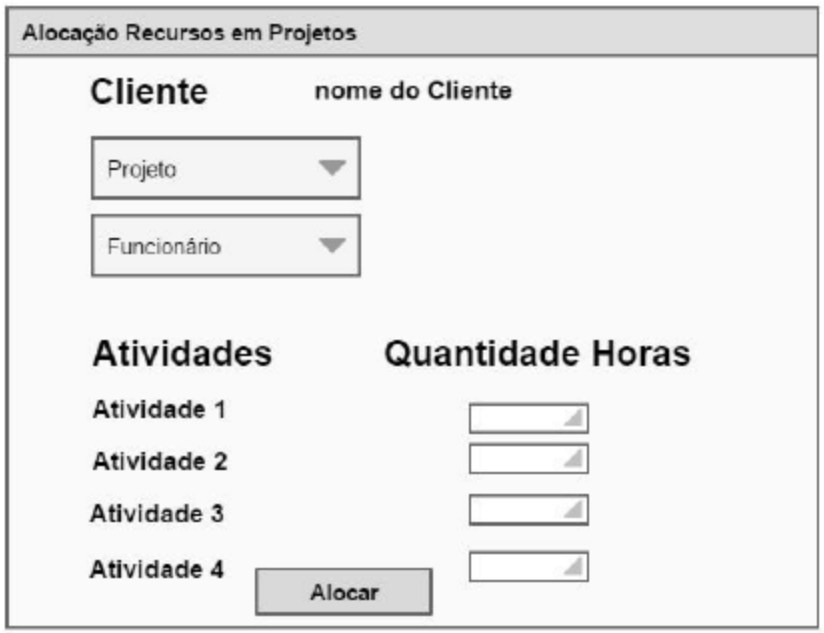
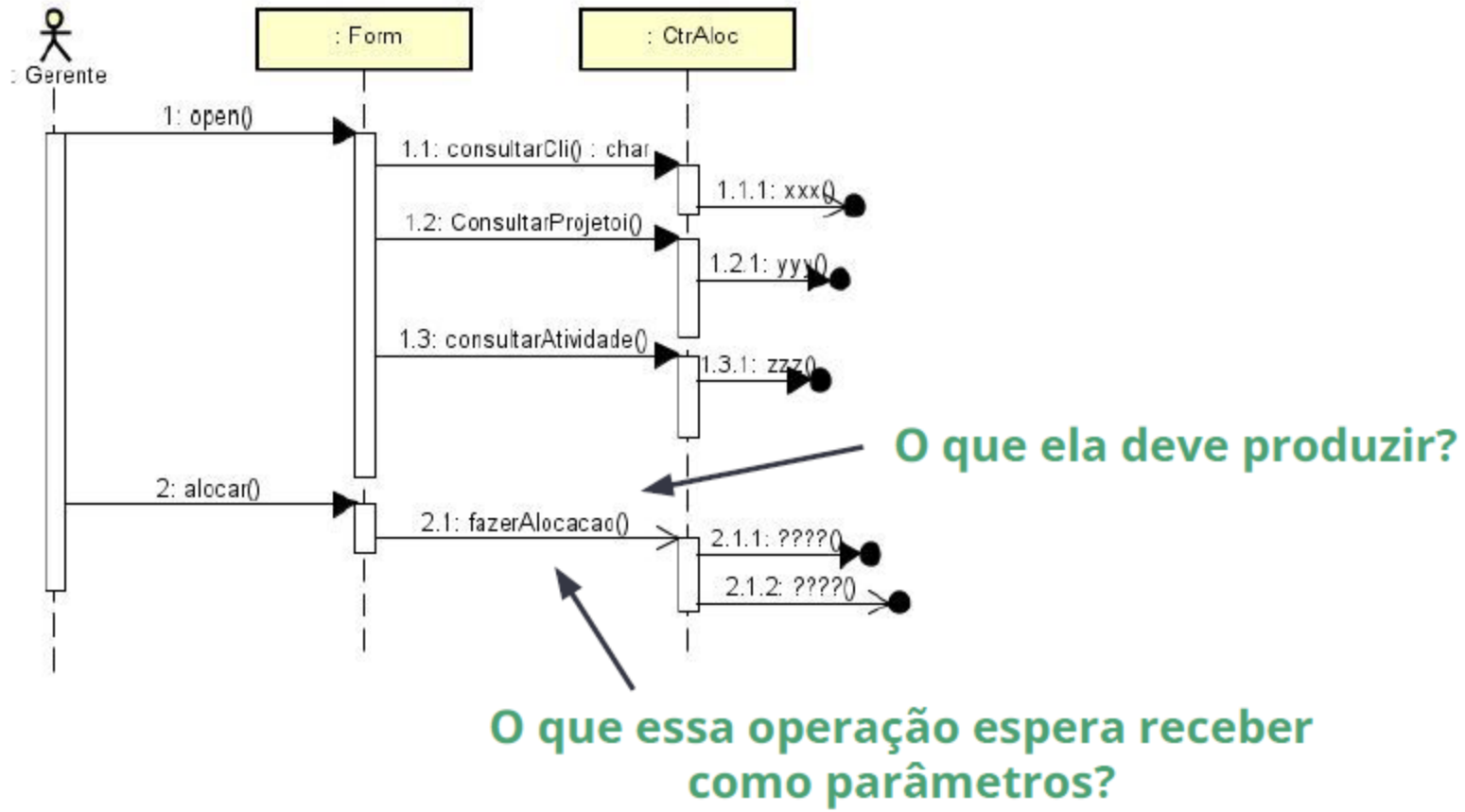
Modelagem Funcional com Contratos

João Pedro Oliveira Batisteli

Responsabilidades de Operações





O que ela deve produzir?

O que essa operação espera receber como parâmetros?

Como gerar informações para testes de unidade?

Contratos em Modelagem de Sistemas

Origem dos Contratos

- Durante a **modelagem funcional**, são produzidos artefatos que descrevem o comportamento do sistema:
  - **Modelo conceitual** (estrutura de classes e relacionamentos)
  - **Casos de uso expandidos** ou **diagramas de sequência**, que detalham os cenários

Identificação de Operações

- Ao construir os **diagramas de sequência**, identificam-se os **comandos e consultas** que o sistema precisa oferecer.
- Cada uma dessas ações reflete uma **intenção do usuário,** algo que ele quer realizar ou obter do sistema.

Contratos em Modelagem de Sistemas

Definição dos Contratos

- Essa **intenção do usuário** é formalizada por meio dos **Contratos de Operação** (ou **Contratos de Consulta**).

- Eles descrevem, para cada operação:
  - O **estado do sistema antes** da execução (pré-condição);
  - O **efeito produzido** após a execução (pós-condição);
  - E as **classes e atributos** afetados.

Contratos de Operação

Por que usar contratos?

- Os **casos de uso** são a principal forma de **descrever o comportamento do sistema**.
- No entanto, em certos momentos, é necessário um **nível maior de precisão,** especialmente sobre **como o sistema muda internamente** após uma operação.

Contratos de Operação

Por que usar contratos?

- Os **casos de uso** são a principal forma de **descrever o comportamento do sistema**.
- No entanto, em certos momentos, é necessário um **nível maior de precisão,** especialmente sobre **como o sistema muda internamente** após uma operação.

E os diagramas de sequência? Eles já não fazem isso?

Contratos de Operação

O que são contratos de operação

- Cada contrato descreve, em **termos de pré e pós-condição**, as **modificações que uma operação causa** nos **objetos do modelo de domínio**.
- Eles tornam explícito **o efeito interno** das ações do sistema, o que muda, o que é criado, e o que é consultado.

Exemplo simplificado:

Operação: registrarPedido(item, quantidade)
Pré-condição: cliente deve estar autenticado
Pós-condição: novo pedido é criado e adicionado à lista de pedidos do cliente

Contratos de Operação

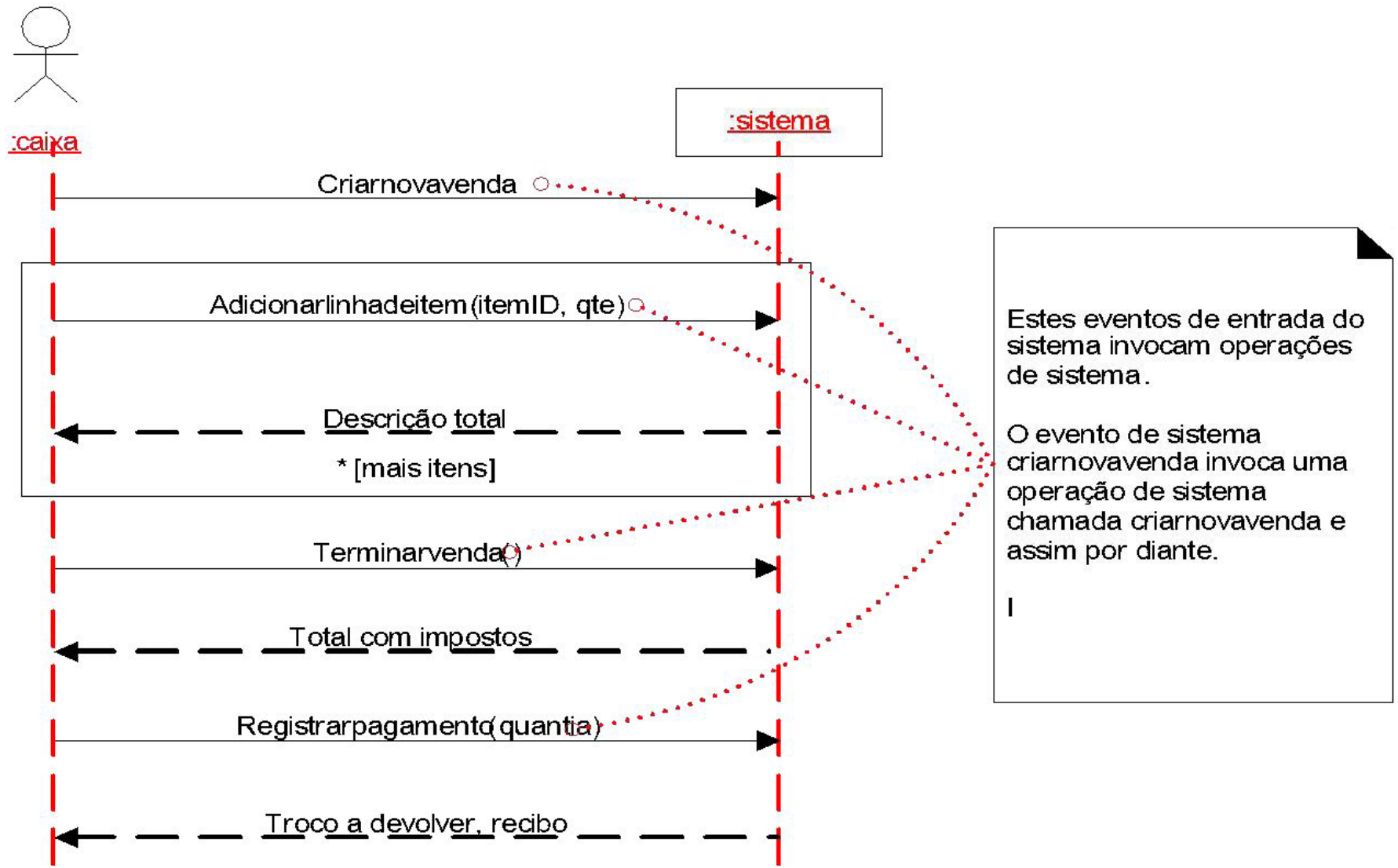
Relação com os Casos de Uso

- Os **contratos complementam** os casos de uso, oferecendo **detalhes analíticos** sobre o **efeito das operações do sistema**.
- Assim, podem ser vistos como **uma extensão do modelo de casos de uso**, refinando o **comportamento implícito** nele.

Em resumo:

- Os **contratos de operação** descrevem *como o sistema reage internamente* às interações modeladas nos casos de uso e diagramas de sequência.

Contratos de Operação



Contratos de Operação

- Um **contrato** é um documento (ou artefato de análise) que **descreve os compromissos e efeitos de uma operação** do sistema.

Características

- **Estilo declarativo:** descreve o *que acontece*, não *como acontece*.
- **Pré-condições:** o estado do sistema **antes da execução** (o que deve ser verdadeiro).
- **Pós-condições:** o estado do sistema **após a execução** (o que muda, é criado ou atualizado).
- **Exceções:** Estabelecem quais condições que poderiam evitar que o comando tivesse sucesso as quais
- serão verificadas por ela
- **Aplicação:** pode ser definido para **métodos**, **classes** ou **operações gerais do sistema**.

Propósito

- Os **contratos de operação** ajudam a:
- Definir **claramente o comportamento esperado** do sistema;
- Descrever os **resultados da execução** de uma operação;
- Tornar explícitas as **mudanças de estado** nos **objetos do domínio**

Pré Condições

- As **pré-condições** definem **o que deve ser verdadeiro** no **estado atual do sistema** (ou em suas informações armazenadas) **antes que uma operação possa ser executada**.

Características Importantes

- Não são **verificadas automaticamente** durante a execução.
- Sua **validação depende de mecanismos externos**, como:
  - Regras de interface (ex.: campos obrigatórios);
  - Controle de acesso (ex.: usuário autenticado);
  - Lógica de negócio que impede chamadas inválidas.
- Visam garantir que a operação **comece em um estado consistente.**

Tipos de Pré-Condições

- Nem todas as pré-condições têm o mesmo propósito, elas podem garantir **diferentes aspectos do estado do sistema** antes da execução da operação.

1. **Garantia de Parâmetros**
   - Verifica se os **parâmetros recebidos** pela operação **correspondem a elementos válidos** do sistema.
   - Evita a execução de operações com **dados inexistentes ou inconsistentes**.
2. **Restrição Complementar**
   - Garante que o **estado atual do sistema** esteja **em uma situação específica desejada**.
   - Evita que a operação ocorra em **contextos indevidos** ou **ordens incorretas de execução**.

Tipos de Pré-Condições - Exemplos

1. **Garantia de Parâmetros**

   Operação: *ExcluirProduto(idProduto)*
   Pré-condição: *deve existir um produto com o identificador informado.*

2. **Restrição Complementar**

   Operação: *FecharPedido()*
   Pré-condição: *o pedido deve conter pelo menos um item e estar em status "aberto".*

Pré-condição de Garantia de Parâmetros (Semântica)

- Uma **pré-condição de garantia de parâmetros** é considerada **semântica** quando sua validação **depende do significado dos dados**, e não apenas do tipo declarado.

Tipo de Verificação	O que valida	Exemplo	Onde é verificado
Sintática	Tipo ou formato dos dados	`x:InteiroPositivo`	Pela linguagem / compilador
Semântica	Significado no contexto do sistema	“x deve ser o ID de um produto existente”	Pelo modelo de domínio

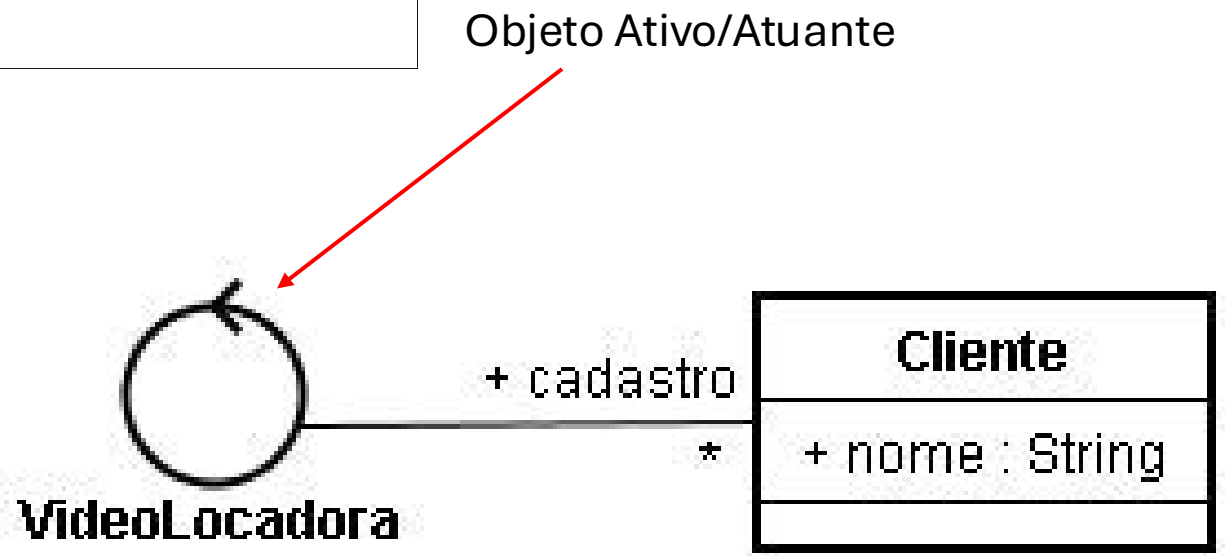
Pré-condição de Garantia de Parâmetros

Classe Videolocadora

operação: identificaCliente(nomeCliente:String)

pré:

Existe uma instância da classe *Cliente* tal que o atributo *nome* desta instância é igual ao parâmetro *nomeCliente*.



Em um contexto não ambíguo

- É possível simplificar a escrita da pré-condição

Classe Videolocadora

operação: identificaCliente(nomeCliente:String)

pré:

Existe um *Cliente* cujo *nome* é igual a *nomeCliente*.

```
self.cadastro→exists(nome=nomeCliente)
```

Pré-condições – Restrição Complementar

- As **pré-condições de restrição complementar** garantem que o **estado atual do sistema** esteja em **uma situação específica** antes da execução da operação.
  - Elas **não alteram** o modelo conceitual, apenas **complementam** suas restrições, tornando-as **mais específicas** para um cenário ou operação.

- **Exemplo**
  - O modelo conceitual define uma associação com **multiplicidade 0..1** (ou seja, um objeto pode estar associado a no máximo uma outra instância).
  - A **pré-condição complementar** pode especificar, para uma operação particular, que:

    "Neste momento, a associação **deve existir** (ou **não deve existir**)."

Pré-condições – Restrição Complementar

- Uma pré-condição **nunca deve contradizer** o modelo conceitual.
- Ela serve apenas para **especializar** ou **restringir** as situações permitidas naquele contexto específico.

Tipos de Restrições Complementares

1) Existência de Instâncias

- “Existe (ou não existe) uma instância, ou conjunto de instâncias, com determinadas propriedades.”
- **Exemplo:**
  - Deve existir um **Cliente** cujo CPF corresponda ao informado.
  - Não deve existir uma **Reserva** ativa para o mesmo exemplar.

2) Propriedade de Todas as Instâncias

- “Todas (ou nenhuma) das instâncias de uma classe possuem determinadas propriedades.”
- **Exemplo:**
  - Todos os **Filmes em estoque** devem ter status = disponível.
  - Nenhum **Cliente bloqueado** pode realizar um novo aluguel.

Tipos de Restrições Complementares

3) Existência de Associação

- “Uma associação opcional (multiplicidade 0..1 ou *) existe (ou não) entre instâncias específicas.”
- **Exemplo:**
  - A operação só pode ser executada se **não existir** uma associação entre Cliente e Filme com status “em andamento”.

4) Valor de Atributo

- “Um determinado atributo de uma instância deve ter um valor específico.”
- **Exemplo:**
  - O atributo saldo do **Cliente** deve ser maior que zero antes de registrar um novo aluguel.

Tipos de Restrições Complementares

Classe Videolocadora

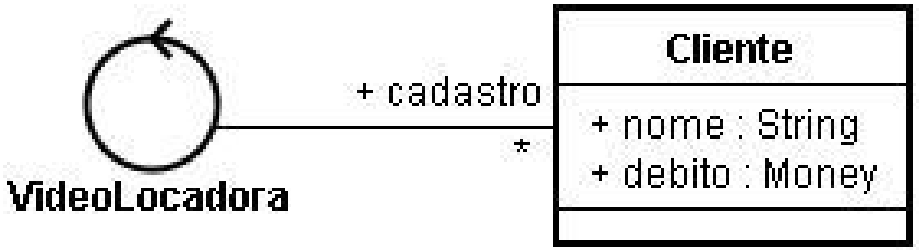
operação: identificaCliente(nomeCliente:String)

pré:

Existe um *Cliente* cujo *nome* é igual a *nomeCliente*.

Este *Cliente* possui *débito* igual a zero.

```
Alias: cliente = self.cadastro→select(nome=nomeCliente)
Pré:
   cliente→size() == 1
   cliente.debito == 0
```



Pós-condições

- As **pós-condições** descrevem o **estado do sistema após a execução** de uma operação.
- Devem ser **expressas no passado**, destacando **mudanças que já ocorreram**.
- **Características Principais:**
  - **Declarativas:** descrevem *o que mudou*, e não *como mudou*.
  - **Baseadas em objetos do modelo de domínio.**
  - **Focadas em estados**, não em ações.

Pós-condições

- **Orientação por Mudança de Estado:**
  - Pós-condições são **declarações sobre resultados**, e **não instruções de execução**.
  - Elas **não representam o passo a passo da operação**, mas o **efeito final** obtido após sua realização.

Exemplo

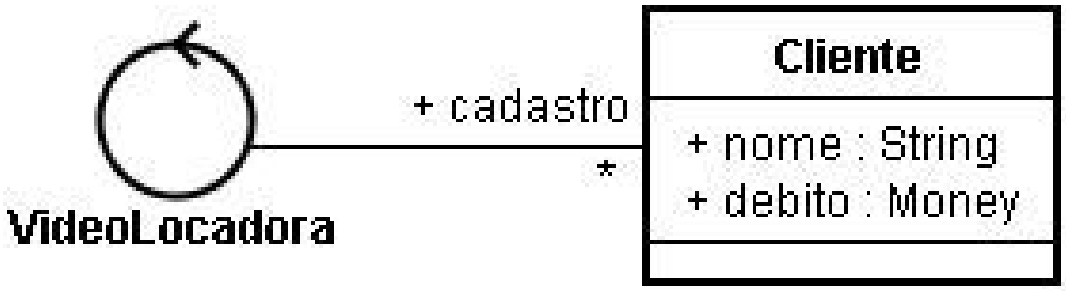
Após a execução da operação registrarCliente(),
foi criada uma nova instância da classe Cliente e
associada ao cadastro da Videolocadora.

Pós-condições semânticas

- **Instância**: criação e destruição
  - Ex:
    - Uma nova instância de Cliente foi criada.
    - A instância de Reserva foi removida do cadastro.
- **Associação**: criação e destruição
  - Ex:
    - *O Filme foi associado ao Cliente*.
    - *A associação entre Pedido e Pagamento foi desfeita*.
- **Atributo**: modificação de valor
  - Ex:
    - O atributo status do Pedido foi atualizado para “concluído”.
    - *O atributo saldo do Cliente foi decrementado*.

Criação de uma instância e sua associação com outra instância preexistente

Pós: foi criado um Cliente e associado à Videolocadora

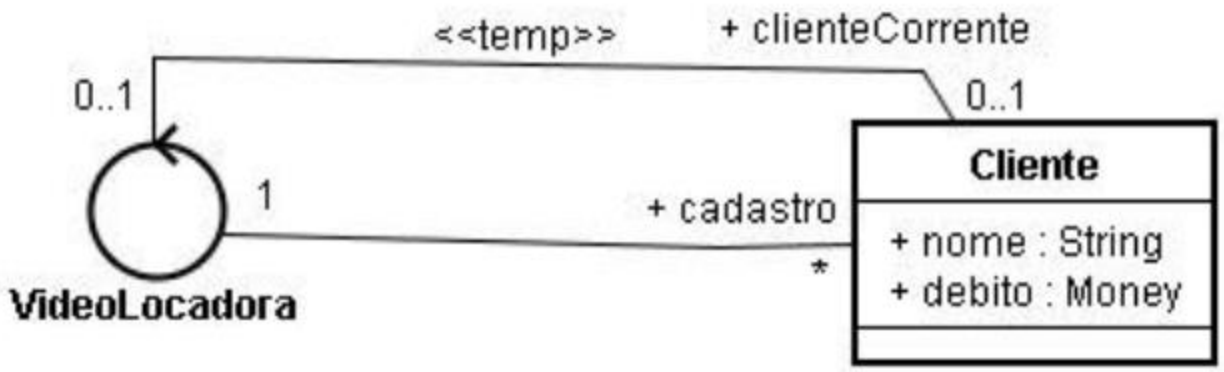


Pós:

```
cliente = Cliente.new()
self.addToCadastro(cliente).
```

Criação de uma associação entre duas instâncias

Pós: O cliente cujo nome é nomeCliente foi associado à VideoLocadora como clienteCorrente

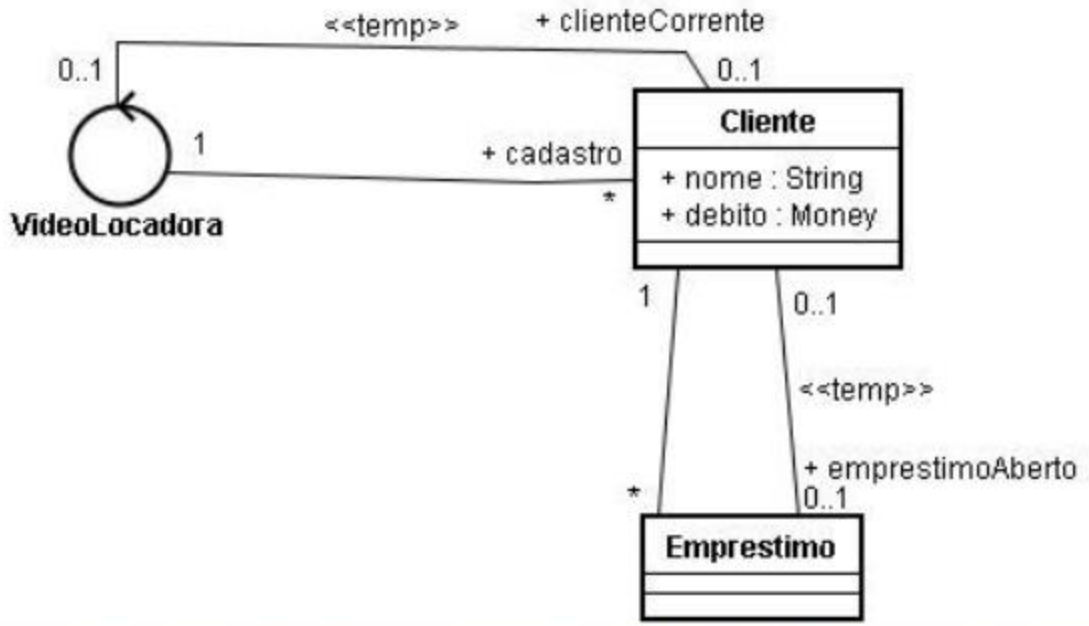


Alias: cliente = self.cadastro→select(nome=nomeCliente)

Pós: self.clienteCorrente = cliente

Pós-condição condicional

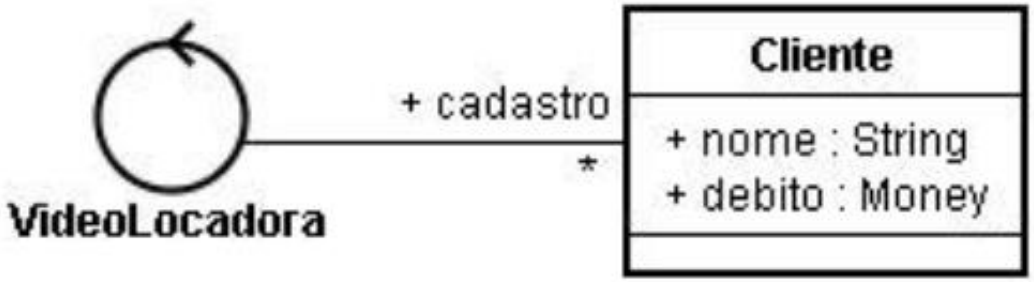
Pós: se não havia nenhum emprestimoAberto associado ao clienteCorrente, então um novo Emprestimo foi criado e associado ao clienteCorrente como emprestimoAberto.



Pós: self.clienteCorrente.emprestimoAberto@pre→size==0 IMPLIES self.clienteCorrente.emprestimoAberto= Emprestimo.new()

Pós-condições - Destruição de uma instância

- Presume-se que quando uma instância é destruída, **todas as associações ligadas a ela também o sejam**.
- Deve-se tomar cuidado com **questões estruturais** (associações obrigatórias) quando um objeto é destruído.



Pós: “foi destruído um Cliente cujo nome é igual a nomeCliente”.

Contrato para Inserção

Classe Videolocadora

operação: cadastraCliente(nomeC,enderecoC,telefoneC:String)

pré:

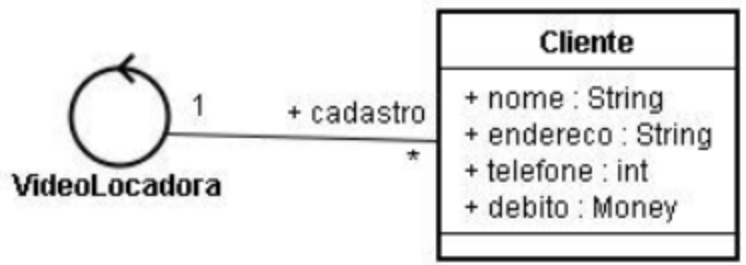
Não existe nenhum *Cliente* com *nome* = *nomeC*.

pós:

Foi criado um *Cliente* e adicionado ao *cadastro*.

Os atributos *nome*, *endereco* e *telefone* do *Cliente* foram alterados para *nomeC*, *enderecoC* e *telefoneC*.

O atributo *debito* do *Cliente* foi alterado para 0,00.



Contrato para Inserção - Em OCL

Classe Videolocadora

operação: cadastraCliente(nomeC,enderecoC,telefoneC:String)

pré:

self.cadastro→select(nome=nomeC)→size==0

pós:

cliente = Cliente.new()

self.addToCadastro(cliente)

cliente.nome = nomeC

cliente.endereco = enderecoC

cliente.telefone = telefoneC

cliente.debito = 0

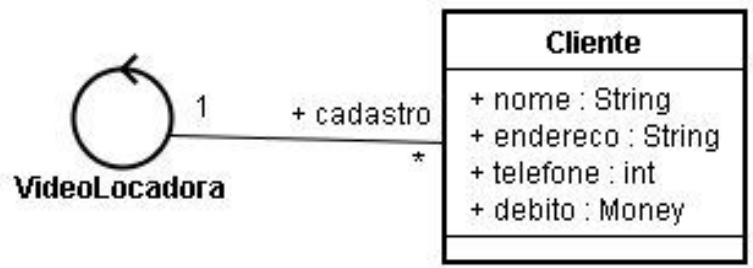
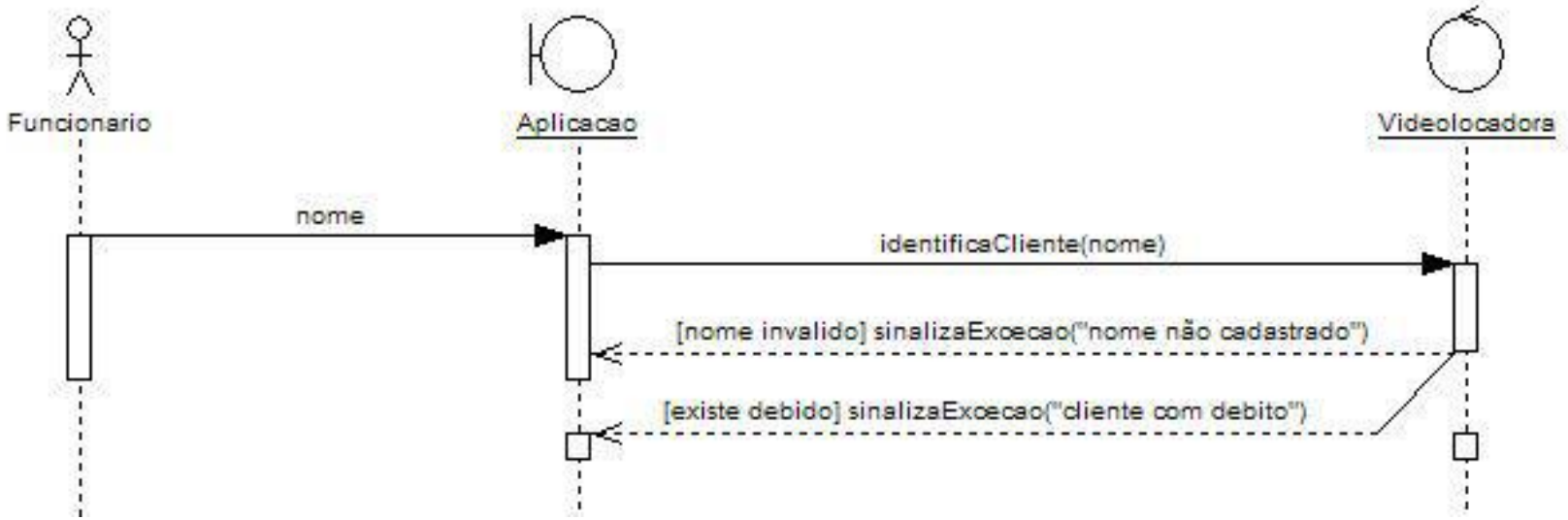
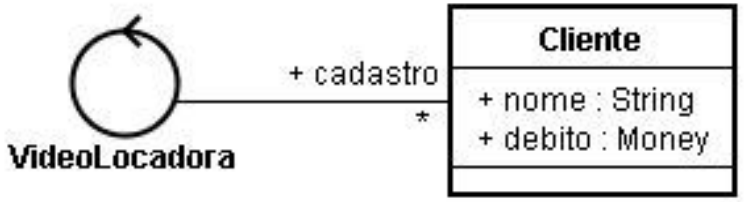


Diagrama de Sequência com Exceções





Operação: identificaCliente(nome:String)

Alias: cliente = self.cadastro→select(nome=nomeCliente)

Pré: -

Exceções:

“Nome invalido” se cliente→size == 0

“Cliente com debito” se cliente.debito != 0

Quiz

Que cuidado deve-se tomar quando um contrato possui uma pós-condição do tipo “foi criada uma instância”?

a) Deve-se inserir a instância em uma lista ou conjunto de instâncias da classe.

b) Deve-se colocar uma pré-condição para verificar se já não existe uma instância alocada para a mesma variável.

c) Deve-se garantir que algum outro contrato terá uma pós-condição do tipo “foi destruída uma instância”.

d) Deve-se garantir que no mesmo contrato exista uma pré-condição do tipo “existe uma instância de...”.

e) Deve-se sempre adicionar uma pós-condição do tipo “foi criada uma associação” entre a instância recém criada e alguma outra instância pré-existente

Quiz

Que cuidado deve-se tomar quando um contrato possui uma pós-condição do tipo “foi criada uma instância”?

a) Deve-se inserir a instância em uma lista ou conjunto de instâncias da classe.

b) Deve-se colocar uma pré-condição para verificar se já não existe uma instância alocada para a mesma variável.

c) Deve-se garantir que algum outro contrato terá uma pós-condição do tipo “foi destruída uma instância”.

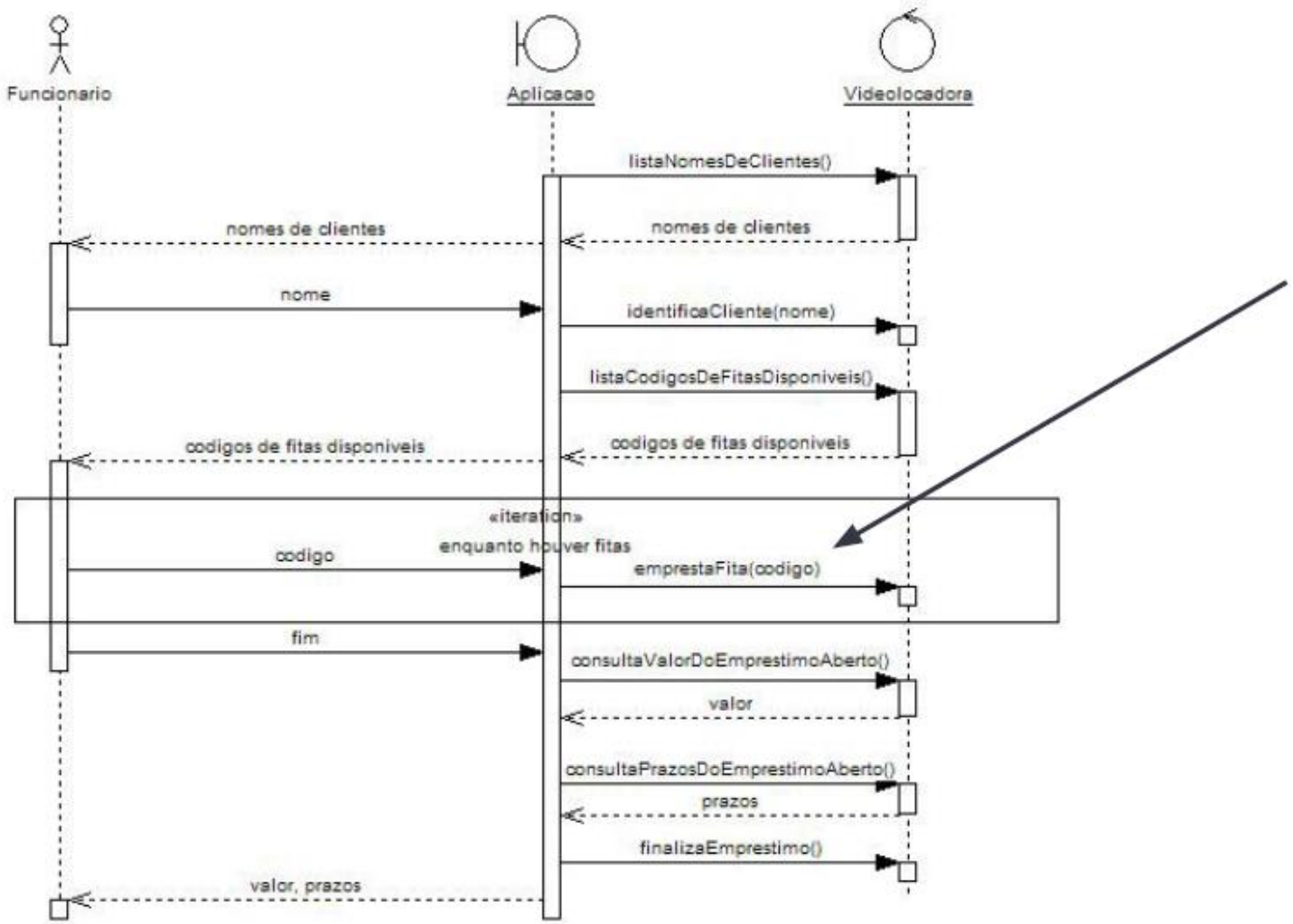
d) Deve-se garantir que no mesmo contrato exista uma pré-condição do tipo “existe uma instância de...”.

e) Deve-se sempre adicionar uma pós-condição do tipo “foi criada uma associação” entre a instância recém criada e alguma outra instância pré-existente

Outras Consultas e Operações (Específicas dos Casos de Uso)

- Em cada **caso de uso**, é comum existir **uma sequência de operações** ou **consultas** que são executadas ao longo de um fluxo.
- Essa **cadeia de execução** costuma estar **explicitada no diagrama de sequência**, mostrando como as mensagens se encadeiam entre os objetos.
- Durante o estudo de cada operação, verifique:
  - **Qual é o objetivo** da operação?
    - *Por que ela existe? O que ela pretende alcançar?*
  - **O que ela espera das anteriores?**
    - *Que informações ou resultados precisam estar disponíveis?*
  - **O que ela produz?**
    - *Quais dados, mudanças ou efeitos resultam da execução?*
  - **Quais exceções podem ocorrer?**
    - *Há falhas possíveis em parâmetros, estados ou associações?*

Exemplo


Funcionario
Aplicacao
Videolocadora
listaNomesDeClientes()
nomes de clientes
nomes de clientes
nome
identificaCliente(nome)
listaCodigosDeFitasDisponiveis()
codigos de fitas disponiveis
codigos de fitas disponiveis
«iteration»
enquanto houver fitas
codigo
emprestaFita(codigo)
fim
consultaValorDoEmprestimoAberto()
valor
consultaPrazosDoEmprestimoAberto()
prazos
finalizaEmprestimo()
valor, prazos

Exemplo : Contrato operação `indentificaCliente`

Classe Videolocadora

operação: identificaCliente(nomeC:String)

alias:

cliente = self.cadastro→select(nome=nomeC)

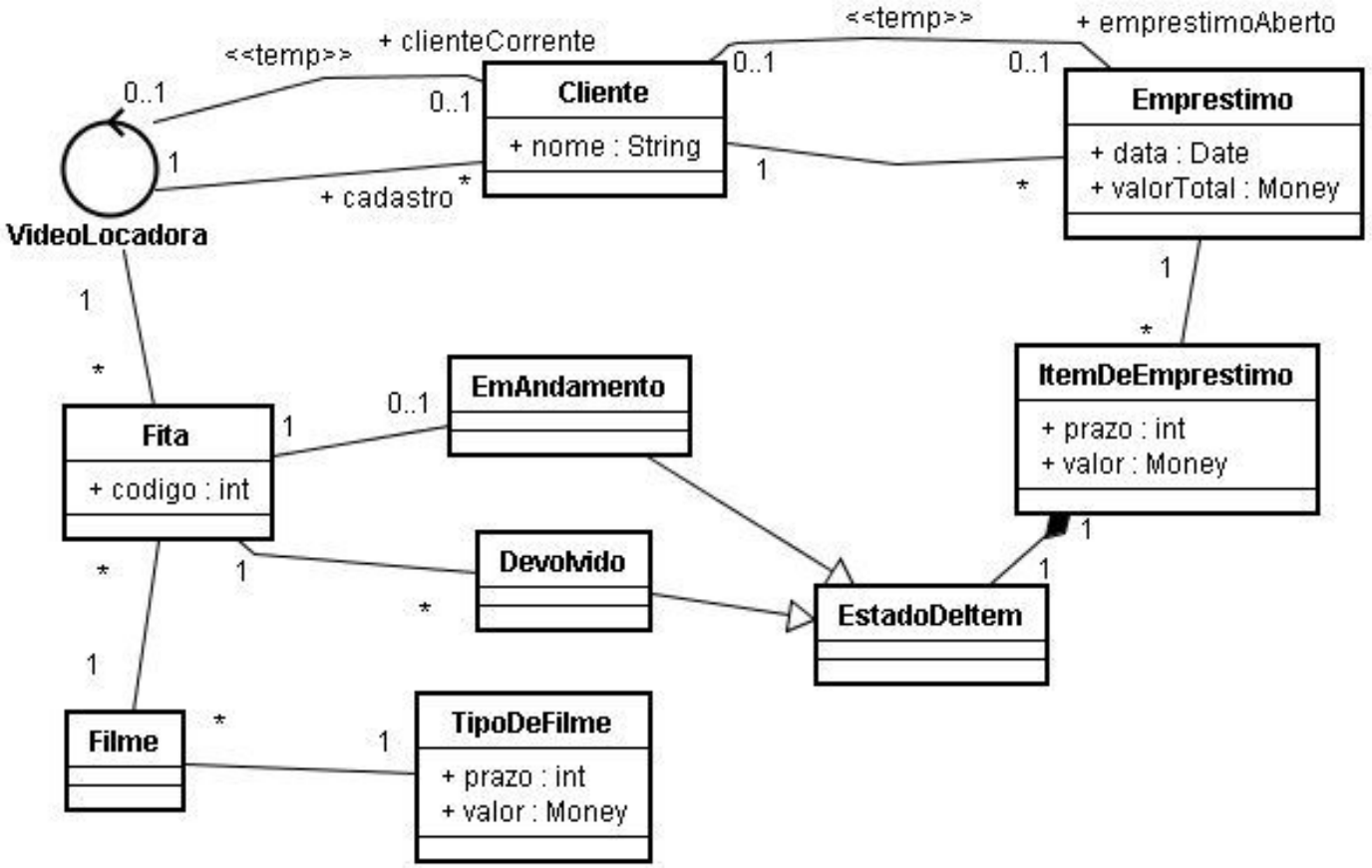
pré:

cliente→size() == 1

self.clienteCorrente→size() == 0

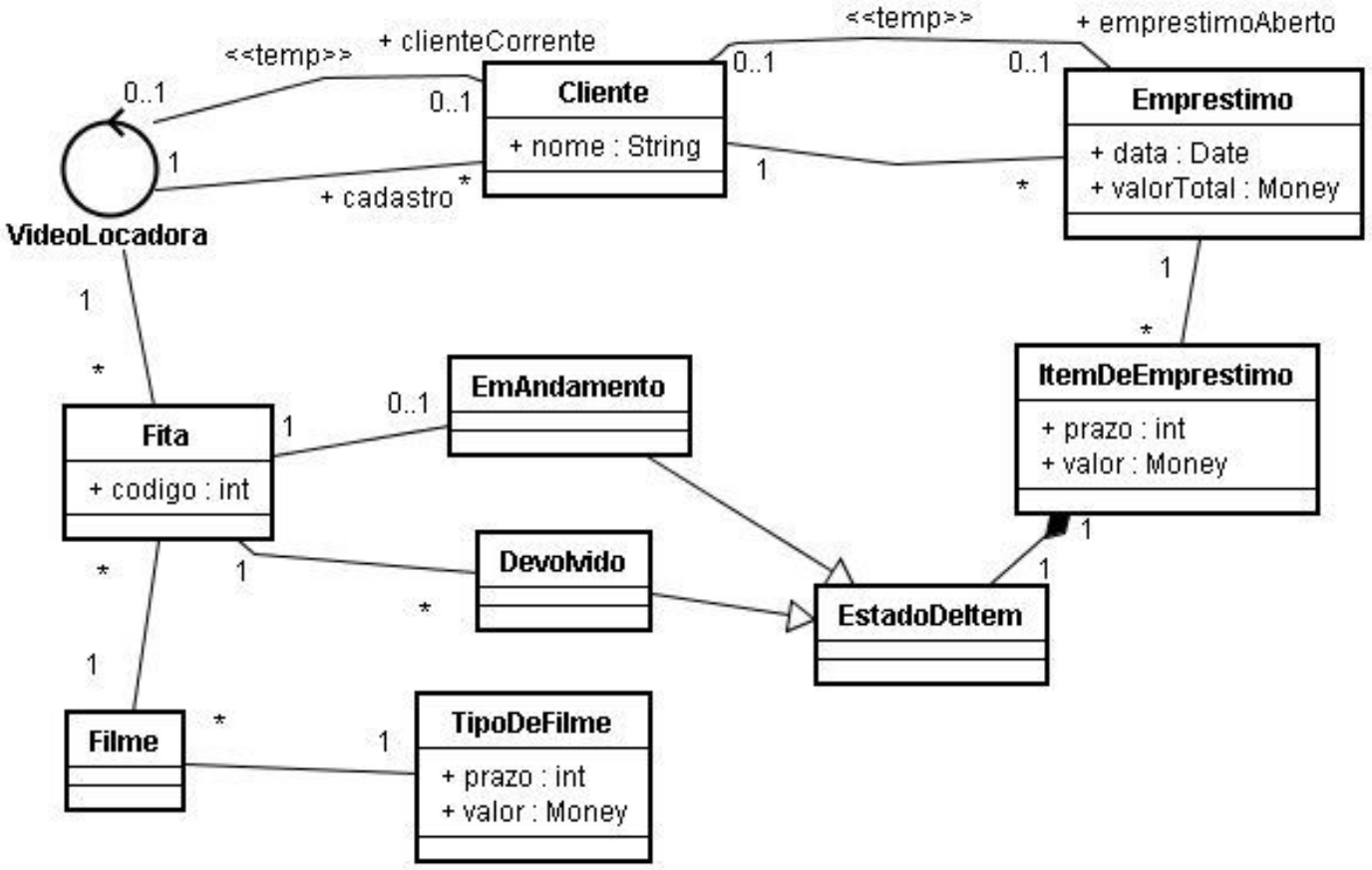
pós:

self.setClienteCorrente(cliente)



Exemplo : Contrato operação `indentificaCliente`

Classe Videolocadora
operação: emprestaFita(codigoF:String)
alias:
fita = self.fitas→select(codigo=codigoF)
pré:
self.clienteCorrente→size() == 1
fita→size() == 1
pós:
self.clienteCorrente.emprestimoAberto→size() == 0 IMPLIES
emprestimo = Emprestimo.new()
emprestimo.data = today()
emprestimo.valorTotal = 0
self.clienteCorrente.setEmprestimoAberto(emprestimo)
item = ItemDeEmprestimo.new()
emprestimo.addItem(item)
estado = EmAndamento.new()
item.addEstado(estado)
estado.setFita(fita)
item.setPrazo(fita.filme.tipoDeFilme.prazo)
item.setValor(fita.filme.tipoDeFilme.valor)
emprestimo.setValorTotal(emprestimo.valorTotal@pre + item.valor)



Quando os Contratos São Úteis

- **Elabore Contratos Quando...**
- Muitos **objetos do domínio** são **criados, atualizados ou associados** em um **único passo** do caso de uso.
- A **descrição do caso de uso** não deixa claro:
  - Quais **atributos** precisam ser atualizados;
  - Quais **associações** devem ser criadas ou removidas.

- **Evite Criar Contratos Quando...**
  - Cada operação do sistema é **simples ou evidente** a partir do caso de uso.
  - Os **casos de uso já fornecem detalhes suficientes** para derivar o projeto, especialmente em operações de **criação ou remoção simples de instâncias**.

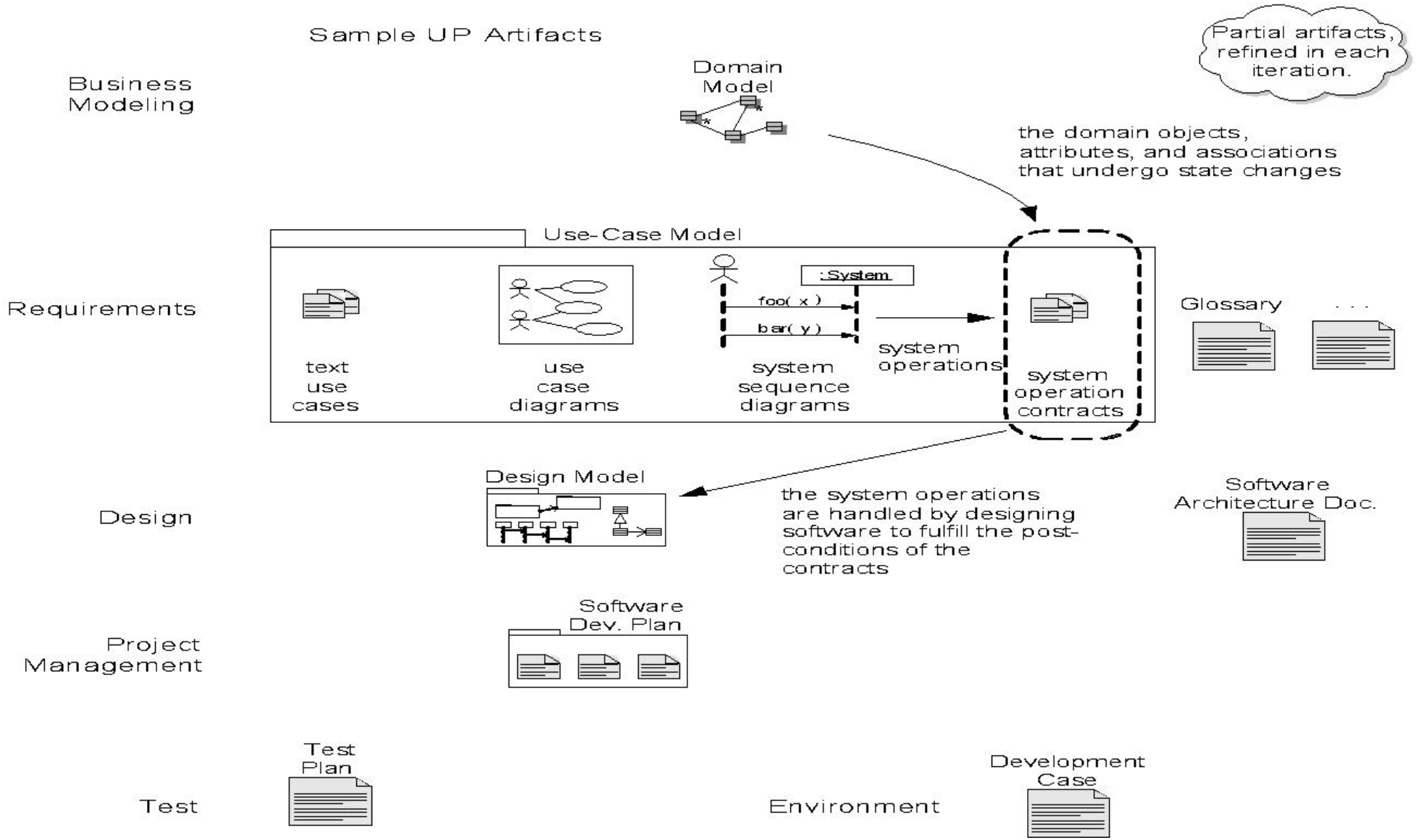
Como Fazer um Contrato

- **1. Identificar as operações de sistema**
  - Analise os diagramas de sequência para localizar as mensagens enviadas ao sistema (as operações principais).
- **2. Construir um contrato para cada operação relevante**
  - Nem toda operação precisa de um contrato, priorize aquelas com impactos múltiplos ou complexos.
- **3. Escrever a seção *Responsabilidades***
  - Descreva de forma informal o propósito da operação:
    - O que essa operação deve realizar e por quê?
  - *Exemplo:* "Registrar um novo cliente no sistema e associá-lo à videolocadora."
- **4. Completar a seção *Pós-condições***
  - Descreva, de maneira declarativa, as mudanças de estado que ocorrem nos objetos do modelo conceitual:
    - Criação e remoção de instâncias
    - Modificação de atributos
    - Formação e quebra de associações *(erro mais comum!)*
  - *Dica:* Use frases no passado, indicando o estado resultante após a execução

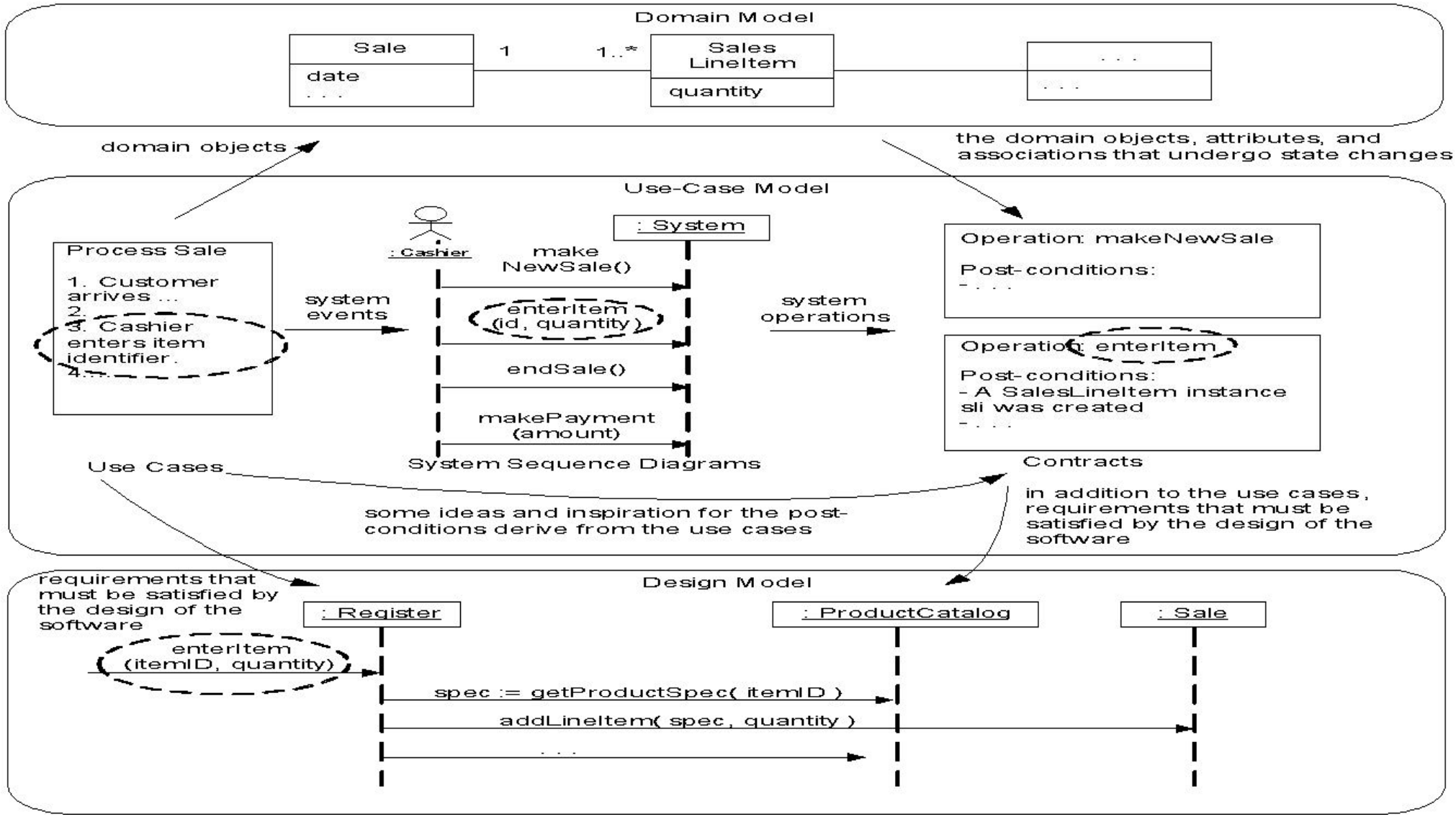
Como Fazer um Contrato

- **5. Completar a seção Pré-condições**,
  - Descreva as pré-suposições sobre o estado do sistema no início da operação:
  - Coisas que devem ser testadas pelo sistema em algum ponto durante a execução da operação
  - Coisas que não são testadas, mas sobre as quais depende fortemente o sucesso da operação

Contratos e Outros Artefatos



Relacionamento entre os artefatos da UML



Contrato - documentação importante