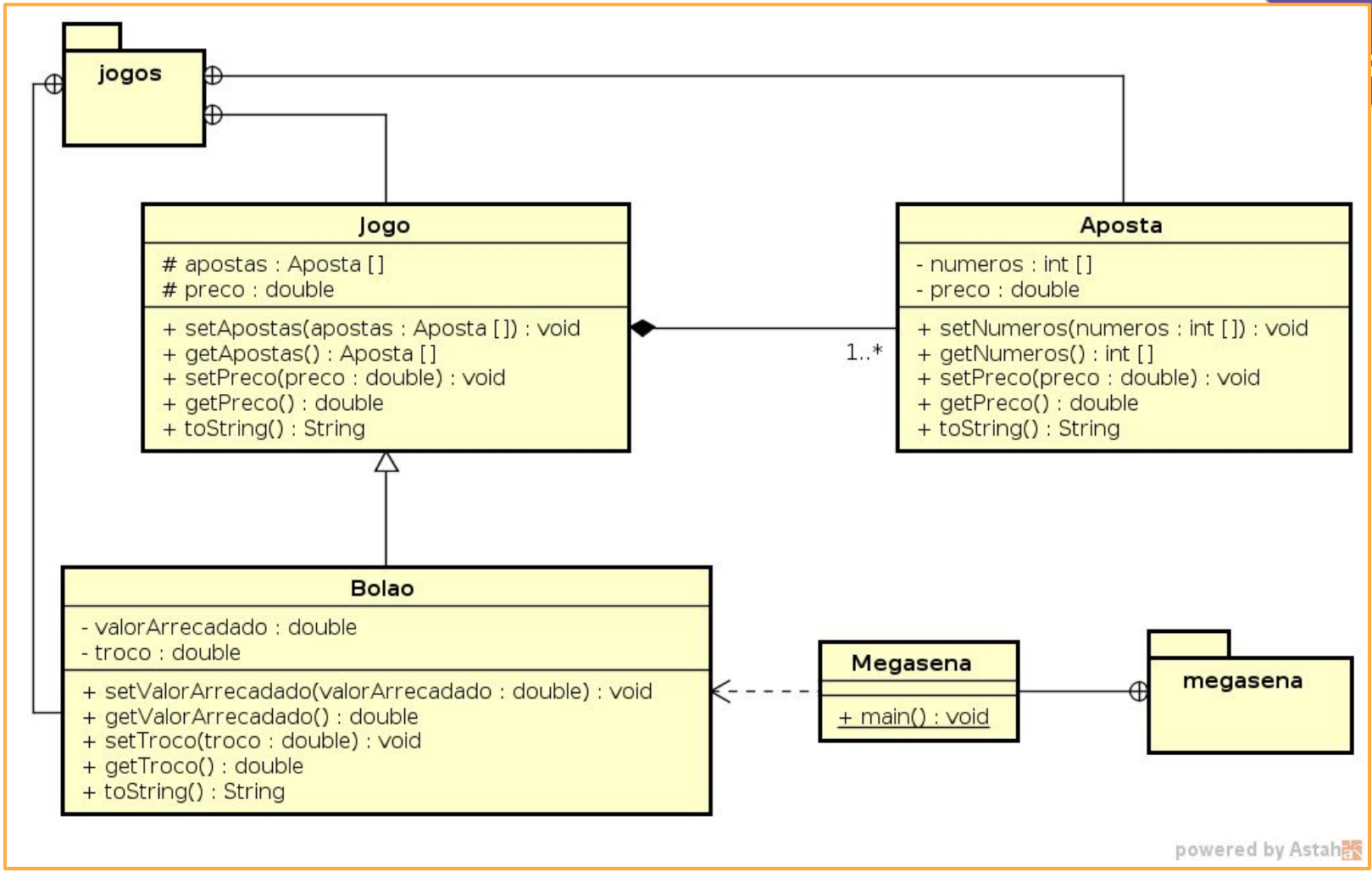
Conhecer as responsabilidades de um objeto inclui:

- saber sobre dados encapsulados privados.
- saber sobre objetos relacionados.
- saber sobre coisas que pode derivar ou calcular.

Projeto Orientado a Objetos

O RDD também inclui a ideia de colaboração: as responsabilidades são implementadas por meio de métodos que agem isoladamente ou colaboram com outros métodos e objetos.

Por exemplo, a classe Venda pode definir um ou mais métodos para saber seu total; digamos, um método denominado getTotal. Para cumprir essa responsabilidade, a Venda pode colaborar com outros objetos, como enviar uma mensagem getSubtotal para cada objeto SalesLineItem solicitando seu subtotal.


jogos
Jogo
apostas : Aposta []
preco : double
+ setApostas(apostas : Aposta []) : void
+ getApostas() : Aposta []
+ setPreco(preco : double) : void
+ getPreco() : double
+ toString() : String
1..*
Aposta
- numeros : int []
- preco : double
+ setNumeros(numeros : int []) : void
+ getNumeros() : int []
+ setPreco(preco : double) : void
+ getPreco() : double
+ toString() : String
Bolao
- valorArrecadado : double
- troco : double
+ setValorArrecadado(valorArrecadado : double) : void
+ getValorArrecadado() : double
+ setTroco(troco : double) : void
+ getTroco() : double
+ toString() : String
Megasena
+ main() : void
megasena
powered by Astah

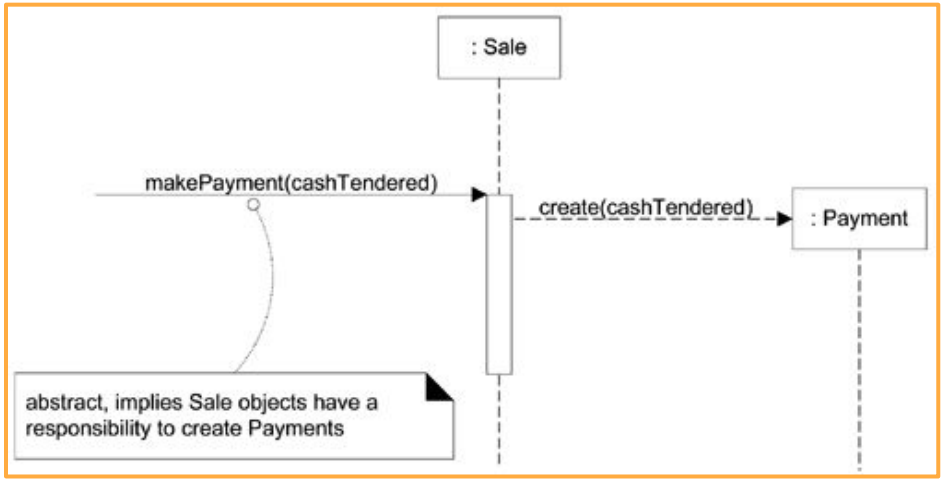
Projeto Orientado a Objetos

Você pode pensar em atribuir responsabilidades a objetos durante a codificação ou modelagem.

Dentro da UML , desenhar diagramas de interação se torna a ocasião para considerar essas responsabilidades (realizadas como métodos).

Projeto Orientado a Objetos

A Figura indica que os objetos Venda receberam a responsabilidade de criar Pagamentos, que é concretamente invocado com uma mensagem makePayment e tratada com um método makePayment correspondente.



Além disso, o cumprimento dessa responsabilidade requer colaboração para criar o objeto Pagamento e invocar seu construtor.

Projeto Orientado a Objetos

General Responsibility Assignment Software Patterns (ou Principles), abreviado para GRASP.

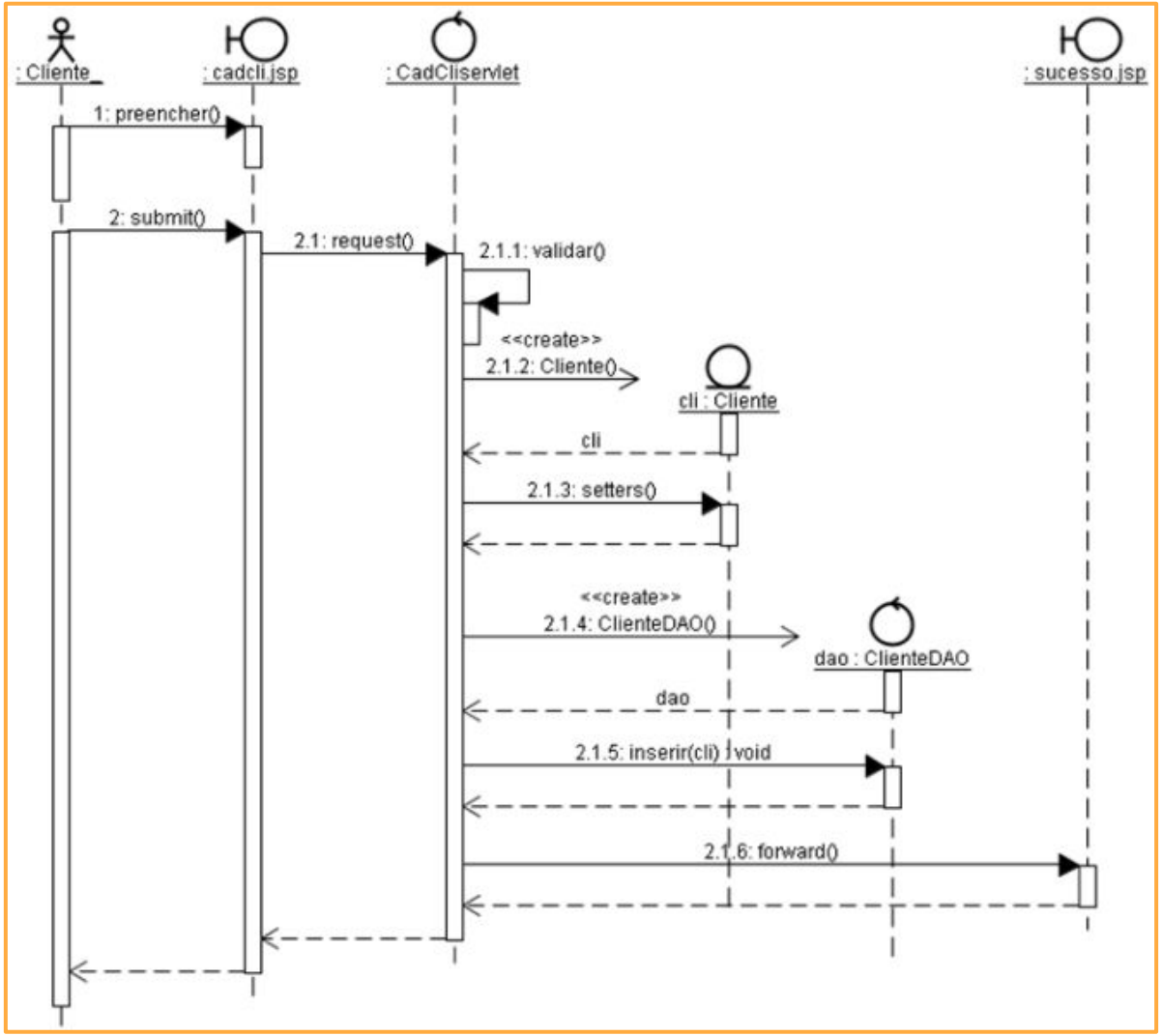
Os diferentes padrões e princípios usados no GRASP são controlador, criador, indireção, especialista em informações, baixo acoplamento, alta coesão, polimorfismo, variações protegidas e fabricação pura.

https://learning.oreilly.com/library/view/applying-uml-and/0131489062/ch17.html

Diagramas de Sequência

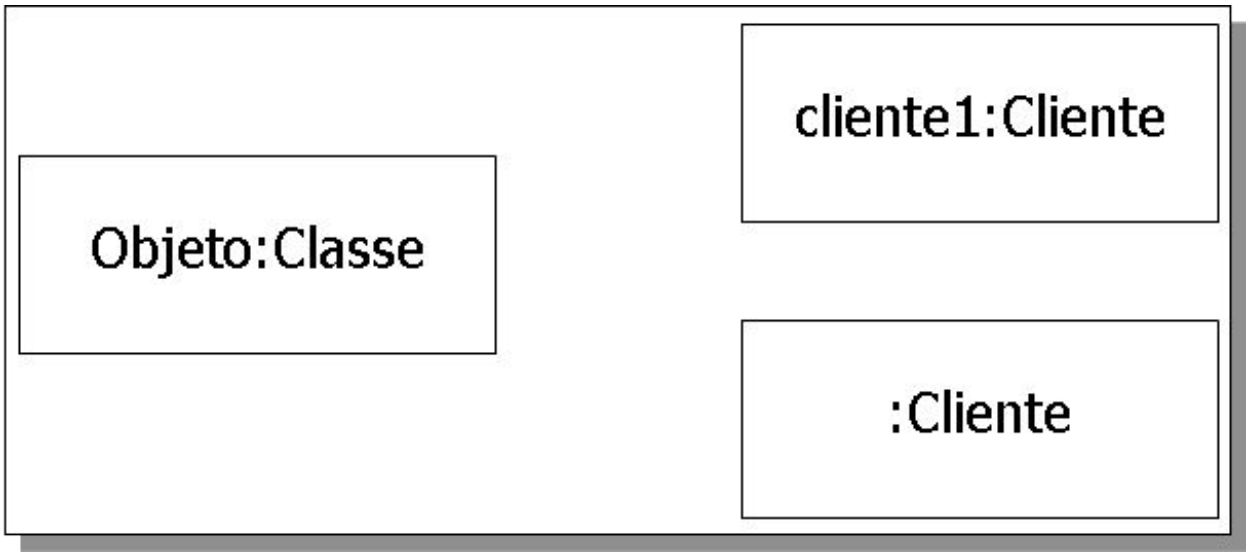
Diagramas de Sequência

- Mostram interações (trocas de mensagens) entre objetos em uma Sequência temporal.
- São utilizados para descrever casos de uso, métodos e serviços.
- Facilitam a visualização da dinâmica do sistema, pois mostram os métodos e situações em que serão chamados.
- Enfatizam o ordenamento temporal das operações.



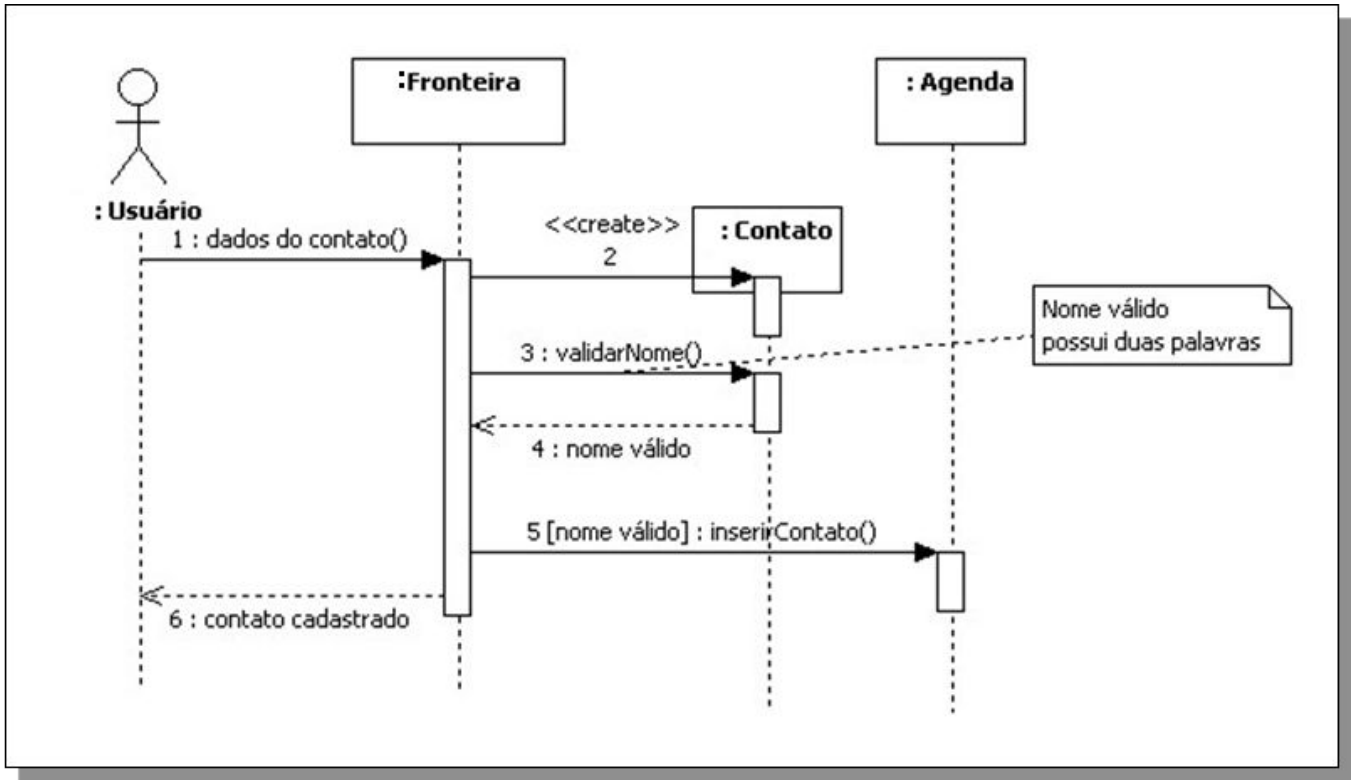
Componentes

- Os objetos são representados no diagrama seguindo a notação a seguir:



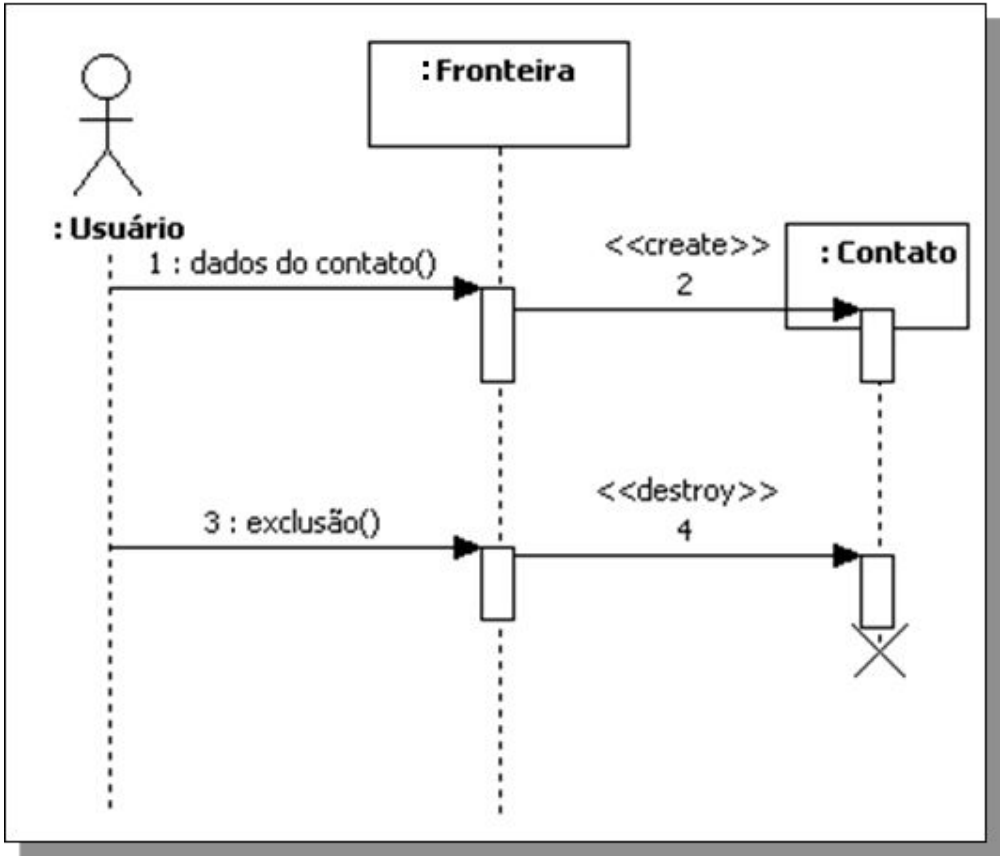
É comum o uso de objetos anônimos

Componentes

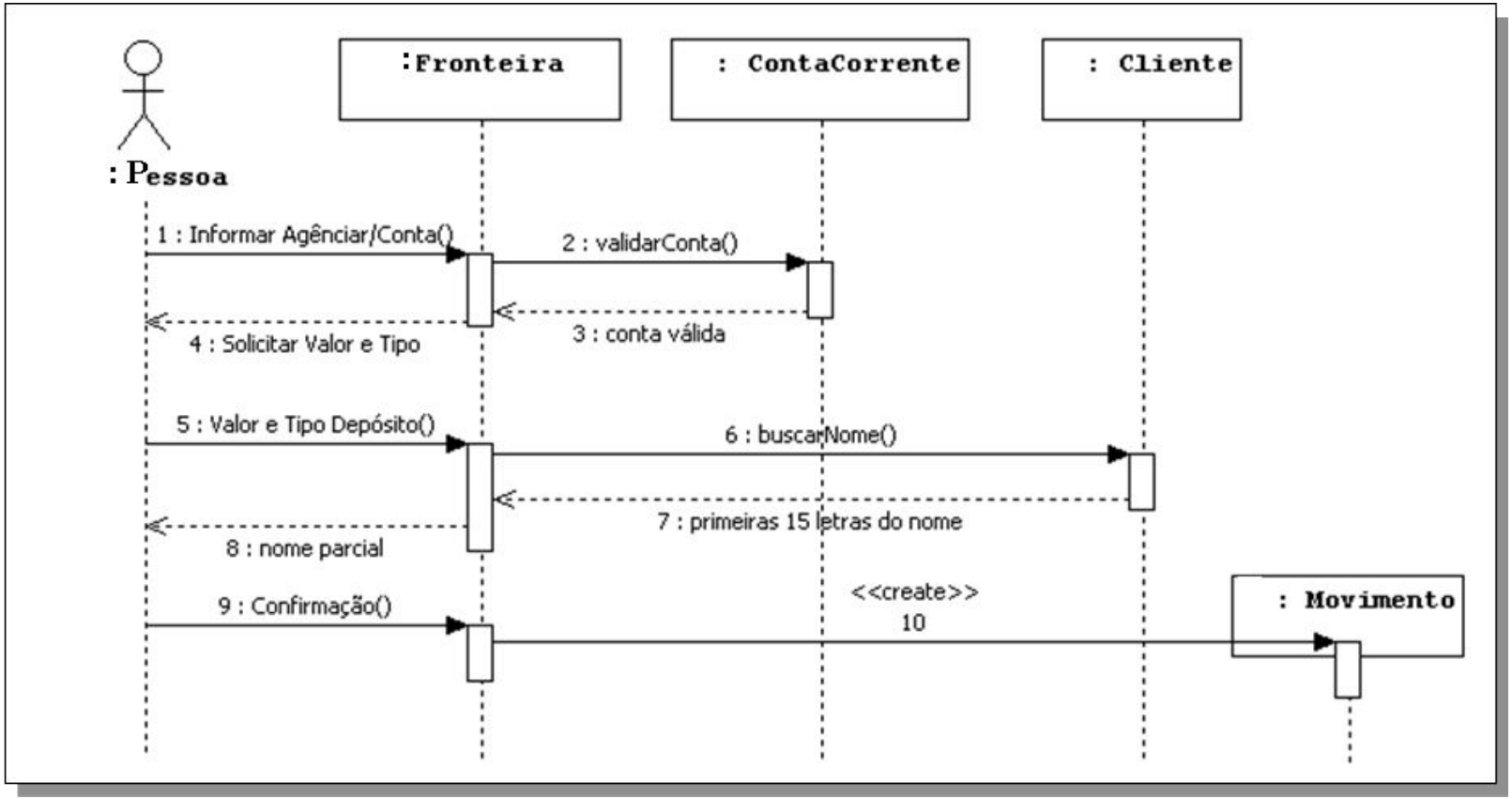


Cadastro em uma Agenda de Contatos

Criação e Destruição de Objetos



Banco Money


:Fronteira
: ContaCorrente
: Cliente
: Pessoa
1 : Informar Agênciar/Conta()
2 : validarConta()
3 : conta válida
4 : Solicitar Valor e Tipo
5 : Valor e Tipo Depósito()
6 : buscarNome()
7 : primeiras 15 letras do nome
8 : nome parcial
9 : Confirmação()
<<create>>
10
: Movimento

Partindo de um Caso de Uso

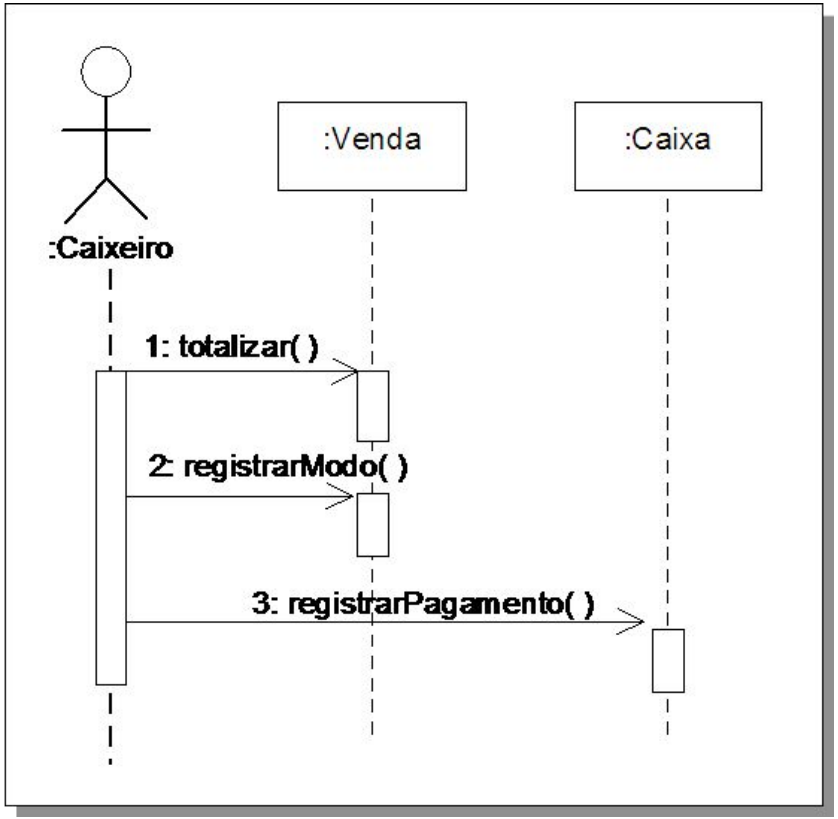
- Os diagramas de Sequência são orientados para exprimir o desenrolar temporal de Sequências de ações.
- É difícil representar lógicas de seleção e repetição sem prejudicar a legibilidade do diagrama.
- Os fluxos representam desdobramentos da lógica do caso de uso.
- É preferível usar diagramas separados para representar fluxos resultantes de diferentes caminhos lógicos.

Partindo de um Caso de Uso

- Registrar Venda:
  1. O Caixa registra itens de mercadoria.
  2. O Sistema totaliza a venda.
  3. O Caixa registra o modo de venda.
  4. Se a venda for a prazo, o Caixa insere a venda em contas a receber.
  5. Caso contrário, o Caixa registra o pagamento.

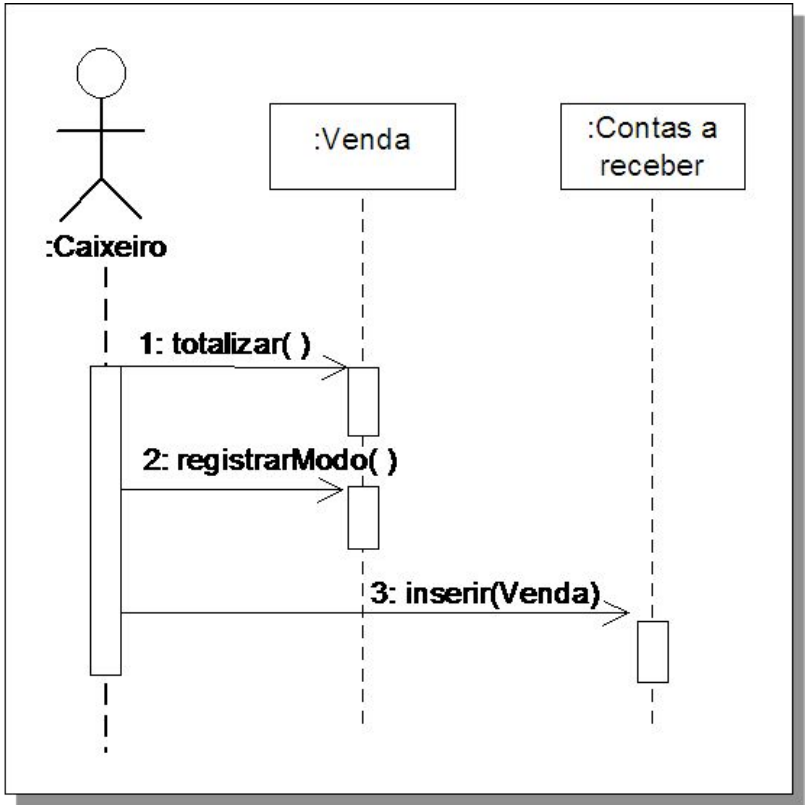
Partindo de um Caso de Uso

- Venda à Vista:
- Cria-se o Diagrama de Sequência Principal.
- O fluxo alternativo é ignorado, pois será desenhado em outro diagrama de Sequência.



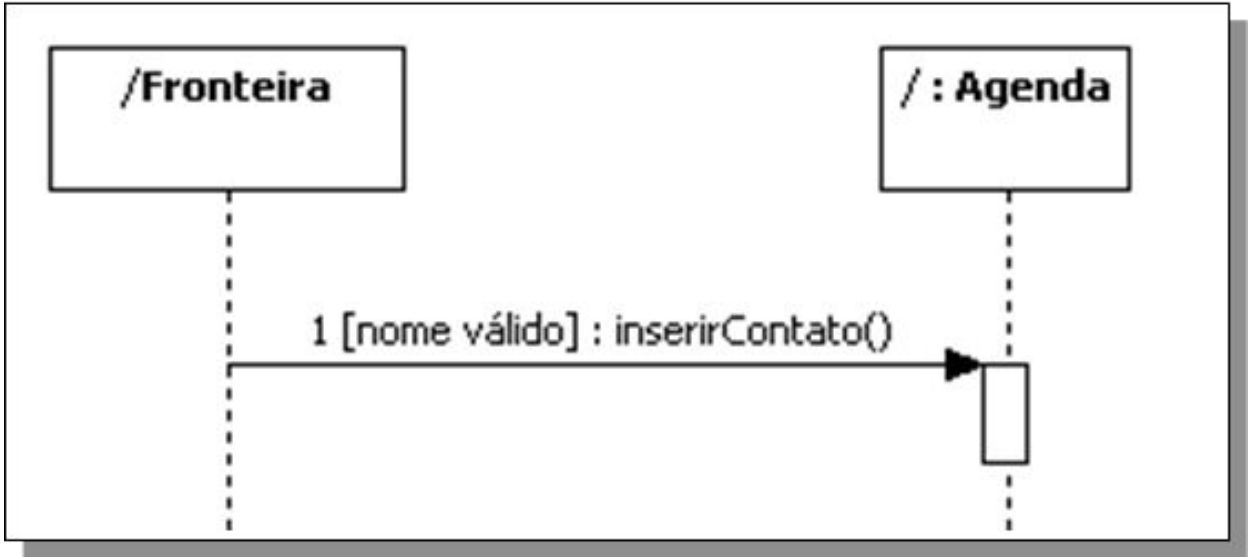
Partindo de um Caso de Uso

- Venda a Prazo:
- Diagramas de Sequência Alternativos.
- Neste diagrama foi desenhado o fluxo alternativo.

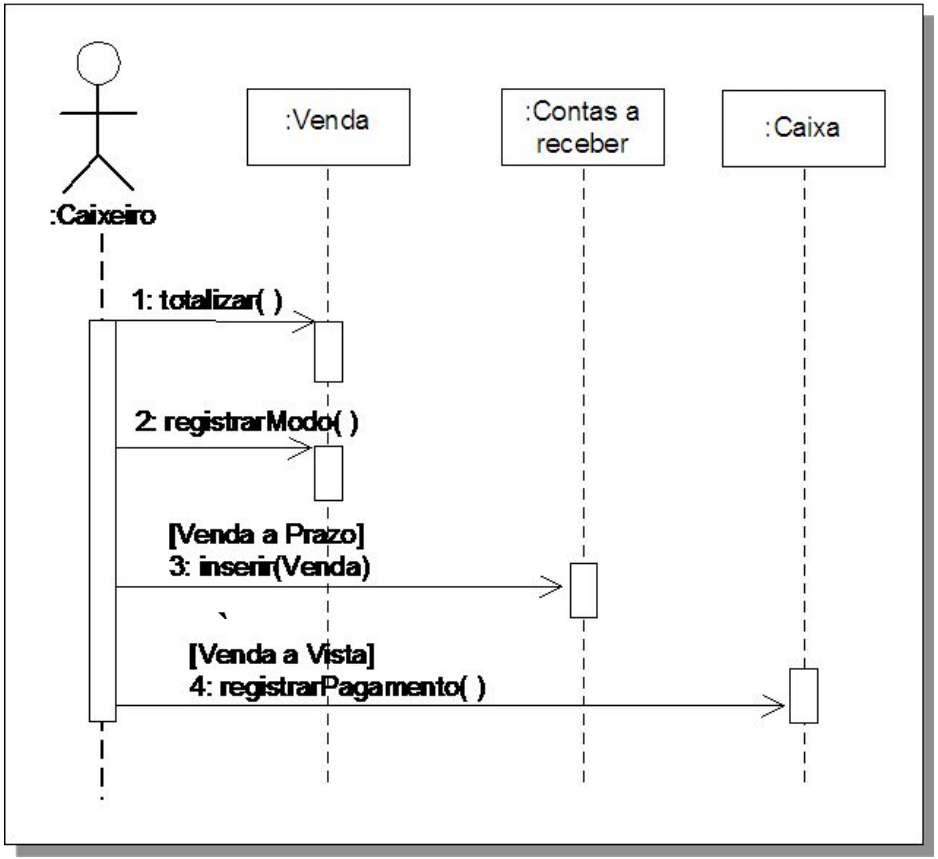


Condição de Guarda

- Um fluxo pode ser executado dependendo de uma determinada condição.
  - [condição] - lógica de seleção.



Condição de Guarda



Anotações

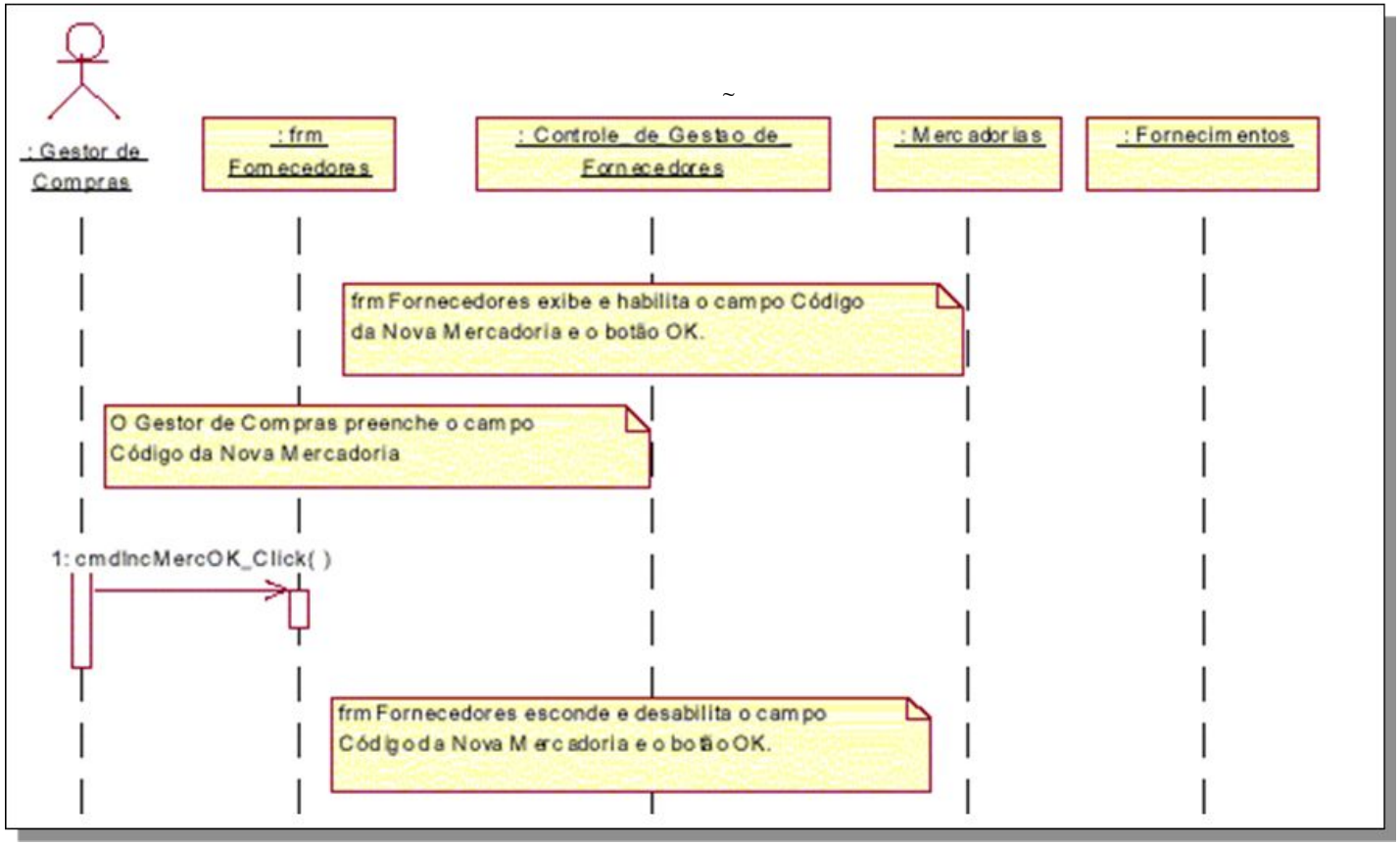
- Diagramas de Sequência também podem ser complementados por meio de anotações.

- Os detalhes de processamento que não correspondem a interações entre classes são lançados como anotações.

- Essas anotações podem servir como especificações das operações referentes às classes envolvidas.

Cuidado para não poluir os diagramas

Anotações


: Gestor de Compras
: frm Fornecedores
: Controle_de_Gestao_de_Fornecedores
: Mercadorias
: Fornecimentos
frm Fornecedores exibe e habilita o campo Código da Nova Mercadoria e o botão OK.
O Gestor de Compras preenche o campo Código da Nova Mercadoria
1: cmdIncMercOK_Click( )
frm Fornecedores esconde e desabilita o campo Código da Nova Mercadoria e o botão OK.

Tarefas para Próxima Aula - 03/09/2020

1. Concluir o TP3 e entregar no dia 31/08/2020.
2. Estudar o capítulo 17 do livro do Craig Larman.
3. Evoluir a Arquitetura Lógica do projeto.