

DCC-UFMG Ensino à Distância

Estudo de Caso

Descrição 1

Virtual LTDA é o assunto do nosso estudo de caso que se estenderá durante o curso. Uma empresa pequena localizada em Belo Horizonte, Minas Gerais, a Virtual LTDA é especializada em equipamentos musicais (guitarra, violão, etc). Virtual LTDA está adicionando a sua linha de produtos, outros produtos. Incluindo tecnologias de mixagem e gravação, microfones e CD *players*. A Virtual LTDA é uma empresa ambiciosa, ela deseja abrir uma filial em São Paulo dentro de um ano.

Algumas de suas vendas originam-se de organizações comerciais. Os fundadores da Virtual LTDA, acreditam que não podem efetivamente administrar sua empresa com um sistema de cobrança e cadastro de pedidos inadequados. Os sócios não são especialistas em tecnologia de informação, mas eles sabem que o sucesso depende da extensibilidade do sistema que será desenvolvido. A preocupação inicial é a aplicabilidade da Internet à seus negócios. Eles querem minimizar os custos com a tecnologia usada enquanto mantêm a flexibilidade na troca de plataforma. Eles fazem as seguintes observações sobre a empresa:

- Muitos de seus produtos são muito caros. Neste caso os balconistas têm participação nas vendas.
- Muitos dos clientes querem ser capazes de obter informação de um determinado produto sem interagir com um representante da empresa.
- Muitos clientes querem pedir outros produtos a partir da Virtual LTDA, os quais não são oferecidos, incluindo microfone, tecnologia de mixagem, etc.

Eles acreditam que uma solução baseada na Internet poderá beneficiar a Virtual LTDA. Nosso desafio é projetar um sistema que atenda as necessidades imediatas da empresa e que seja flexível o suficiente para suportar outros tipos de produtos no futuro.

Descrição 2

No início de cada semestre os alunos devem requisitar um catálogo de cursos contendo os cursos oferecidos no semestre. Este catálogo deve conter informações a respeito de cada curso tais como: professor, departamento e pré-requisitos. Desse modo, os alunos podem tomar suas decisões mais apropriadamente.

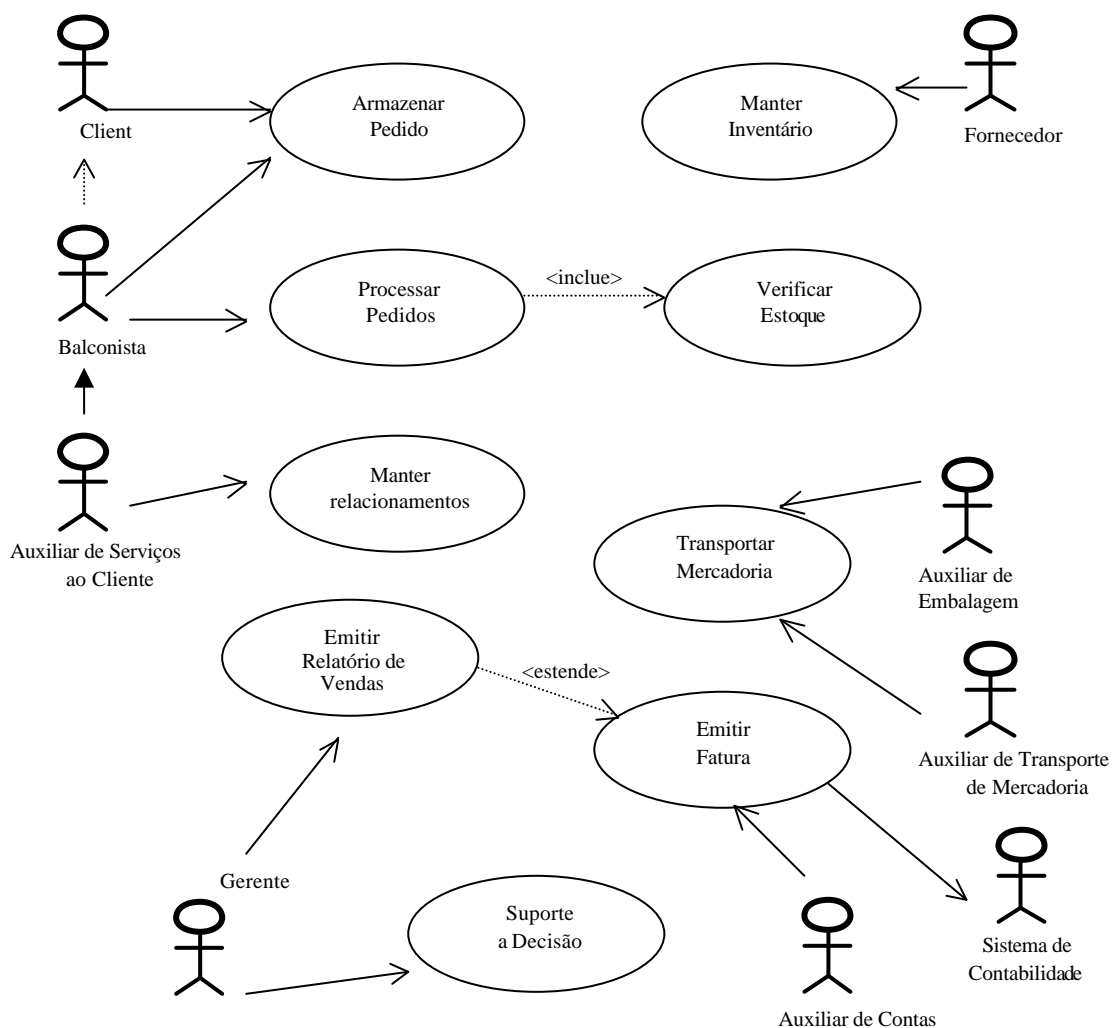
O novo sistema permitirá que os alunos selecionem quatro cursos oferecidos para o próximo semestre. Além disso, o aluno indicará dois cursos alternativos, caso o aluno não possa ser matriculado na primeira opção. Cada curso deverá ter no máximo de 10 e mínimo de 3 alunos. Um curso com o número de alunos inferior a 3 será cancelado. Para cada matrícula feita por um aluno, o sistema envia informação ao sistema de cobrança para que cada aluno possa ser cobrado durante o semestre.

Os professores devem acessar o sistema “on-line”, indicando quais cursos irão lecionar. Eles também podem acessar o sistema para saber quais alunos estão matriculados em cada curso. Em cada semestre, há um prazo para alteração de matrícula, que corresponde

ao período de matrícula. Os alunos devem poder acessar o sistema durante esse período para adicionar ou cancelar cursos.

1. Diagrama de Caso de Uso

É um diagrama usado para se identificar como o sistema se comporta em várias situações que podem ocorrer durante sua operação. Descreve o sistema, seu ambiente e a relação entre os dois. Os casos de uso são usados para obter requisitos a partir da perspectiva do usuário. Os componentes deste diagrama são os Atores e os Casos de Uso. Abaixo é mostrado um diagrama de Caso de Uso da Virtual LTDA.



a. Evento

Eventos podem ser definidos como ações ou estímulos que causam mudanças no sistema e exigem alguma reação do sistema. Os eventos podem ser classificados em duas categorias: eventos externos e eventos internos.

- i. **Eventos externos:** São os mais fáceis de encontrar. Eles são qualquer estímulo ao sistema que origina a partir de fora dos limites do sistema. Por exemplo, a solicitação de um pedido pelo cliente. Para identificar eventos externos, focalize nos eventos e faça perguntas, quem e o que esta estimulando o evento?
- ii. **Eventos internos:** São mais diversos. Um tipo de evento interno é um temporizador que provoca alguma reação no sistema. Novamente focalize nos eventos e faça perguntas, quem e o que esta estimulando o evento?

b. Ator



Representa um Ator.

Representa qualquer entidade que interage com o sistema. Pode ser uma pessoa, outro sistema, etc. Algumas de suas características são descritas abaixo:

- Um ator é representado por um boneco, o qual é rotulado com o nome do papel. O ator descreve o papel que um usuário assume com relação ao sistema.

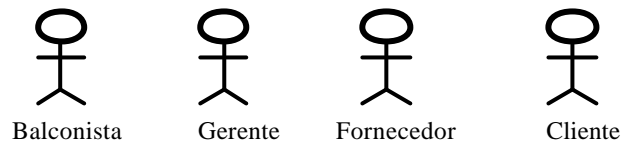


Cliente

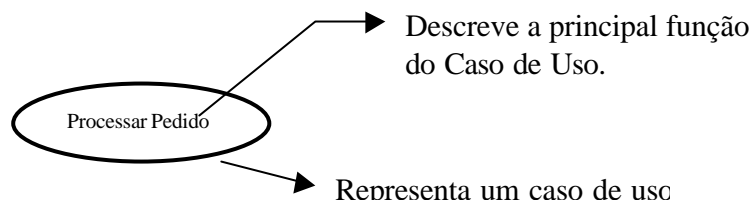
- Ator não é parte do sistema. Representa os papéis que o usuário do sistema pode desempenhar.
- Ator pode interagir ativamente com o sistema.
- Ator pode ser um receptor passivo de informação
- Ator pode representar um ser humano, uma máquina ou outro sistema.

c. Papel

Podemos definir um papel como sendo o personagem representado por um ator. Por exemplo, em uma peça de teatro você como ator poderia representar ou o papel de bandido ou o papel de policial. Portanto, um ator representa um papel, não um usuário individual do sistema: uma pessoa pode assumir diferentes atores no sistema.

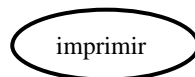


d. Caso de Uso



Um Caso de Uso (em inglês *use case*) é uma interação típica entre um usuário e um sistema. Um caso de uso captura alguma função visível ao usuário e, em especial, busca atingir uma meta do usuário. Assim um caso de uso pode ser definido como uma seqüência de ações que o sistema executa e produz um resultado de valor para o ator. Algumas de suas características são descritas abaixo:

- Um caso de uso é representado por uma elipse, a qual é rotulada com um verbo que representa uma função do sistema.



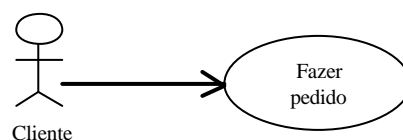
- Um caso de uso modela o diálogo entre atores e o sistema.
- Um caso de uso é iniciado por um ator para invocar uma certa funcionalidade do sistema.
- Um caso de uso é um fluxo de eventos completo e consistente.
- O conjunto de todos os caso de uso representa todas as situações possíveis de utilização do sistema.

e. Relacionamentos de Caso de Uso

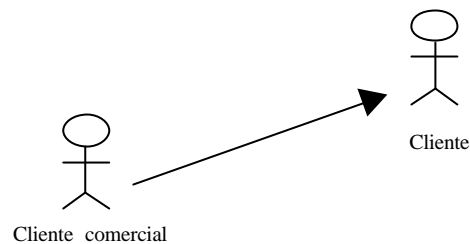
Um relacionamento é definido como uma conexão entre objetos e serve de canal de comunicação entre eles. Geralmente, é representado por uma linha que conecta dois ou mais objetos, podendo ser direcional.

Representam um tipo de conexão entre atores e caso de uso ou entre casos de uso.

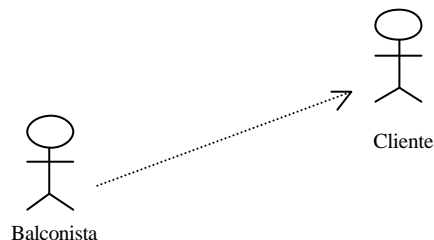
- **associação:** Representamos um relacionamento entre um ator e um caso de uso por uma seta simples. Único relacionamento entre atores e casos de uso.



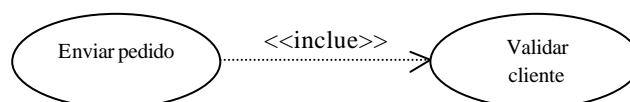
- **Generalização:** Representamos um relacionamento de herança entre atores como mostrado abaixo.



- **Dependência:** a dependência é representada por uma seta tracejada, a partir do Balconista ao Cliente. Neste caso temos um relacionamento em que um ator depende de outro ator. No exemplo abaixo, a interface do usuários será gerada visando o Balconista, não o cliente. Portanto o Balconista não é capaz de cadastrar um pedido sem a entrada do Cliente.



- **<<include>>:** um relacionamento de inclusão entre dois casos de uso significa que um caso de uso base incorpora explicitamente o comportamento de outro caso de uso. Uma relação de inclusão se representa usando uma seta pontilhada e rotulada com a palavra *include*.



O relacionamento acima poder ser lido do seguinte modo: o caso de uso Enviar pedido inclui o caso de uso Validar Cliente.

Quando deve ser usado o relacionamento de extensão:

- Comportamentos comuns em diferentes casos de uso.
 - Necessidade de melhorar o entendimento de um caso de uso: gerencia redundância e flexibiliza mudanças.
 - O relacionamento de inclusão permite a um caso de uso, incluir o fluxo de eventos especificado em um outro caso de uso.
- **<<estende>>**: um relacionamento de extensão indica que um caso de uso base incorpora implicitamente o comportamento de outro caso de uso. Um caso de uso pode estender somente em certos pontos, chamados pontos de extensão. Uma relação de extensão se representa usando uma seta pontilhada e rotulada com a palavra *estende*. Um relacionamento de extensão é usado para modelar a parte de um caso de uso que o usuário pode ver como o comportamento opcional do sistema. Desta forma se separa o comportamento opcional do comportamento obrigatório.



O relacionamento acima poder ser lido do seguinte modo: o caso de uso Enviar pedido parcial estende o caso de uso Enviar pedido.

Como funciona o relacionamento de extensão :

- Em um ponto de extensão, sobre certas circunstâncias, o comportamento estendido é executado.
- O controle é retomado para o caso de uso base no mesmo ponto onde a extensão foi executada.
- Cada ponto de extensão deve ter um nome único no caso de uso base.

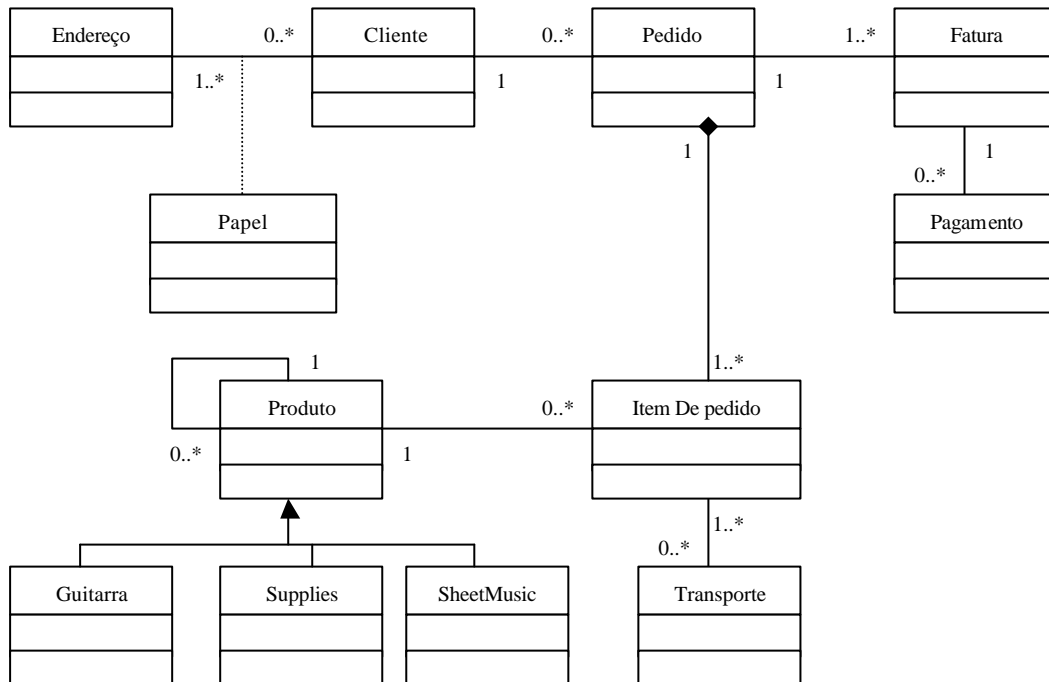
Quando deve ser usado o relacionamento de extensão:

- Para adicionar novos comportamentos sob certas condições, ou seja, um comportamento opcional do sistema.
- Incorporar requisitos funcionais específicos que não fazem parte do fluxo do caso de uso base.

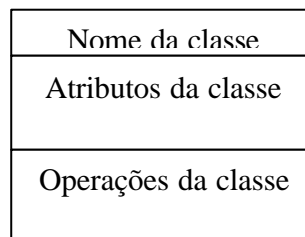
2. Diagrama de Classes

O diagrama de classes descreve os vários tipos de objetos que existem no sistema e os relacionamentos estáticos que existem entre eles. O diagrama de classes também mostra os atributos (propriedades) e as operações (métodos) de uma classe. Diagramas de classe

permitem, planejar como as classes/objetos funcionarão e interagirão. A figura abaixo mostra o diagrama de classes para a Virtual LTDA.



Uma classe é representada como mostrada abaixo:



Uma classe é uma descrição de um conjunto de objetos que partilham os mesmos atributos, operações, relações e semântica. Por exemplo, O cliente “João da Silva” pode ser considerado um objeto relevante num sistema que pretende manipular informação referente aos clientes de uma empresa.

Atributo da classe: são propriedades que permite identificar uma classe ou um objeto. O João da Silva além do nome, também é caracterizado por outros atributos, nome, endereço, número do contribuinte, CPF, etc.

Operações da Classe: O João da Silva possui uma identidade própria, isto é, para a empresa, ele é distinto de todos os outros clientes. Sobre o cliente João da Silva podem-se efetuar várias operações, nomeadamente emitir-lhe faturas, efetuar alterações de endereço, apaga-lo do sistema.

O João da Silva relaciona-se com a empresa através, por exemplo, dos produtos ou serviços que adquire. Provavelmente existirão outros clientes na empresa, e todos eles são caracterizados pelo mesmo conjunto de atributos, pelas mesmas operações e relações e todos são distintos um dos outros. Esses diversos clientes podem ser agrupados em uma classe, a classe dos clientes da empresa. Note-se que os objetos não têm necessariamente que corresponder a entidades humanas ou, mais genericamente, a entidades com representação física (por exemplo, uma fatura).

a. Tipos de classes

- i. Entidade: É uma classe que modela objetos cuja informação e o comportamento associado são, de maneira geral, persistentes. No presente estudo de caso, as classes de objetos: clientes, produtos, são exemplos de classes de entidade.
- ii. Fronteira: classes de fronteira servem como fronteira entre os atores externos desejando interagir com a aplicação e a classe de entidade. Muitas classes de fronteira são componentes da interface do usuário, as quais tomam a forma de um formulário ou tela usados para interagir com a aplicação. Portanto classe de fronteira é uma classe que modela a comunicação entre a vizinhança do sistema e suas operações. Exemplos: interface do tipo janela, protocolo de comunicação, interface de impressão, etc.
- iii. Controle: classes de controle são coordenadoras da atividade no domínio da aplicação. Tipicamente, uma classe de controle pode assumir um dos vários papéis: como comportamento relacionado a transações, um serviço que separa os objetos de entidade a partir dos objetos de fronteira. Basicamente, uma classe de controle, é uma classe que modela o comportamento de controle específico para uma ou mais Caso de Uso. Suas principais características são:
 - Cria, ativa e anula objetos controlados.
 - Controla a operação de objetos controlados.
 - Controla a concorrência de pedidos de objetos controlados.

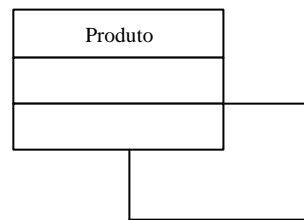
b. Relacionamentos entre Classes

- i. **Associação entre Classes:** o tipo mais comum de relacionamento em UML, uma associação define como os objetos de uma classe são conectados a objetos de outra classe. Sem essa associação nenhuma mensagem pode passar entre objetos da classe em tempo de execução. Existe uma associação entre duas classes se um instância de uma classe deve conhecer

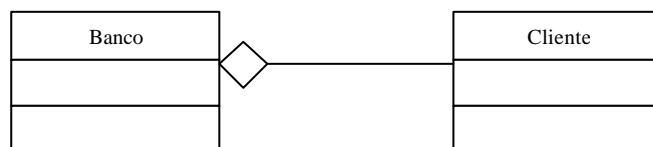
sobre a existência da outra de modo a realizar seu trabalho. No diagrama , uma associação é uma linha conectando duas classes.



- ii. **Associação Reflexiva entre Classes:** algumas vezes uma associação é necessária entre dois objetos da mesma classe. Chamamos esse tipo de associação de associação reflexiva. Isso poderá ser útil, por exemplo, quando um balconista esta vendendo uma guitarra e quer recomendar algum produto relacionado. Para prover essa característica e modelar e modelar –la de acordo com a UML, nós precisamos usar uma associação reflexiva na classe Produto.



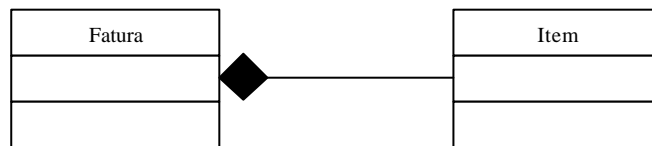
- iii. **Associação de Agregação entre Classes:** As agregações são um caso especial das associações. Uma agregação modela um relacionamento *tem um* (ou *parte de*, no jargão da UML) entre pares. Este relacionamento significa que um objeto contém outro. No contexto de uma agregação, os objetos podem existir independentemente uns dos outros. Nenhum objeto é mais importante do que o outro no relacionamento. Um losango aberto simboliza a agregação. O losango toca o objeto que é considerado o todo do relacionamento. O todo é constituído de partes.



Como o Banco e Cliente são independentes, eles são pares. Você pode dizer que o objeto Banco e o objeto Cliente são pares, porque os objetos Cliente podem existir independentemente do

objeto Banco. Isso significa, que se o objeto encerrar suas operações, os clientes não desaparecerão com o banco. Em vez disso, os clientes podem se tornar clientes de outro banco. Do mesmo modo o cliente pode sacar seus fundos e o banco continuará.

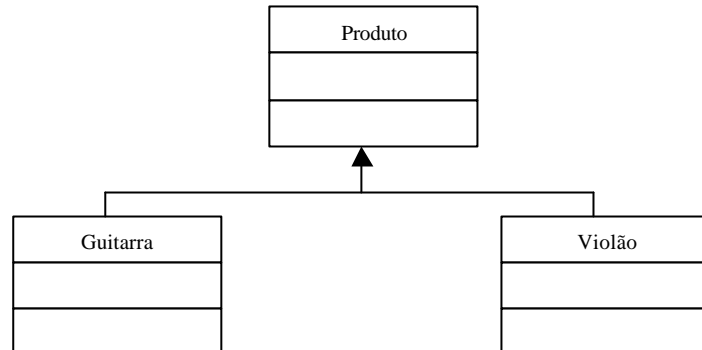
- iv. **Associação de Composição entre Classes:** As composições são um caso especial das associações de agregação. Elas representam a noção de composição e apenas têm sentido em associações “um para muitos” e, mais raramente, em associações “muitos para muitos”. Enquanto que nas associações anteriormente referidas não existem classes mais importantes, nas agregações existem uma classe (supra classe) que representa a agregação dos objetos da outra classe (sub classe). A título de exemplo, considere-se os itens (linhas) de uma fatura. Usualmente uma fatura é (ou pode ser) composta por vários itens, cada um deles diz respeito a um produto faturado. Os itens têm atributos próprios (quantidade, preço unitário, descrição do produto, etc.) e entidade própria (é possível distinguir um item de outro na mesma fatura). Uma possível representação deste domínio é o diagrama abaixo. Uma fatura possui vários itens, mas um item apenas diz respeito a uma fatura. Uma agregação tem um losango fechado apontando para a parte contendo o todo.



É importante observar que os itens apenas existem enquanto existir a fatura da qual fazem parte. Dito de outra forma, uma fatura é uma agregação, ou seja uma composição de itens. Se removermos uma fatura, os itens dessa fatura também serão removidos. Neste caso temos um tipo de agregação conhecido como **Composição**. Uma composição representa uma relação mais forte entre o objeto agregador e os objetos componentes.

- v. **Generalização de Classes:** uma generalização define uma herança, tal que uma classe refina, isto é, especializa detalhes sobre a classe mais geral. A classe generalizada é frequentemente chamada de *superclasse*, e a classe especializada *subclasse*. Todos os atributos e operações da classe generalizada que tem visibilidade pública e protegida, estão disponíveis para a

subclasse. Uma generalização tem um triângulo apontando para a superclasse.



- vi. **Dependência entre Classes:** a dependência é um relacionamento no qual a mudança de uma classe pode afetar o comportamento ou estado de outra classe. Tipicamente, dependências são usadas no contexto de classes para mostrar que uma classe usa outra como um argumento na assinatura de uma operação. Ou seja, a dependência indica que um objeto depende da especificação de outro objeto.



Podemos dizer que Psiquiatra depende de Paciente, por dois motivos. Primeiro, o método `examine()` de Psiquiatra recebe um Paciente como argumento. Segundo, a operação `examinar()` chama a operação `consultarPaciente()` de Paciente. Se o nome ou lista de argumentos da operação `consultarPaciente()` mudar, você precisará atualizar o modo como Psiquiatra chama a operação. Do mesmo modo, se o nome da classe Paciente mudar, você terá de atualizar a lista de argumentos da operação `examinar()`.

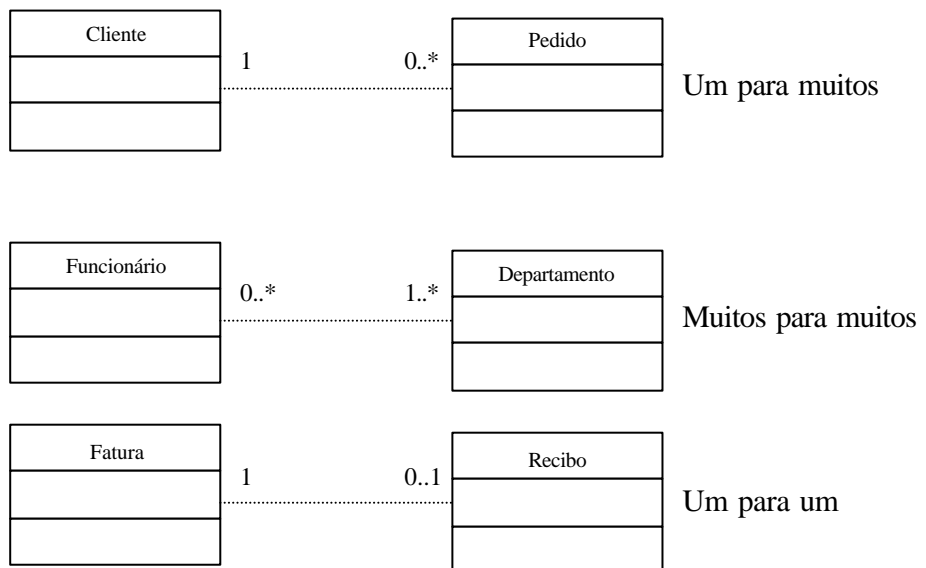
c. Multiplicidades

A multiplicidade de uma associação é o número de instâncias possíveis da classe associada com uma única instância da outra. Ou seja, número de objetos de uma classe

relacionada com um único objeto de outra. Multiplicidades são números simples ou intervalos de números. A tabela abaixo exemplifica os tipos comuns de multiplicidades.

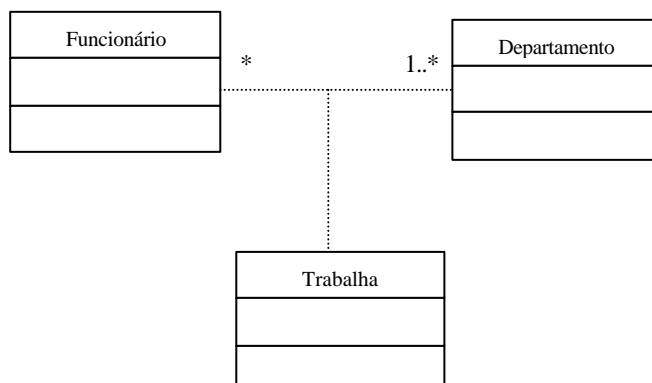
Multiplicidade	Descrição
0..1	Zero ou mais instâncias. A notação $n..m$ indica de n á m instâncias.
0..*	Nenhum limite no número de instâncias (incluindo nenhuma instância).
1	Exatamente uma instância.
1..*	Pelo menos uma instância.

No nosso exemplo abaixo, pode existir somente um único Cliente para cada Pedido, mas um Cliente pode ter um número qualquer de Pedidos.



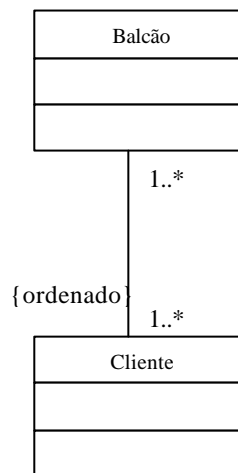
d. Classes Associativas

A associação pode ser suficientemente complexa para ela própria ser traduzida por uma classe (Classe Associativa).



Neste exemplo, considera-se que um Funcionário pode estar atuando a vários Departamentos. As associações também têm atributos, nessas situações (Classes Associativas) cria-se uma classe (unida á associação por uma reta tracejada) onde se colocam os respectivos atributos. Existem situações onde, mesmo que a associação não possua atributos próprios, é necessário criar a classe associativa (nomeadamente quando se pretende representar uma associação entre uma associação e uma classe).

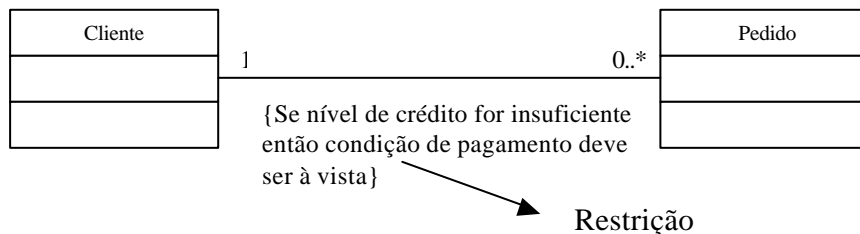
e. Associações Ordenadas



{ordenado} indica que os objetos da classe Cliente estão ordenados segundo uma determinada ordem.

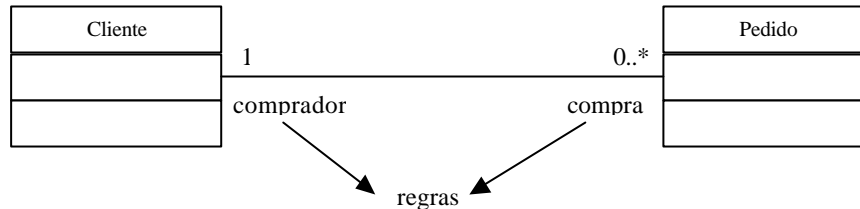
f. Restrições

Uma restrição é um relacionamento semântico entre elementos de um modelo que especifica condições e proposições que devem ser verdadeiras.



g. Estabelecendo Regras

Uma regra qualifica como um objeto atuará em seu relacionamento com outra classe. As regras são opcionais, mas às vezes elas podem melhorar a legibilidade de um diagrama de classes. Uma regra é posicionada próxima a classe que esta relacionada. No caso da Virtual LTDA, Pedido poderia conter a regra *compra*, enquanto Cliente poderia conter a regra comprador.

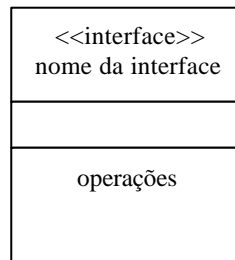


h. Interface

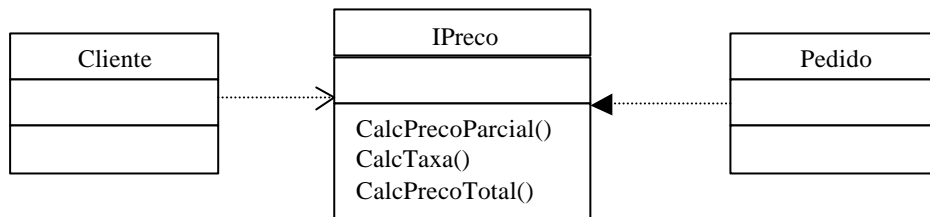
Uma interface é uma coleção de operações que são usadas para especificar um serviço de uma classe. Gráficamente, uma interface é representada como um círculo. Toda interface deve ter um nome que a distingue de outras interfaces.



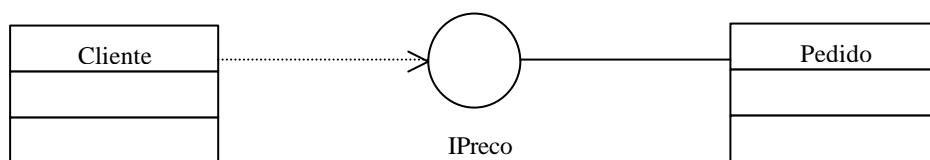
Uma interface pode também ser representada como um estereótipo. Neste caso, é possível visualizar as operações permitidas na interface.



No caso da Virtual LTDA, podemos definir uma interface a qual nomearemos de *IPreco* (a letra 'I' foi usada para permitir identificar a classe como sendo uma classe de interface). A interface *IPreco* prove um contrato de definição de preço para a classe Pedido. A classe pedido, a qual realiza, isto é, prove a implementação concreta da interface *IPreco* contém um relacionamento, chamado realização. Este relacionamento é uma combinação da notação de generalização ou herança e o relacionamento de dependência. Note que a classe Cliente depende da interface *IPreco*.

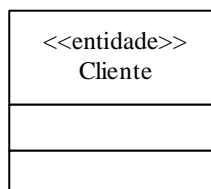


Abaixo é mostrado a mesma modelagem anterior. Note que nesta modelagem as operações da interface IPreco não são mostradas.



i. Estereótipos

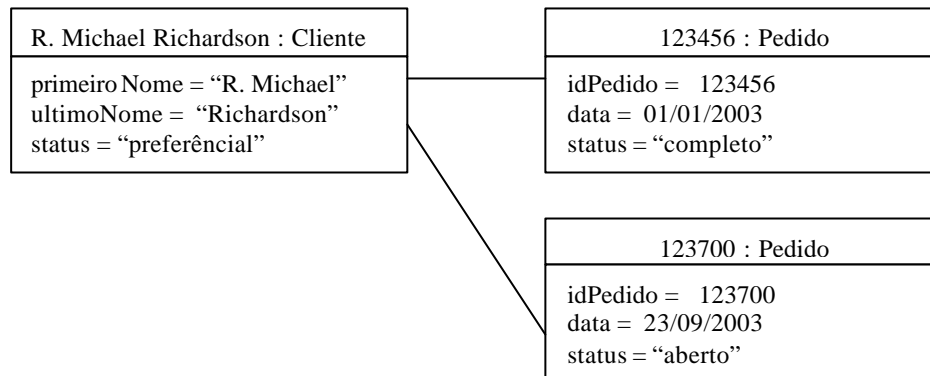
É um elemento da UML que permite estender o vocabulário da própria linguagem UML ou classificar uma marcação. Um estereótipo consiste em uma palavra ou frase incluída entre <<>>. A notação usada pela UML para estereótipos, dentro da representação gráfica da classe de objeto, é coloca-lo entre <<>> na área reservada para o nome da classe e acima deste, como mostrado abaixo.



O exemplo acima especifica que a classe Cliente possui o estereótipo <<entidade>>, isto é, a classe Cliente é um tipo de classe de entidade.

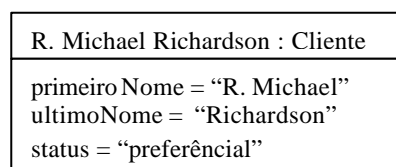
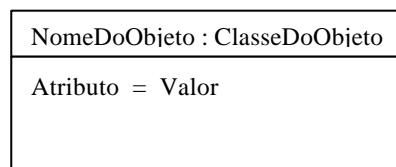
3. Diagrama de Objetos

O diagrama de objetos modela as instâncias das classes contidas no diagrama de classes, isto é, o diagrama de objetos mostra um conjunto de objetos e seus relacionamentos no tempo. Estes diagramas são importantes para construir os aspectos estáticos do sistema. Normalmente, são compostos por: objetos e relacionamentos.

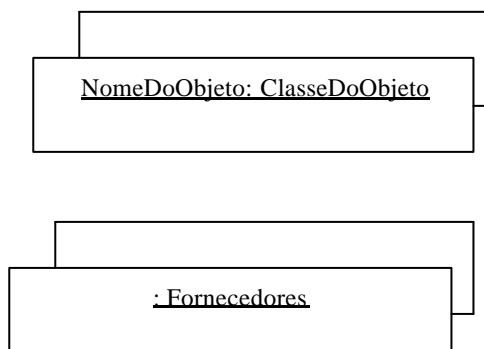


O exemplo acima mostra um diagrama de objetos para o cliente Michael Richardson e seus dois pedidos na Virtual LTDA. O diagrama pode ser lido da seguinte maneira: O objeto R. Michael Richardson da classe Cliente está associado a ambos os objetos 123456 e 123700 da classe Pedido.

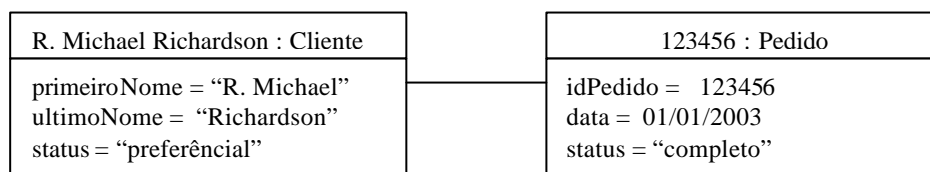
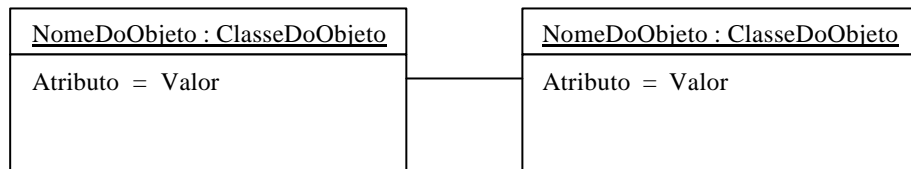
- a. **Objetos:** cada objeto é representado como um retângulo, o qual contém o nome do objeto e de sua classe sublinhados e separados por dois pontos (:). Os objetos podem conter também seus atributos. Entretanto, diferente das classes, os atributos dos objetos devem ter valores associados a eles.



Quando os atributos de objetos individuais não são importantes, é possível modelar múltiplos objetos usando a notação abaixo.



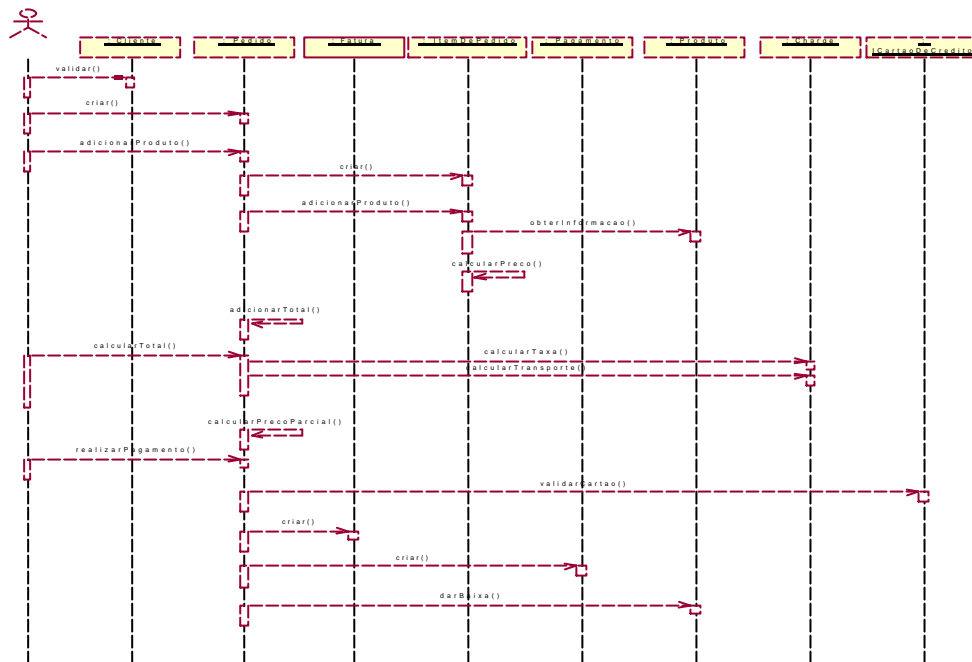
- b. **Relacionamentos:** é representado por uma linha que conecta dois objetos.



No exemplo acima temos uma relação entre o objeto R. Michael Richardson e o objeto 123456.

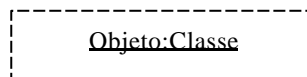
4. Diagrama de Seqüências

O diagrama de seqüência descreve a interação entre classes em termos de uma troca de mensagens sobre o tempo. Um diagrama de seqüência é composto por: papeis da classe, Barras de ativação, mensagens e linhas de vida. O diagrama de seqüência abaixo foi modelado para o caso de uso Processar Pedido da Virtual LTDA.

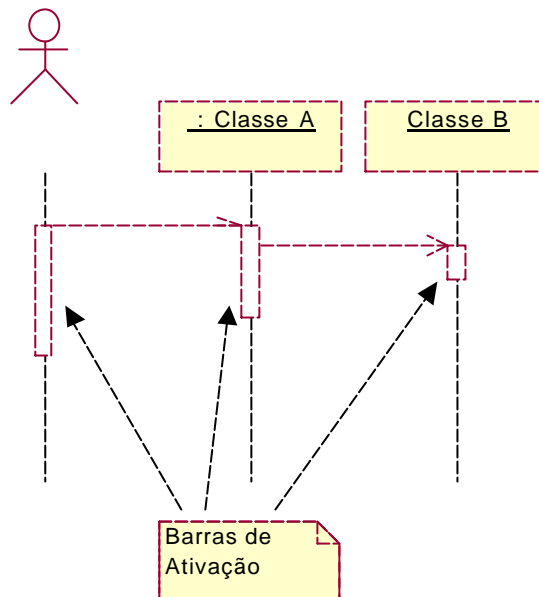


Os casos de usos, como vimos representam um conjunto de cenários que descrevem os diferentes processos que ocorreram no sistema. O diagrama de seqüência de eventos permite modelar estes processos através da troca de mensagens (eventos) entre os objetos do sistema. Cada mensagem no diagrama de seqüência de eventos corresponde a uma operação no diagrama de classes. Como as mensagens são operações invocadas, elas devem estar presentes nos objetos de destino, que são ativadas pelas mensagens no objeto de origem.

- a. **Papeis da Classe:** descreve maneira que um objeto se comportará no contexto. Usa o símbolo de objeto da UML para ilustrar papeis da classe, mas não lista atributos do objeto.

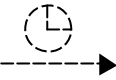


- b. **Barras de ativação:** caixas de ativação representam o tempo que um objeto necessita para completar uma tarefa.



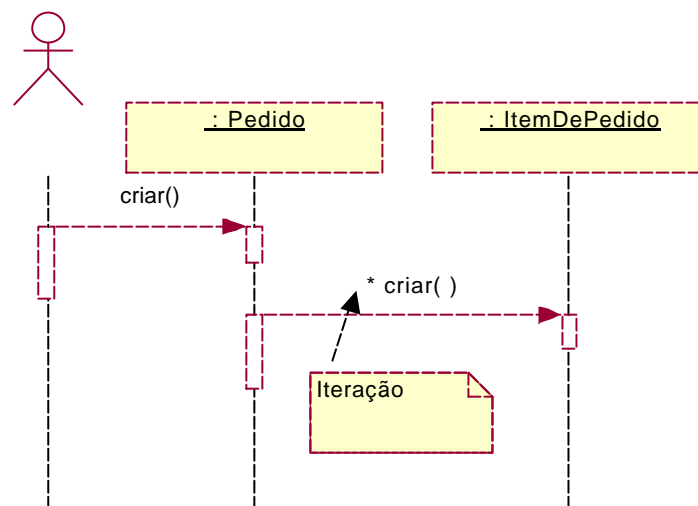
- c. **Mensagens:** são setas que representam a comunicação entre objetos. Estas setas são rotuladas com o nome da mensagem.

Seta	Tipo de Mensagem	Descrição
----->	Simples	Indica o envio de uma nova mensagem. Pode ser síncrona ou assíncrona.
----->=	Retorno	Indica o retorno do controle após uma nova mensagem ter sido enviada.
----->	Síncrona	Envio de mensagens síncronas.
-----\	Assíncronas	É uma mensagem que não bloqueia o emissor. Ou seja o emissor e o receptor executam concorrentemente.
-----<=	Balking	Também conhecida como autodelegação, é usada quando um

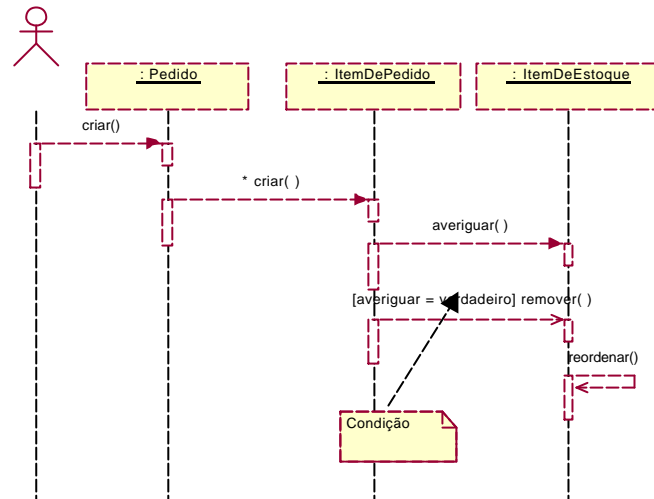
		objeto necessita enviar uma mensagem para ele mesmo.
	Time Out	Indica que a mensagem enviada tem um tempo de vida.

Ainda sobre as mensagens podemos considerar:

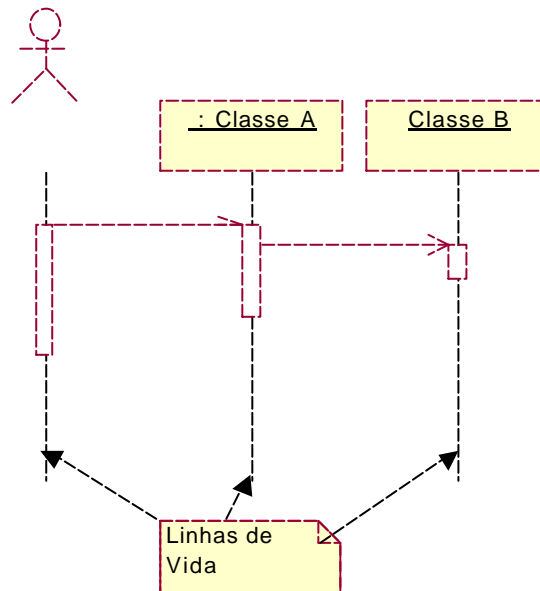
- **Iteração:** indica que a mensagem é enviada várias vezes para múltiplos usuários. A iteração é representada por um asterisco (*) antes da mensagem.



- **Condição:** reporta se uma mensagem é enviada ou não. A condição geralmente é localizada no lado esquerdo da mensagem e entre colchetes [condição] .

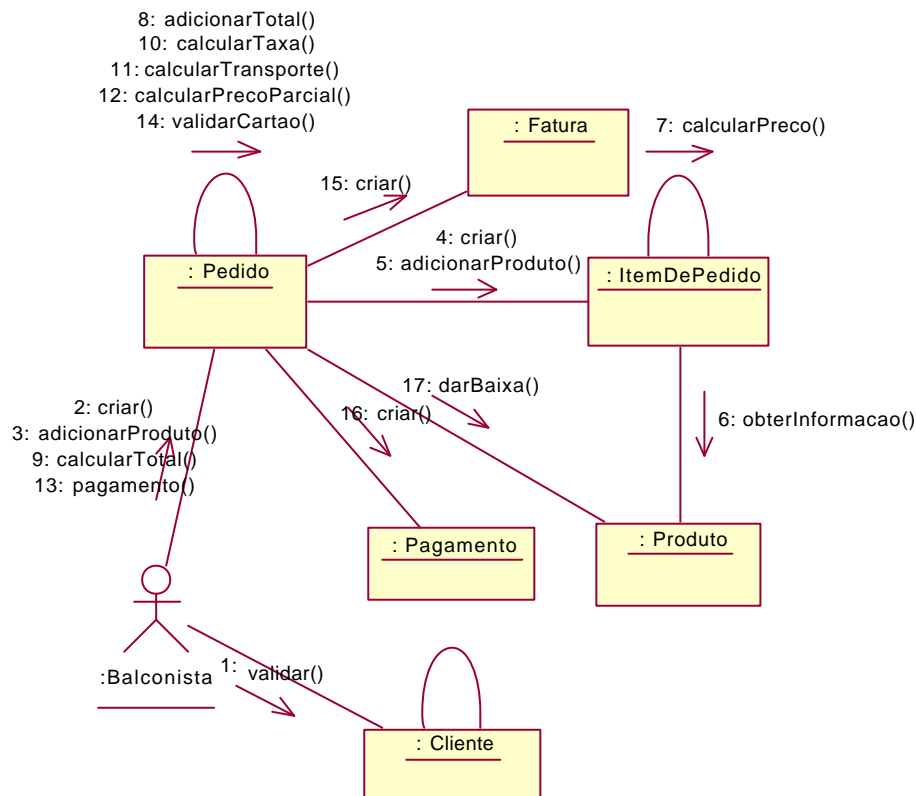


- d. **Linhas de Vida:** são linhas verticais tracejadas que indicam a presença de objeto sobre o tempo. Ou seja, uma linha de vida representa o tempo de vida do objeto.



5. Diagrama de Colaboração

Um diagrama de colaboração descreve as interações entre objetos em termos de mensagens seqüenciais. Isto é, os diagramas de informação representam uma combinação de informação obtida a partir dos diagramas de classes, seqüência e caso de uso, descrevendo ambos a estrutura estática e o comportamento dinâmico de um sistema. Os símbolos usados no diagrama de colaboração são: papéis da classe, papel da associação e mensagens. O diagrama de colaboração deve ser usado quando se deseja modelar os relacionamentos entre atores e o sistema. Abaixo é mostrado o diagrama de colaboração, para o caso de uso Processar Pedido da Virtual LTDA.

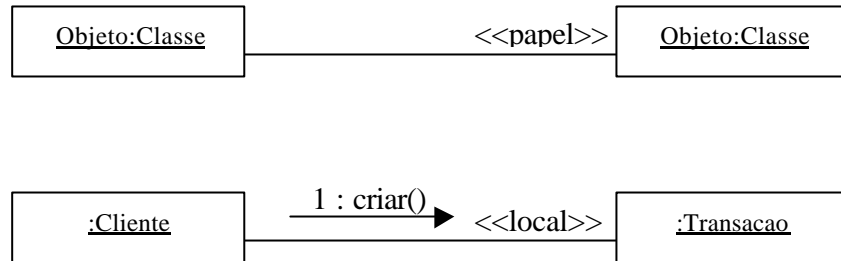


- a. **Papeis da Classe:** descreve como os objetos se comportam. Usa-se o símbolo de objetos da UML para representar os papéis da classe, mas não lista os atributos dos objetos.

Objeto:Classe

:Cliente

- b. **Papel da Associação:** descreve como uma associação comportará dada uma situação particular. O papel da associação é representado usando uma linha rotulada com um estereótipo.



O exemplo acima indica que o objeto Transação esta local ao Cliente.

- c. **Mensagens:** contrário ao diagrama de seqüência , o diagrama de colaboração não tem um modo explicito para denotar tempo, ao invés disso usa mensagens numeradas na ordem de execução.

1.1 Nome da mensagem →

Ainda sobre as mensagens podemos considerar:

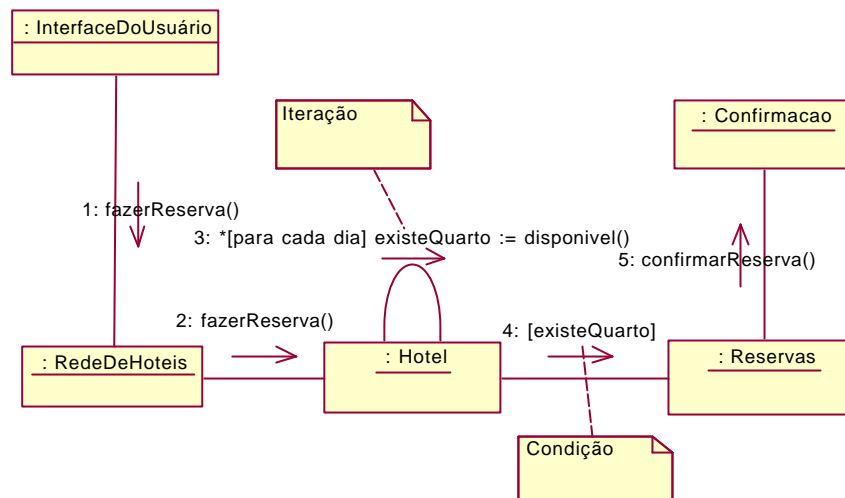
- **Iteração:** é representado por um asterisco depois do número de seqüência para indicar uma repetição.

1.1 * Nome da mensagem →

1.2 * [expressão] Nome da mensagem →

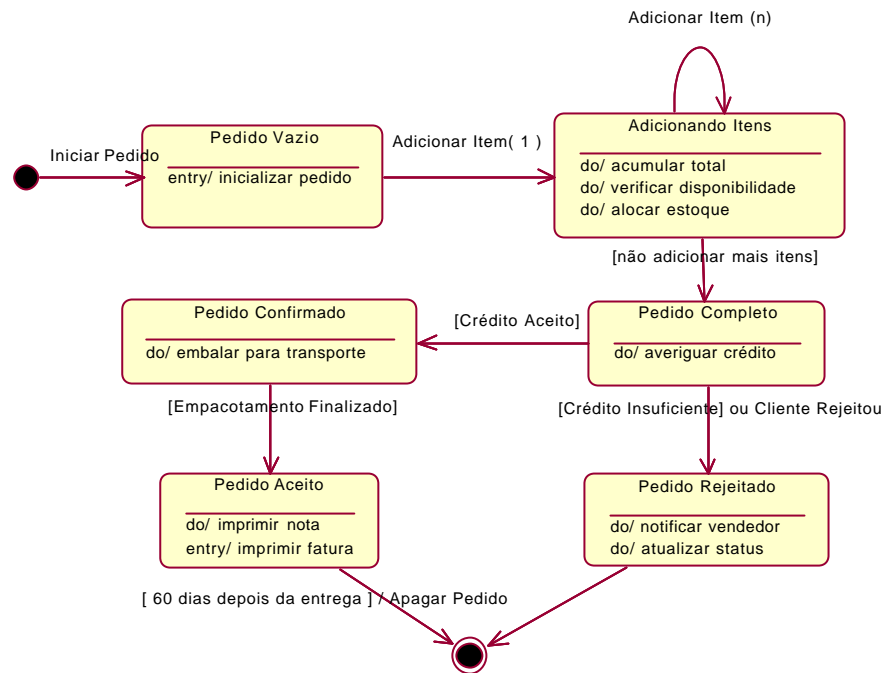
- **Condição:** a condição para uma mensagem é normalmente posicionada entre colchetes [condição] e após o número de seqüência da mensagem.

1.3 [condição] Nome da mensagem →

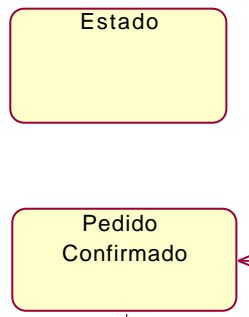


6. Diagrama de Estados

Os diagramas de transição de estados mostram a dinâmica interna de uma classe. Apenas os eventos e estados de uma única classe são representados neste diagrama. Entende-se por eventos os fatos que ocorreram em uma classe, provocados por elementos externos (mensagens) ou internos como condições internas da classe que provocam uma troca de estado. Uma classe pode ter vários estados, caracterizados por situações em que a classe se encontra. O diagrama de estados pode possuir ainda estados especiais como o estado inicial e o estado final e outros estados de controle internos. Um diagrama de estados é composto normalmente, pelos seguintes itens: estados, transição, estado inicial e estado final. Abaixo é mostrado o diagrama de estados para a classe Pedido da Virtual LTDA. Note que após ter sido adicionado um item “adicionar item (1)” o objeto Pedido move-se para o estado “Adicionando Itens”.



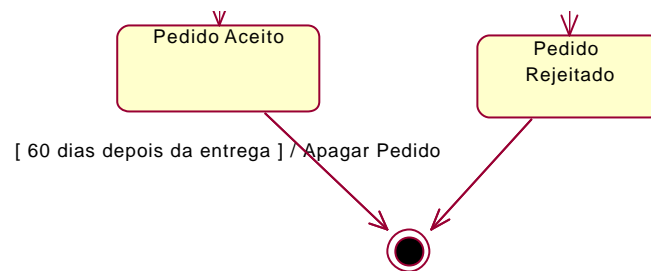
- a. **Estado**: estados representam, situações durante a vida de um objeto. Um estado é representado por um retângulo com as esquinas arredondadas.



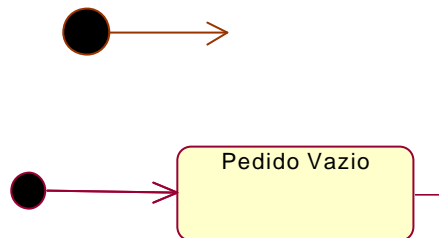
O exemplo acima mostra um dos estados assumidos pelo objeto Pedido.

- b. **Transição**: uma transição é representada por uma seta que indica o caminho entre os diferentes estados de um objeto. A transição é rotulada com a identificação do evento e a ação a ser executada. Um objeto no estado original irá realizar um conjunto de ações e entrar no estado destino quando um evento especificado ocorrer ou quando certas condições forem satisfeitas.

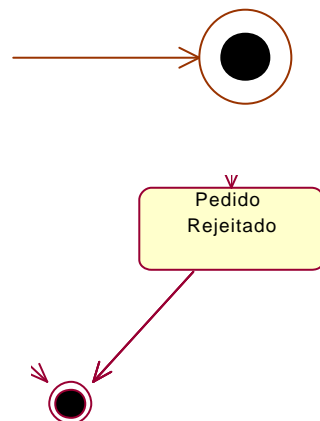




- c. **Estado Inicial:** um círculo preenchido seguido por uma seta representa o estado inicial do objeto. Um objeto deve ter um único estado inicial.



- d. **Estado Final:** uma seta apontando para um círculo com um outro círculo interno e preenchido representa o estado final de um objeto. Um objeto pode ter vários estados finais.



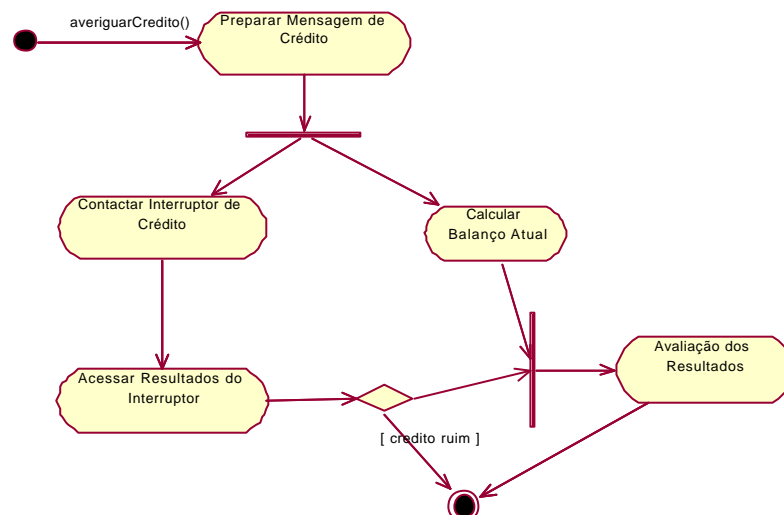
- e. **Ações de um Estado:**

- i. **On Entry:** executada quando o objeto entra no estado ou na atividade.
- ii. **On Exit:** executada quando o objeto estiver naquele estado ou atividade.

- iii. **On Do:** executada enquanto o objeto estiver naquele estado ou atividade.
- iv. **On Event:** executada apenas quando um determinado evento é recebido.

7. Diagrama de Atividades

Um diagrama de atividade ilustra a natureza dinâmica de um sistema pela modelagem do fluxo de controle de atividade à atividade. Uma atividade representa uma operação em alguma classe no sistema que resulta em uma mudança no estado do sistema. Tipicamente, diagramas de atividades são usados para modelar fluxo de processos ou processos de negócios e operações internas. O diagrama de atividades é um tipo especial de diagrama de estados.

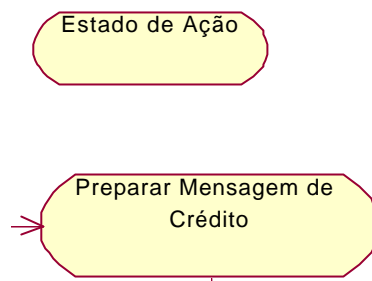


O diagrama acima, mostra o diagrama de atividades para a operação `averiguarCredito` na classe `Pedido` da `Virtual LTDA`. Note que a atividade “Preparar Mensagem de Credito” informa nos o que fazer, mas não como fazer.

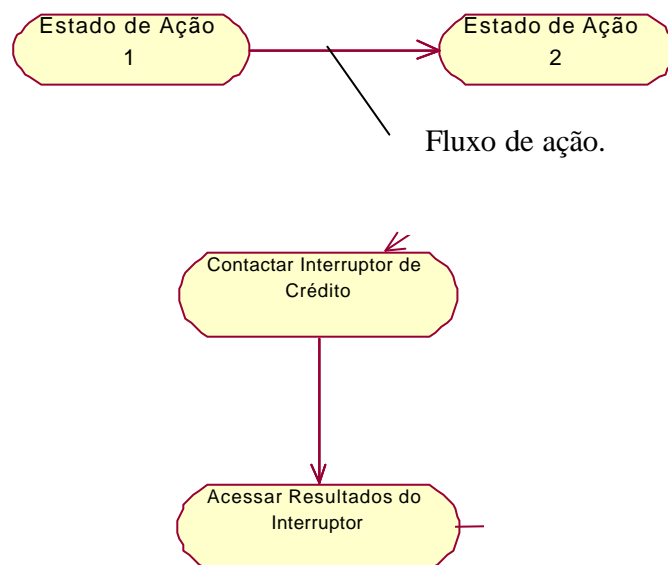
Um diagrama de atividades é normalmente composto pelos seguintes símbolos: estados de ação, fluxo de ação, fluxo de objeto, estado inicial, estado final, branching, sincronização, raias.

- a. **Estados de Ação:** No fluxo de controle modelado por um diagrama de atividades, coisas acontecem. Podemos avaliar algumas expressões que atualiza o valor de um atributo ou que retorna algum valor. Alternativamente, podemos chamar uma operação de um objeto, enviar sinal para um objeto, ou até mesmo criar ou destruir um objeto. Essas computações atômicas são chamadas Estados de Ação, pois eles são estados de um sistema, cada um representando a execução de uma ação.

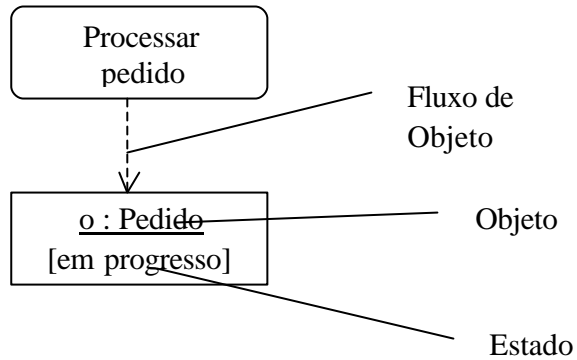
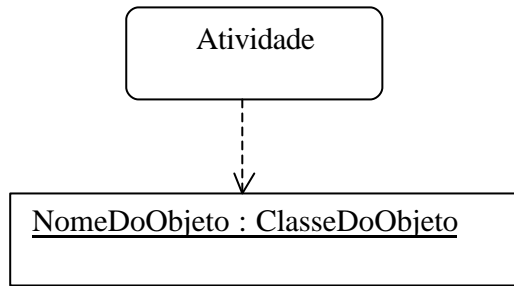
Estados da ação é representado por um retângulo com as quinas arredondadas.



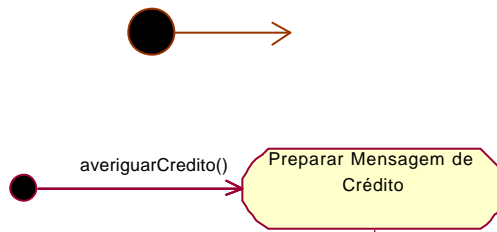
- b. **Fluxo de Ação:** Uma seta representa o relacionamento entre estados de ação.



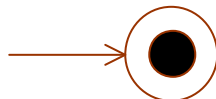
- c. **Fluxo de Objeto:** fluxo de objetos refere-se a criação e modificação de objetos pelas atividades. Um fluxo de objeto é representado por uma seta tracejada, a partir de uma ação para um objeto, significando que a ação cria ou influencia o objeto. Caso contrário, indica que o estado de ação usa o objeto.

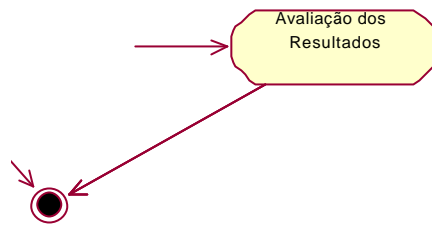


- d. **Estado Inicial:** um círculo preenchido seguido por um seta representa o início de um atividade.

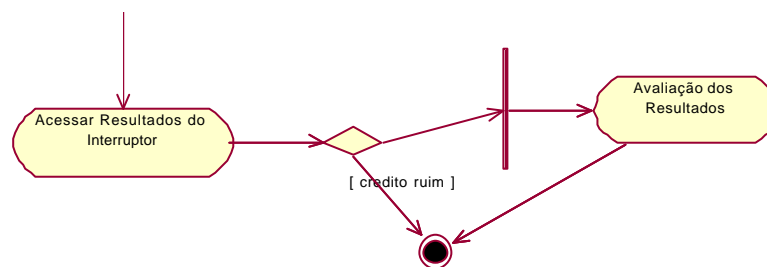
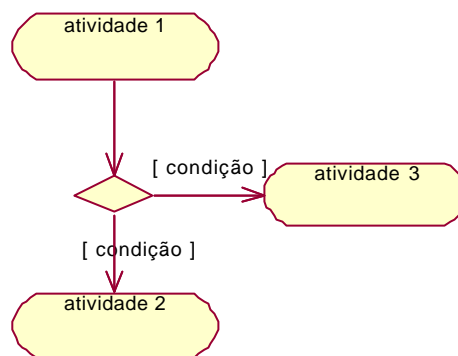


- e. **Estado Final:** uma seta apontando para um círculo que inclui um outro círculo interno preenchido representa o estado final de uma atividade.

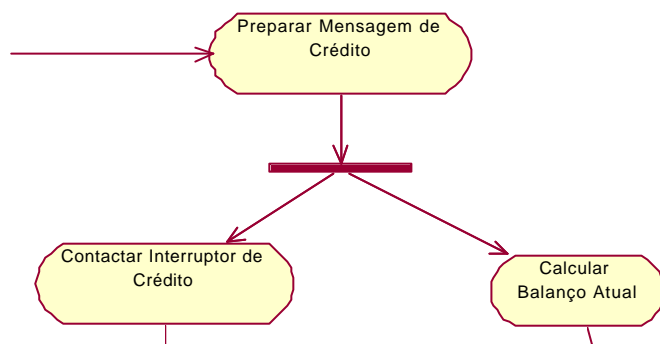
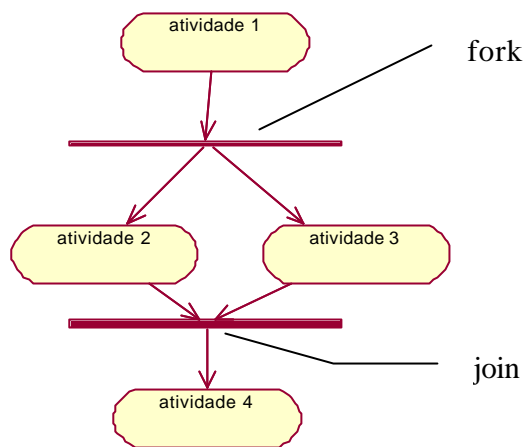




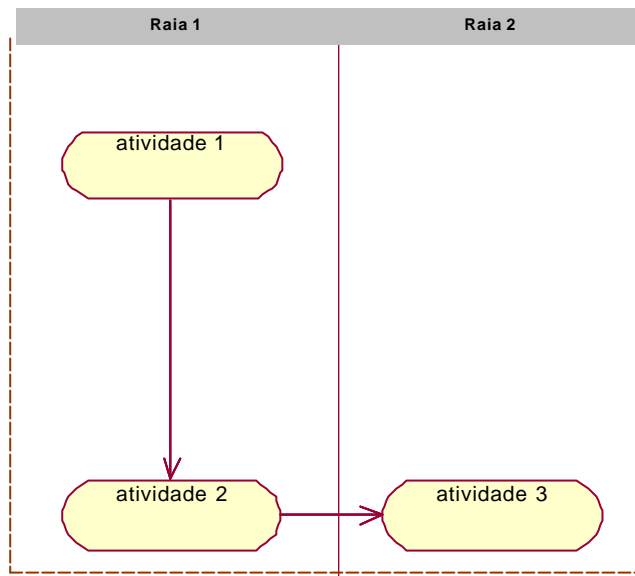
- f. **Branching:** um losango representa uma decisão com caminhos alternados. A alternativa de saída deve ser rotulada com uma condição ou expressão. Pode-se também rotular apenas um dos caminhos.



- g. **Sincronização:** Uma barra de transição ajuda a ilustrar as transições paralelas. Sincronização também é chamada de forking e joining.

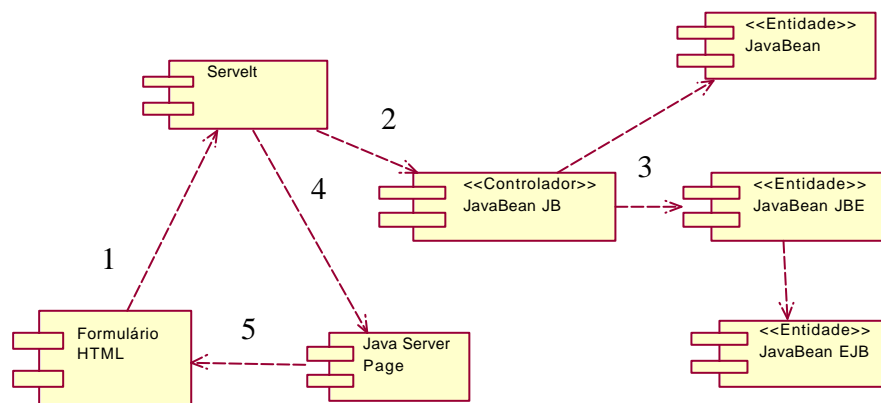


- h. Raias:** diagramas de atividades podem ser divididos em grupos e posicionados dentro de raias que determinam qual objeto é responsável por qual atividade.



8. Diagrama de Componentes

Diagrama físico que mostra os vários componentes em um sistema e suas dependências. Similar a um pacote, mas com um enfoque ao empacotamento físico do código. Basicamente, um diagrama de componentes é composto por: componente, interface e dependências. Abaixo é proposto um diagrama de componentes para a Virtual LTDA.



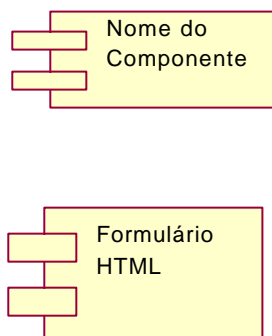
Os estágios do diagrama acima são:

- Passo 1: o formulário HTML requisita um recurso. Esse pedido de recurso é mapeado para um Servlet.
- Passo 2 : o Servlet instancia uma classe de controle que é um JavaBean.
- Passo 3 : A missão do controlador Bean é implementar o caminho para o caso de uso. Existirá uma classe de controle para cada caso de uso. Por

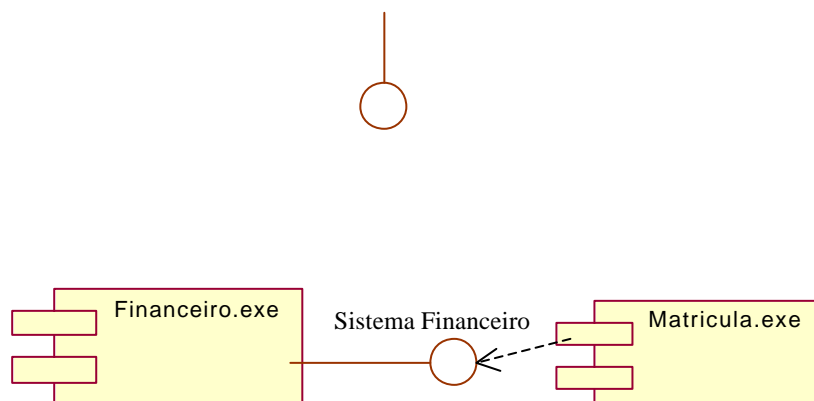
exemplo, no caso da Virtual LTDA, existirá um controlador chamado ControladorMantemRelacionamento que contém uma operação chamada investigarCliente(). A missão dessa operação é enviar mensagens para outros Beans.

- Passo 4 : O Servlet então transmite a requisição ao Java Server Page apropriado.
- Passo 5 : O Java Server Page, usando os objetos recentemente posicionados no escopo de requisição do Servlet, formata uma página de retorno HTML como resposta ao browser.

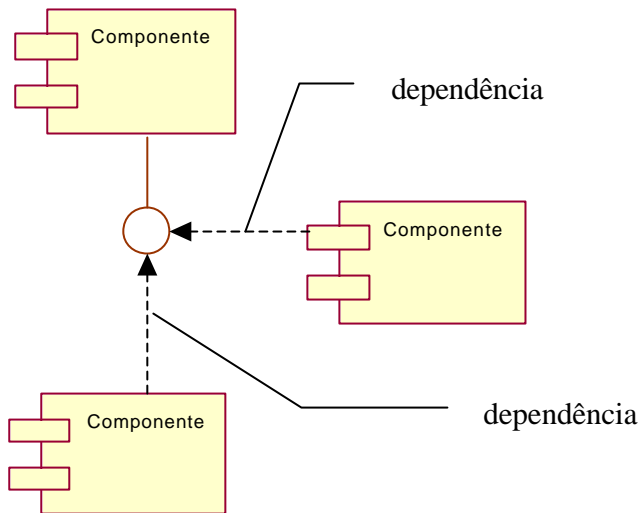
- a. **Componente:** um componente é um bloco de construção físico do sistema. Pode ser um arquivo executável, arquivos com classes Java, biblioteca estáticas ou DLL's.



- b. **Interface:** Uma interface descreve um grupo de operações usadas ou criadas pelos componentes. Um componente realiza um conjunto de interfaces (operações públicas que podem ser chamadas por outros componentes). Uma interface é representada como mostrado abaixo.



- c. **Dependências:** A dependência entre componentes é representada usando setas tracejadas.

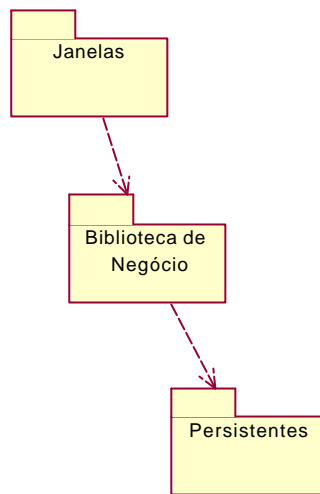


9. Diagrama de Distribuição

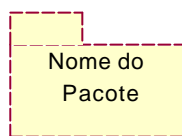
Os diagramas de distribuição mostram a distribuição de hardware do sistema, identificando os servidores como nós do diagrama e a rede que relaciona os nós. Os componentes de software vão estar mapeados nestes nós. O diagrama de distribuição é composto por: nodo, associação, componentes e nodos. Abaixo é mostrada uma proposta para o diagrama de distribuição da Virtual LTDA.

10. Diagrama de Pacotes

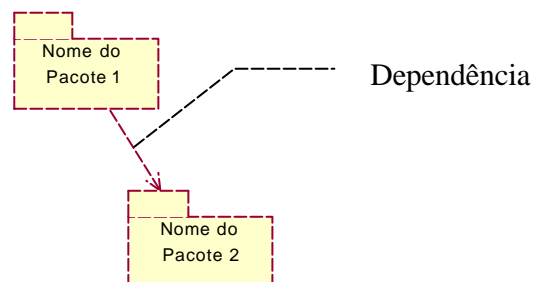
Em muitos casos um único diagrama de classes pode ser exageradamente grande para representar todo o sistema. Assim é conveniente utilizar um elemento para organizar os subsistemas do modelo. Para isto utiliza-se o diagrama de pacotes. Um diagrama de pacotes pode ser utilizado em qualquer fase do processo de modelagem e visa organizar os modelos. O diagrama de pacotes abaixo é uma proposta para a Virtual LTDA.

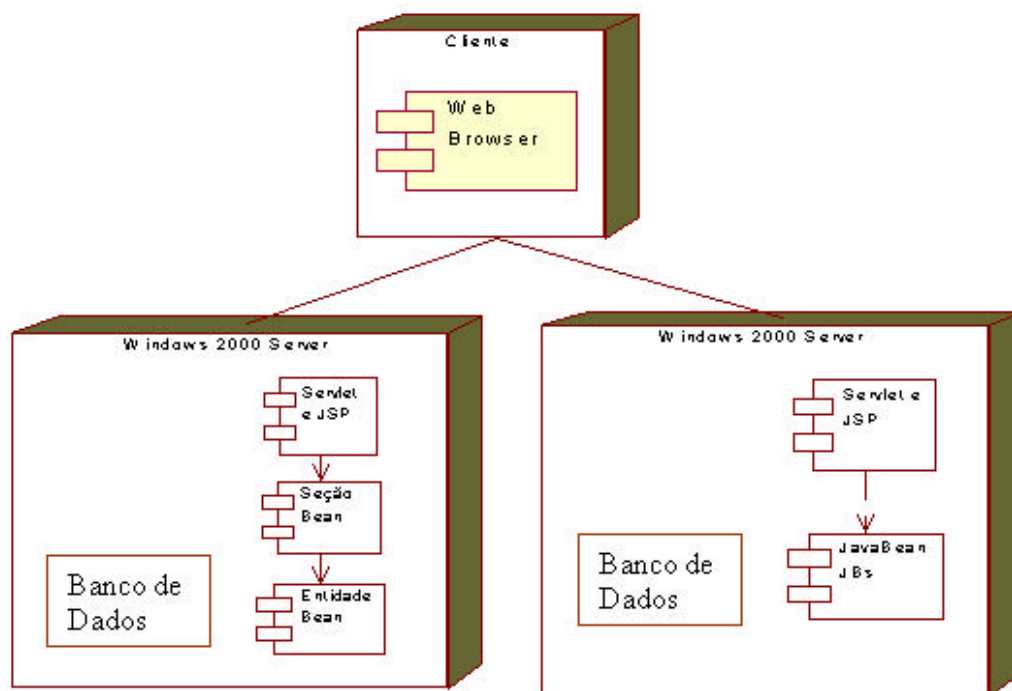


- a. **Pacote:** Um pacote representa um grupo de classes (ou outros elementos) que se relacionam com outros pacotes através de uma relação de dependência.

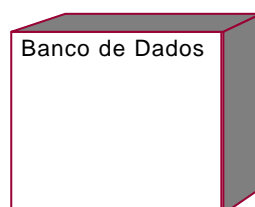
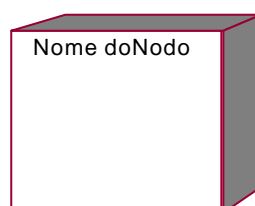


- b. **Dependência entre Pacotes:** define um relacionamento em que mudanças em um pacote afetarão o outro pacote.

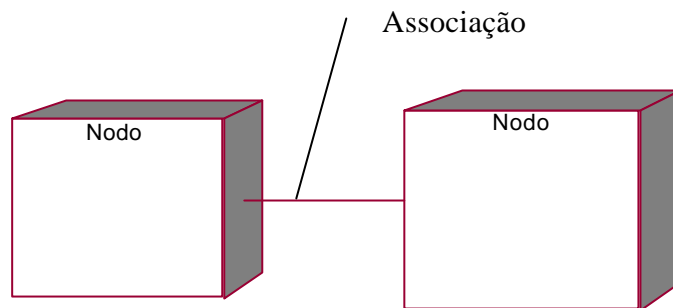




c. **Nodo:** um nodo é um recurso físico que executa componentes.



d. **Associação:** Refere-se a uma conexão física entre nodos, tal como a Internet.



- e. **Componentes e Nodos:** Posiciona componentes dentro dos nodo que distribui eles.

