Projeto detalhado
Diagrama de classes (de projeto)

Bruna Diirr

brunadiirr@ic.uff.br

Divisão da fase de Projeto

Projeto geral ou preliminar

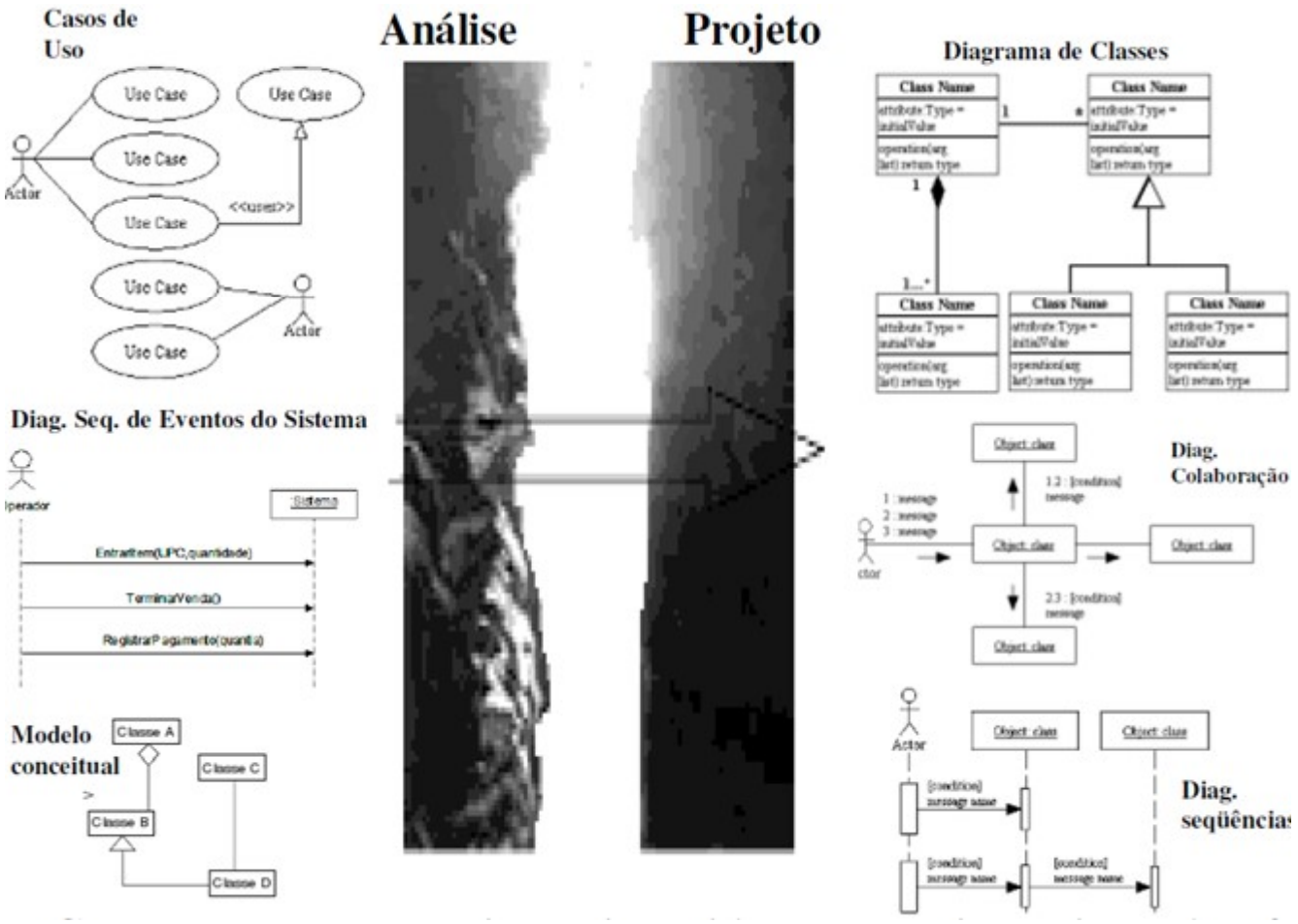
Tradução da especificação do sistema em termos da arquitetura a partir do conhecimento adquirido com requisitos (funcionais e não funcionais

Projeto detalhado

Refinamento progressivo e adição de detalhes à arquitetura visando a codificação → *Modelos de projeto*

Modelos de análise → Modelos de projeto

Capturam resultados do processo de investigação do domínio do problema



Capturam resultados do processo de investigação do domínio da solução

Modelos de análise → Modelos de projeto

Artefatos da fase de análise servem como insumo para desenvolvimento de uma solução lógica

> *"Identificados os requisitos e o modelo conceitual, acrescente os métodos às classes de software e defina as mensagens/troca de mensagens para atender aos requisitos"*

O que isso quer dizer? Com que artefatos vamos trabalhar?

- Diagramas de classes e Diagramas de interação!
- Nossa estratégia
  - Primeiro: Diagramas de interação (Sequência e Colaboração) → Última aula!
  - Em seguida: Diagramas de classes (de projeto) → A partir de agora ☺

Diagrama de classes de projeto

Ilustra as especificações de software para classes e interfaces do sistema

Obtido através da adição de detalhes ao diagrama de classes de análise conforme a solução de software escolhida

Inclui as seguintes informações típicas:

- classes, associações e atributos
- interfaces, com operações e constantes
- métodos
- tipos de atributos
- navegabilidade
- dependências

Diagrama de classes de análise x Diagrama de classes de projeto

Diagrama de classes de análise

abstrações de conceitos, ou objetos, do mundo real

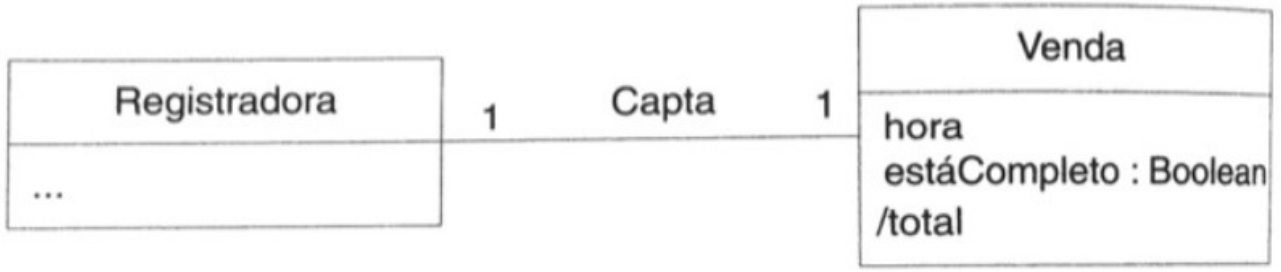
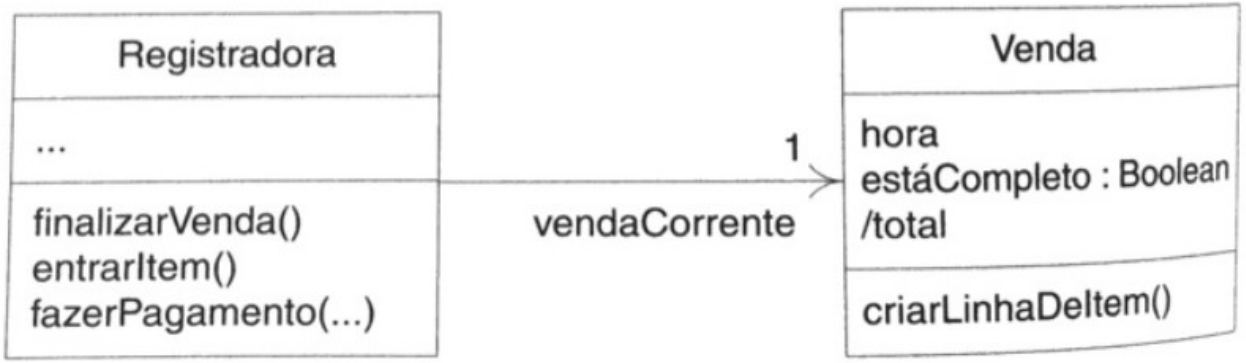
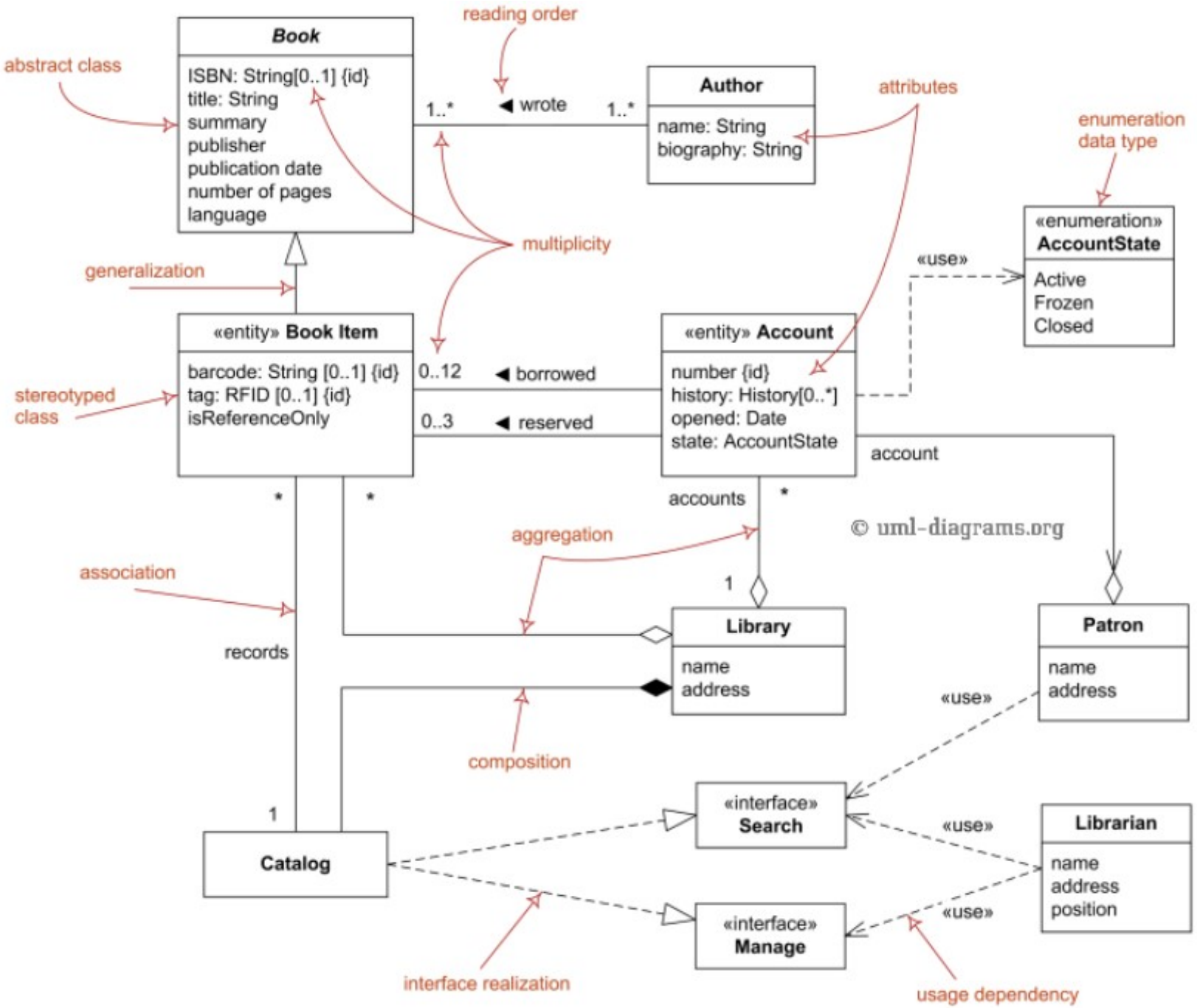
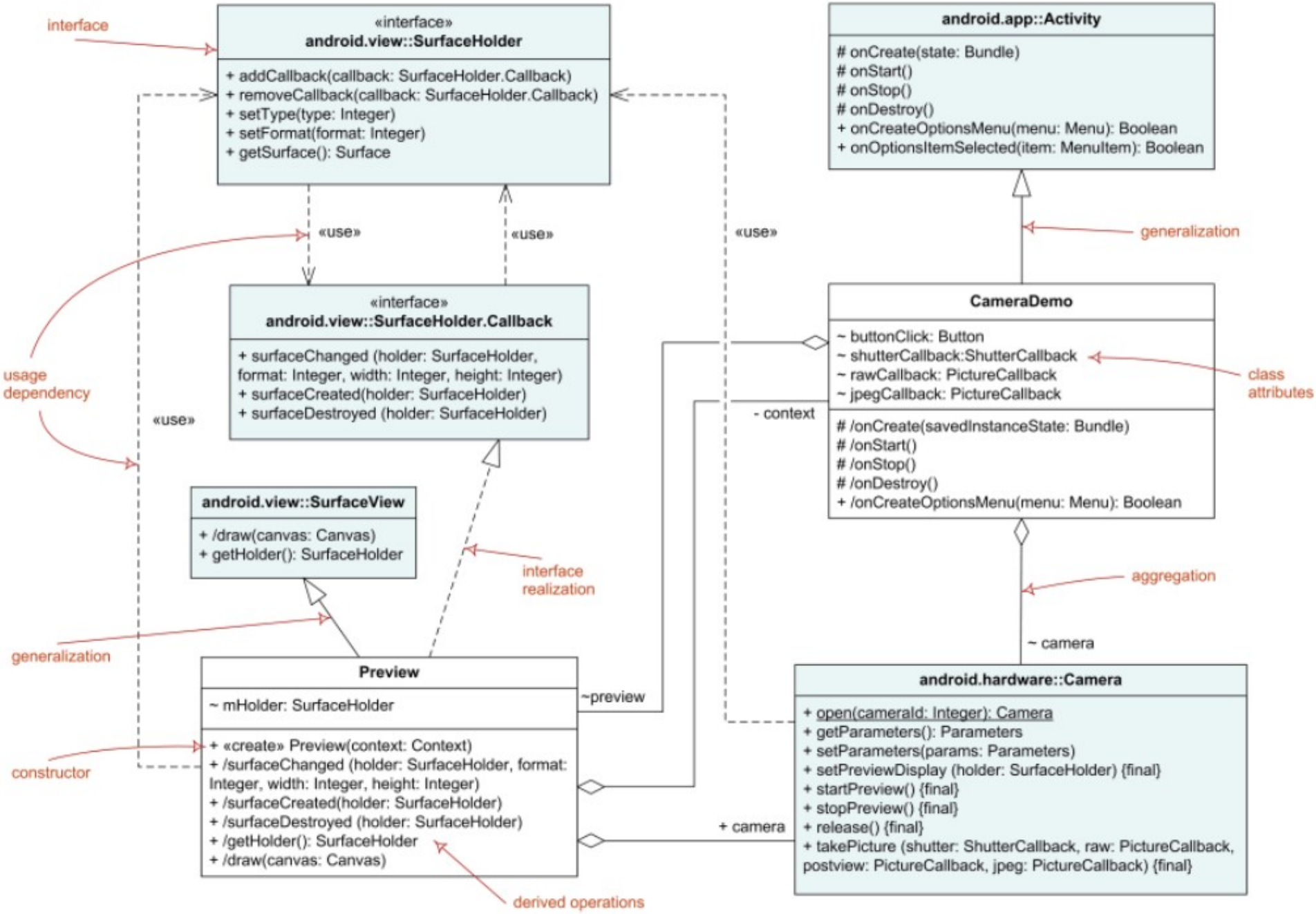


Diagrama de classes de projeto

Definição de classes como componentes de software




Book
ISBN: String[0..1] {id}
title: String
summary
publisher
publication date
number of pages
language
abstract class
reading order
1..*
wrote
1..*
Author
name: String
biography: String
attributes
enumeration
data type
«enumeration»
AccountState
Active
Frozen
Closed
«use»
multiplicity
generalization
«entity» Book Item
barcode: String [0..1] {id}
tag: RFID [0..1] {id}
isReferenceOnly
stereotyped
class
0..12
borrowed
0..3
reserved
«entity» Account
number {id}
history: History[0..*]
opened: Date
state: AccountState
account
accounts
*
*
*
1
© uml-diagrams.org
aggregation
association
records
Library
name
address
Patron
name
address
«use»
composition
1
«interface»
Search
«use»
Librarian
name
address
position
Catalog
«interface»
Manage
«use»
interface realization
usage dependency


interface
«interface»
android.view::SurfaceHolder
+ addCallback(callback: SurfaceHolder.Callback)
+ removeCallback(callback: SurfaceHolder.Callback)
+ setType(type: Integer)
+ setFormat(format: Integer)
+ getSurface(): Surface
android.app::Activity
onCreate(state: Bundle)
onStart()
onStop()
onDestroy()
+ onCreateOptionsMenu(menu: Menu): Boolean
+ onOptionsItemSelected(item: MenuItem): Boolean
«use»
«use»
«use»
generalization
«interface»
android.view::SurfaceHolder.Callback
+ surfaceChanged (holder: SurfaceHolder, format: Integer, width: Integer, height: Integer)
+ surfaceCreated(holder: SurfaceHolder)
+ surfaceDestroyed (holder: SurfaceHolder)
CameraDemo
~ buttonClick: Button
~ shutterCallback:ShutterCallback
~ rawCallback: PictureCallback
~ jpegCallback: PictureCallback
# /onCreate(savedInstanceState: Bundle)
# /onStart()
# /onStop()
# /onDestroy()
+ /onCreateOptionsMenu(menu: Menu): Boolean
class attributes
usage dependency
«use»
- context
android.view::SurfaceView
+ /draw(canvas: Canvas)
+ getHolder(): SurfaceHolder
interface realization
aggregation
~ camera
generalization
Preview
~ mHolder: SurfaceHolder
+ «create» Preview(context: Context)
+ /surfaceChanged (holder: SurfaceHolder, format: Integer, width: Integer, height: Integer)
+ /surfaceCreated(holder: SurfaceHolder)
+ /surfaceDestroyed (holder: SurfaceHolder)
+ /getHolder(): SurfaceHolder
+ /draw(canvas: Canvas)
~preview
android.hardware::Camera
+ open(cameraId: Integer): Camera
+ getParameters(): Parameters
+ setParameters(params: Parameters)
+ setPreviewDisplay (holder: SurfaceHolder) {final}
+ startPreview() {final}
+ stopPreview() {final}
+ release() {final}
+ takePicture (shutter: ShutterCallback, raw: PictureCallback, postview: PictureCallback, jpeg: PictureCallback) {final}
+ camera
constructor
derived operations

Como construir?

Na prática, o diagrama de classes pode ser construído à medida que a fase de projeto avança, a partir dos diagramas de interação e de classes de análise

Perspectivas consideradas:

- Classes
  - Identificadas através dos diagramas de interação
- Atributos
  - Copiados do diagrama de classes de análise
  - Adicionar tipos e valor inicial
- Métodos
  - Identificados através dos diagramas de interação
  - Adicionar parâmetros e retornos de métodos (incluindo tipos)
- Associações
  - Investigar dependências, navegabilidade e conectividade

Como construir?

- Classes (considerando a perspectiva de implementação)
  - Corresponde a um tipo de uma linguagem de programação
  - Um modelo genérico para criar variáveis que armazenarão objetos correspondentes
  - Objetos (nessa mesma perspectiva)
    - Representa um módulo de software que recebe e produz dados, possuindo
      - Identidade: Identificador em linguagem de implementação
      - Atributos: Variáveis e seus tipos, que recebem diferentes valores e definem o estado do objeto
      - Comportamento: Funções ou procedimentos. Seus resultados determinam o comportamento do objeto
  - No Projeto: Identificadas a partir dos diagramas de interação

Como construir?

Atributos

Permitem que a classe armazene informações necessárias à realização de suas tarefas

No Projeto:

Copiados do diagrama de classes de análise

Refinamento para adicionar tipos e valor inicial, segundo notação UML

Como construir?

Atributos

Notação

+ (pública): Atributo visível no exterior da classe
- (privada): Atributo visível somente por membros da classe
(protegida): Atributo visível também por suas subclasses

Valor ou conteúdo do atributo imediatamente após sua criação – valor default (ex.: resultado: int=0)

[Visibilidade] Nome [Multiplicidade] : [Tipo] = [Valor] {[Propriedades]}

Identificador do atributo

Usada para especificar atributos que são arranjos
Indica dimensão de vetores e matrizes (ex.: matrizDeValores[5,10])

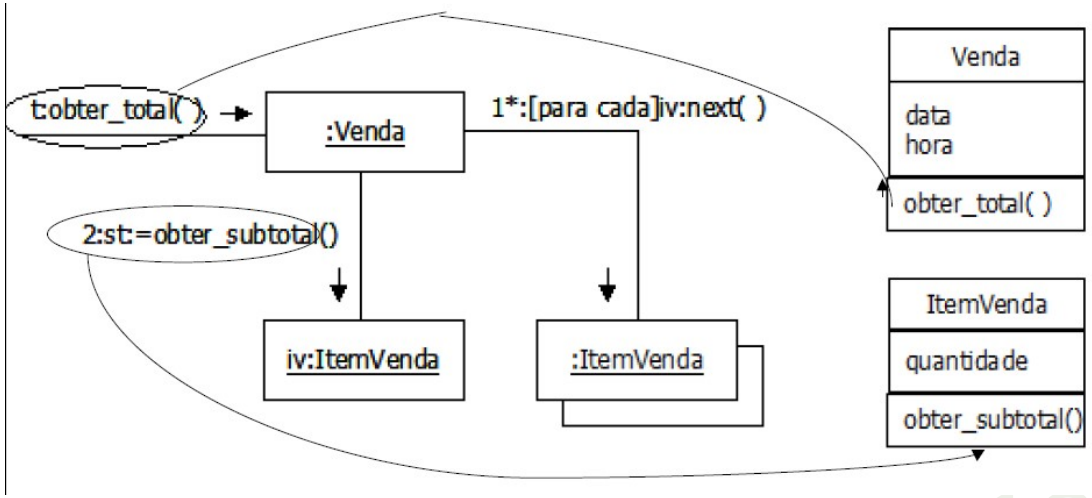
Indicam formato dos valores que atributo pode assumir (ex.: salario: float)

Descrevem comentários ou indicações sobre atributo, podendo mostrar se ele é ou não opcional (ex.: dataDaVenda {valor constante})

Como construir?

- Métodos
  - Operações x Métodos
    - Operações: Algo que se evoca sobre um objeto (a chamada do procedimento)
    - Método: Forma como classe realiza uma operação a (o corpo do procedimento)

Ex.: Se uma classe recebe uma mensagem A, ela deverá executar em resposta uma operação A, que é implementada por um método A.



Como construir?

- Métodos
  - Manipulam valores dos atributos, com objetivo de atender às mensagens que o objeto recebe

  - No Projeto:
    - Identificados através dos diagramas de interação
    - Gerar métodos para construção, destruição, acesso (*get*, *set*) etc.
    - Refinamento para adicionar parâmetros e retornos de métodos (incluindo tipos), segundo notação UML
    - OBS:
      - Notas explicativas também são úteis no esclarecimento de como método deve ser implementado
      - Diagrama de atividades também pode ser usado para detalhar o funcionamento de métodos mais complexos

Como construir?

Métodos

Notação

+ (pública): Método visível no exterior da classe
- (privada): Método visível somente por membros da classe
(protegida): Método visível também por suas subclasses

Indica se método retorna algum valor ao término da execução e qual tipo de dado do valor retornado (ex.:ArmazenarDados(): bool)

[Visibilidade] Nome (Parametros) : [Retorno] {[Propriedades]}

Identificador do método

Dados de entrada e/ou saída para o método.
Representados por nomeParametro: Tipo = ValorPadrão
(ex.: ArmazenarDados(nome:char[30],salario:float=0.0))

Descrevem comentários ou indicações sobre método (ex.: Area() {Área <= 600})

Como construir?

Associações

No Projeto:

Investigar dependências, navegabilidade e conectividade

Objetivos:

Aumentar encapsulamento de cada classe

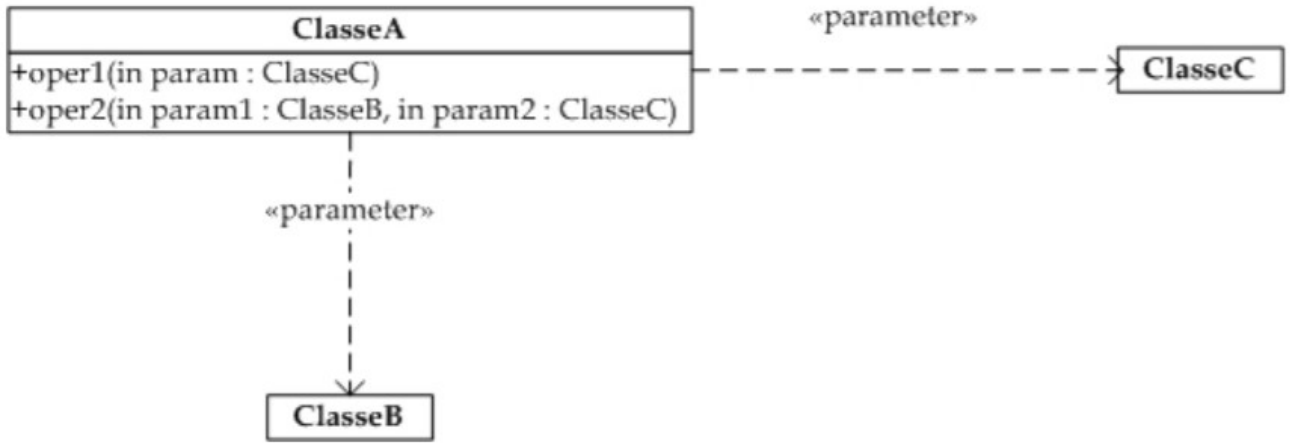
Diminuir acoplamento entre as classes

Como construir?

- Associações
  - Dependência
    - Indica que uma classe depende das operações fornecidas por outra classe
    - Na análise: Apenas dependência por atributo (ou estrutural)
      - Classe dependente possui atributo que é referência para outra classe
    - Porém, também existem dependências não estruturais:
      - Por variável global: Um objeto de escopo global é referenciado em algum método da classe dependente
      - Por variável local: Um objeto recebe outro como retorno de método, ou possui uma referência para outro objeto como variável local em algum método
      - Por parâmetro: Um objeto recebe outro como parâmetro em um método

Como construir?

- Associações
  - Dependência
    - Notação
      - Seta tracejada ligando classes envolvidas
      - Direção
        - Classe dependente (cliente) → Classe da qual ela depende (fornecedora)
      - Estereótipos: <<global>>, << local>>, << parameter>>



Como construir?

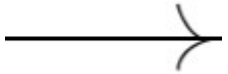
- Associações
  - Navegabilidade
    - Associações podem ser bidirecionais ou unidirecionais
      - Bidirecional: Indica que há um conhecimento mútuo entre objetos associados
      - Unidirecional: Indica que apenas um dos extremos da associação tem ciência da existência da mesma
    - Escolha através do estudo dos diagramas de interação

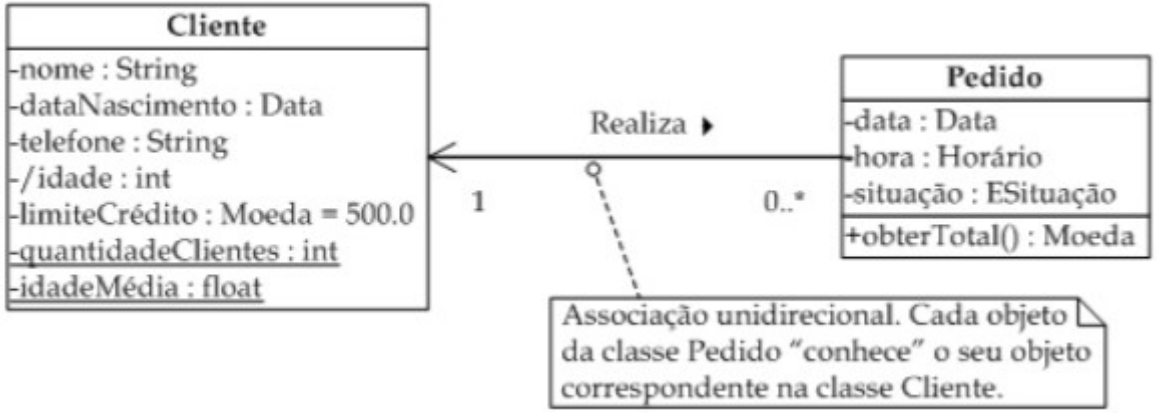
Como construir?

Associações

Navegabilidade

Notação

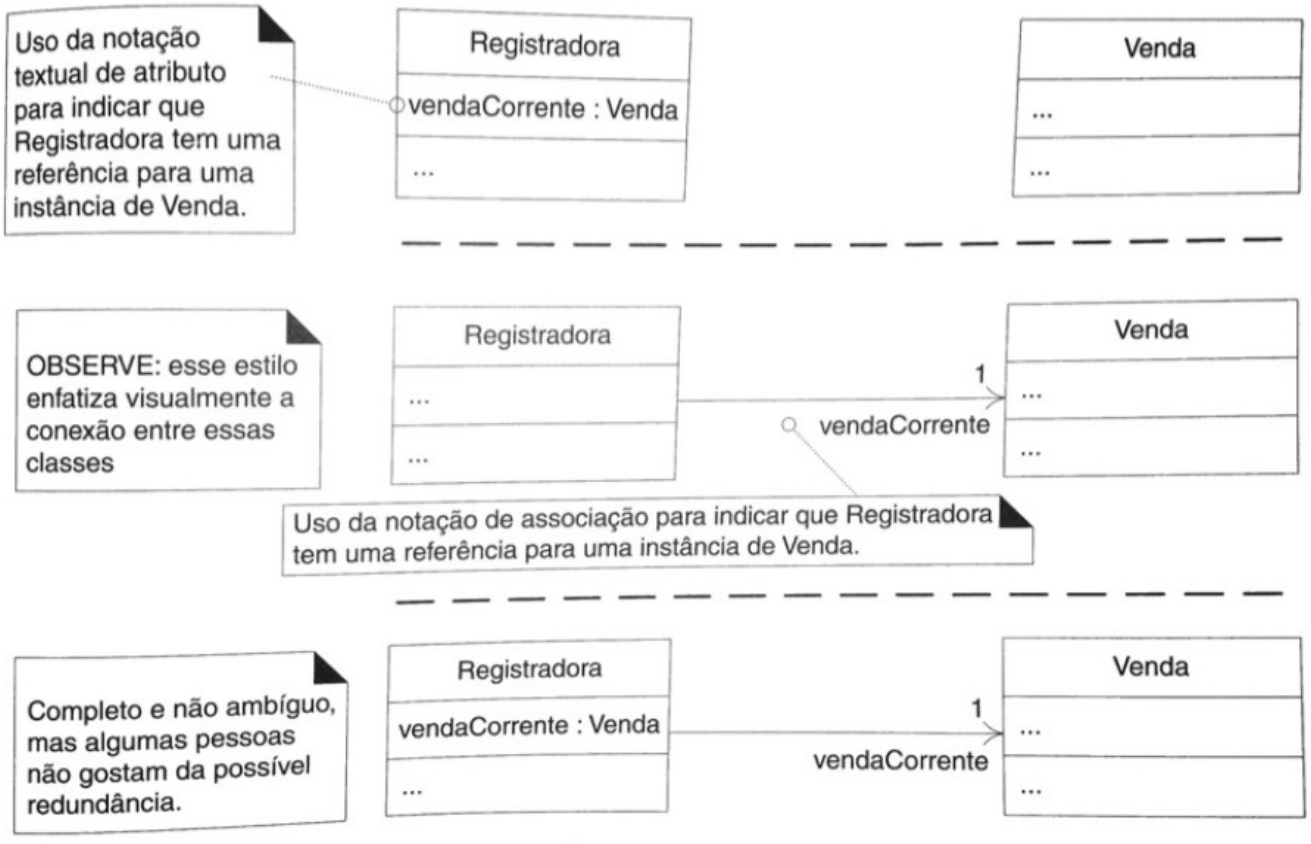
Adição de um sentido (seta contínua) à associação 



Como construir?

Associações

Navegabilidade



Como construir?

- Associações
  - Conectividade
    - Três casos, em função da conectividade: 1:1, 1:N e N:M

    - Associação 1:1
      - Navegabilidade unidirecional (sentido A→B): Definir atributo tipo B na classe A
      - Navegabilidade bidirecional: Aplicar procedimento acima para ambas classes

    - Associação 1:N ou N:M
      - Utilizados atributos cujos tipos representam coleções de elementos
      - Também é comum o uso de classes parametrizadas
        - Definir classe parametrizada cujo parâmetro é classe correspondente ao lado muitos da associação
        - Caso N:M semelhante ao refinamento das associações 1:N

Como construir?

Associações

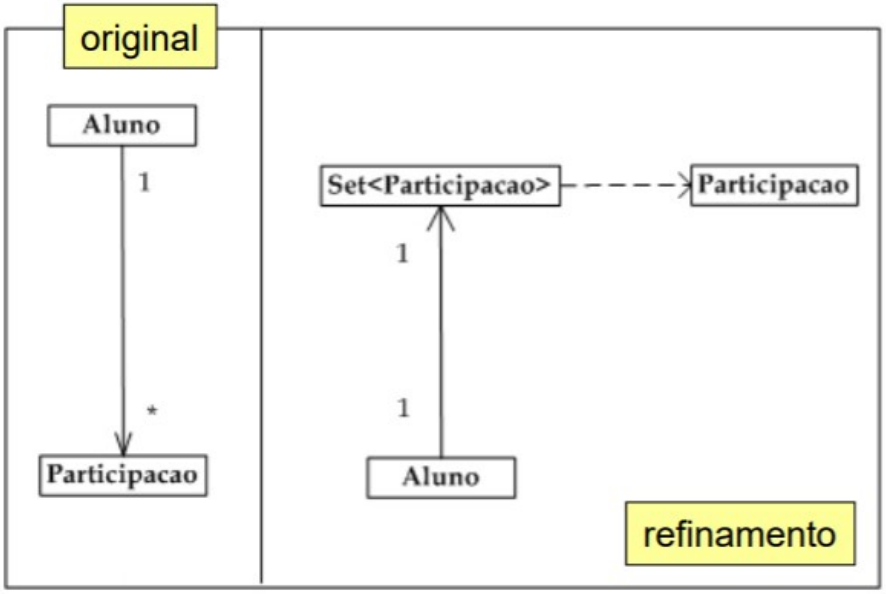
Conectividade 1:1


```
public class Professor {
  private GradeDisciplinas grade;
  ...
}
```

Como construir?

Associações

Conectividade 1:N

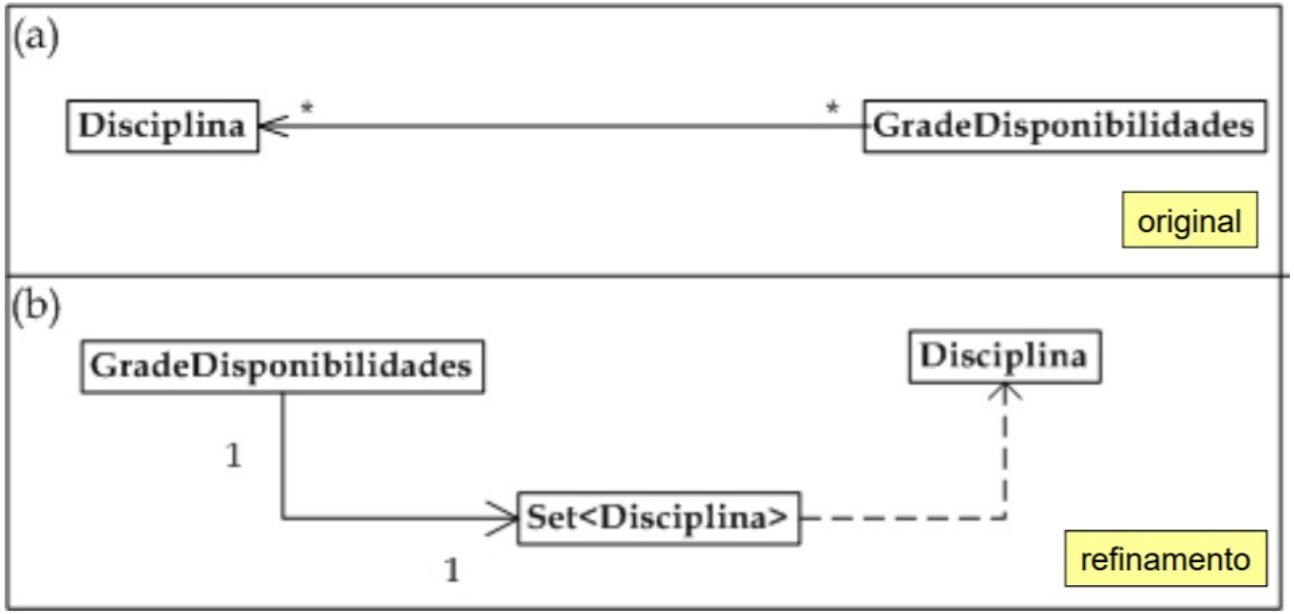



```
public class Aluno {
  private Set<Participacao> participacoes;
  ...

  public boolean adicionarParticipacao(Participacao p){
       ...
  }

  public boolean removerParticipacao(Participacao p) {
       ...
  }
}
```

Como construir?

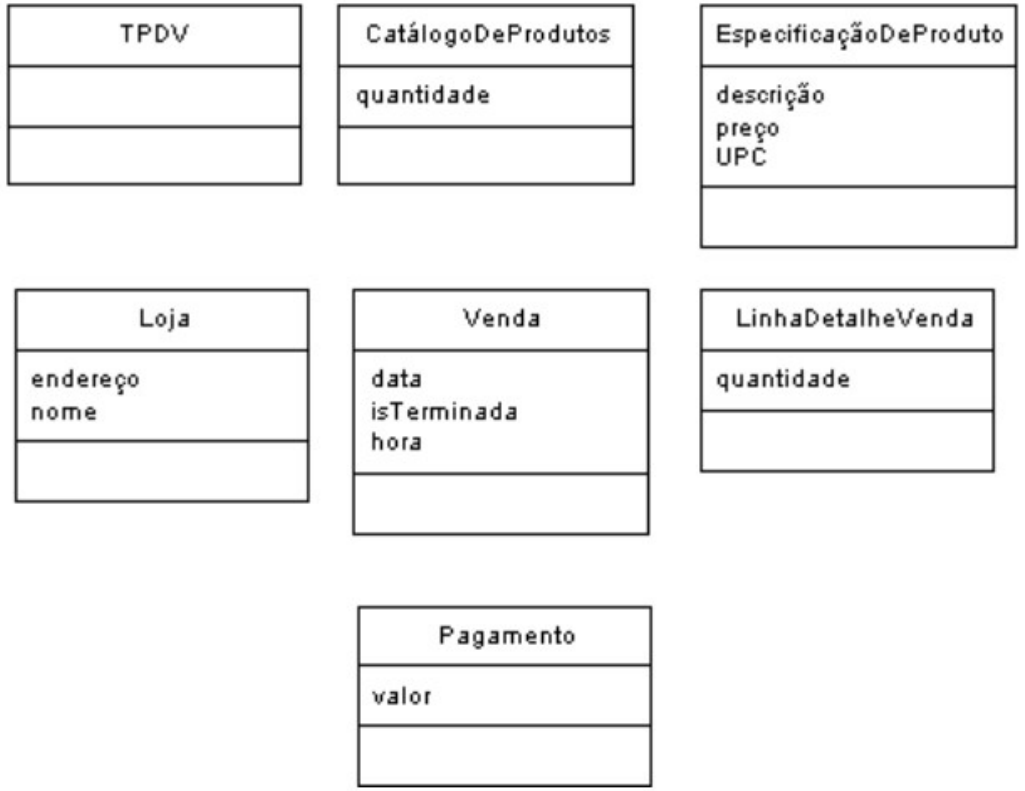
Associações

Conectividade N:M



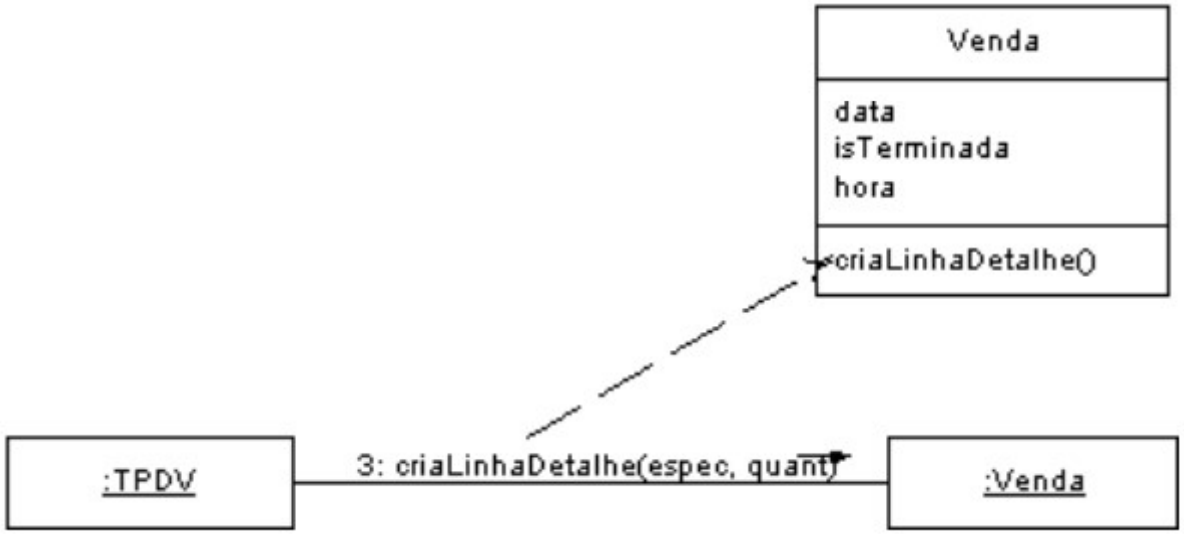
Exemplo: Terminal de Ponto de Venda (TPV)

1) Identificação de classes (diagramas de interação) e atributos (diagrama de classes de análise)



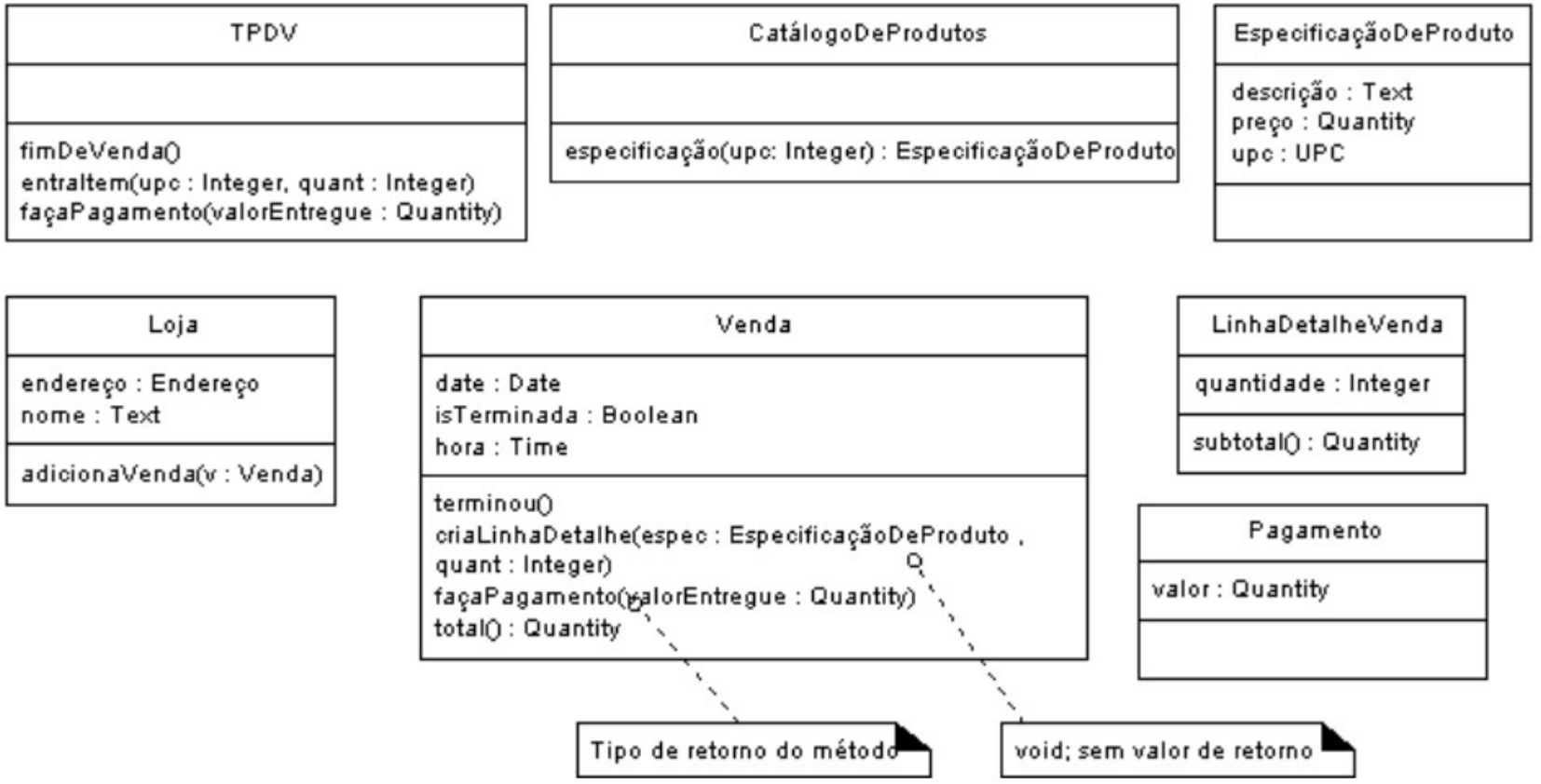
Exemplo: Terminal de Ponto de Venda (TPV)

2) Identificação de métodos (diagramas de interação)



Exemplo: Terminal de Ponto de Venda (TPV)

3) Refinamento de atributos e métodos (notação UML)



Exemplo: Terminal de Ponto de Venda (TPV)

4) Identificação de navegabilidade (diagramas de classes de análise e diagramas de interação)

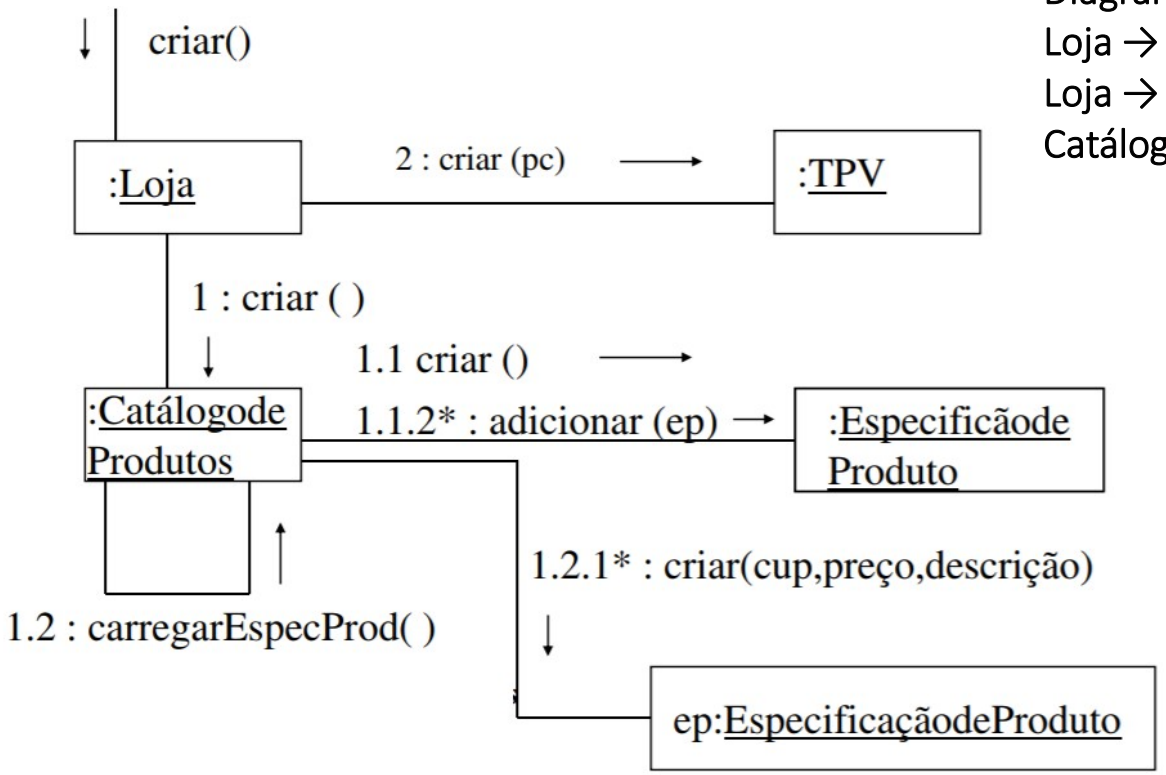
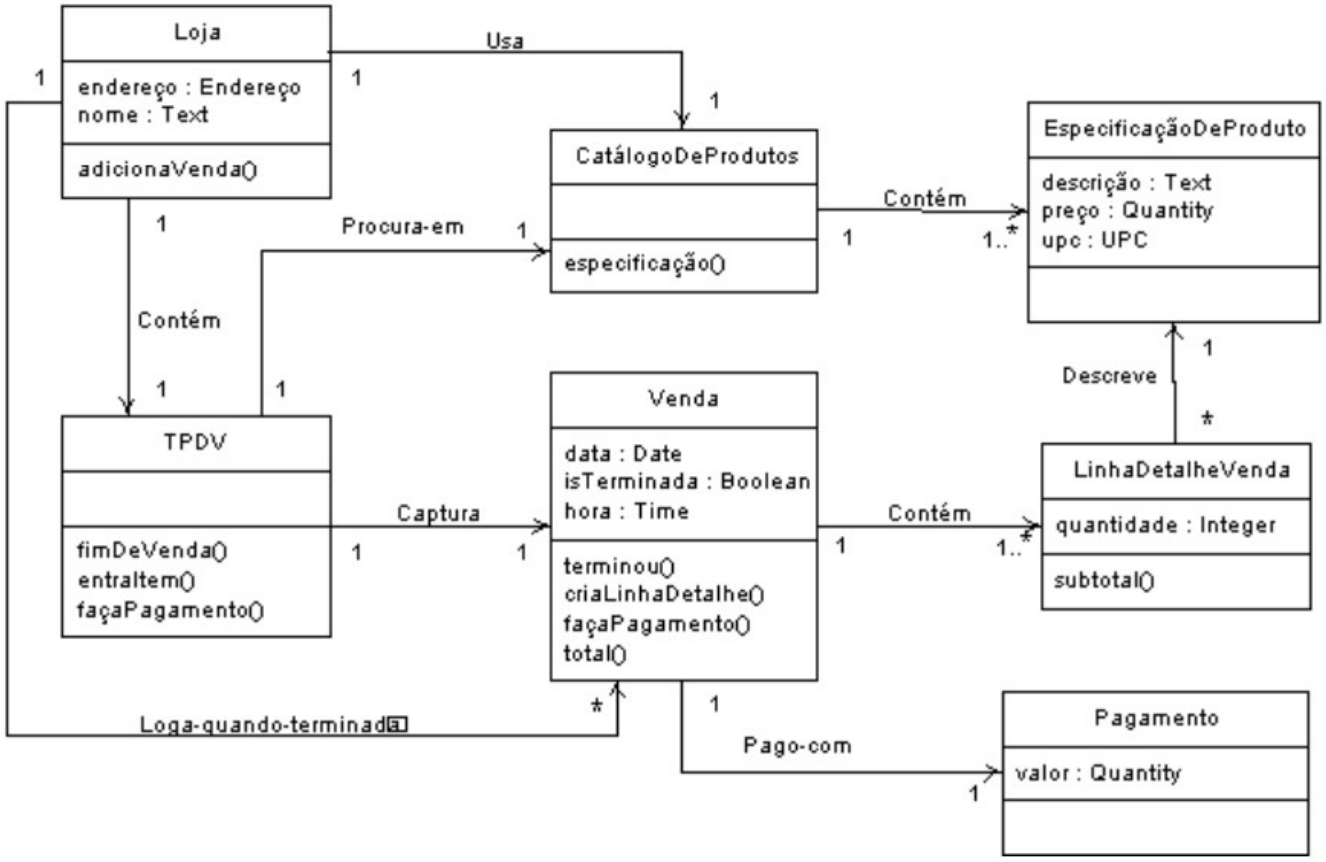


Diagrama implica quais navegabilidades?
Loja → TPV
Loja → CatálogodeProdutos
CatálogodeProdutos → EspecificaçãodeProduto

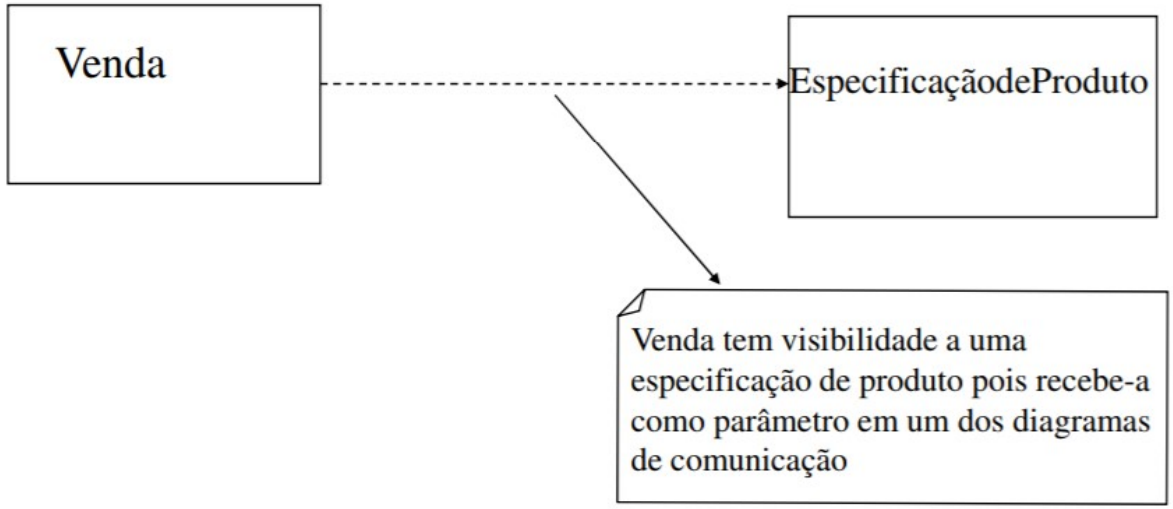
Exemplo: Terminal de Ponto de Venda (TPV)

4) Identificação de navegabilidade (diagramas de classes de análise e diagramas de interação)



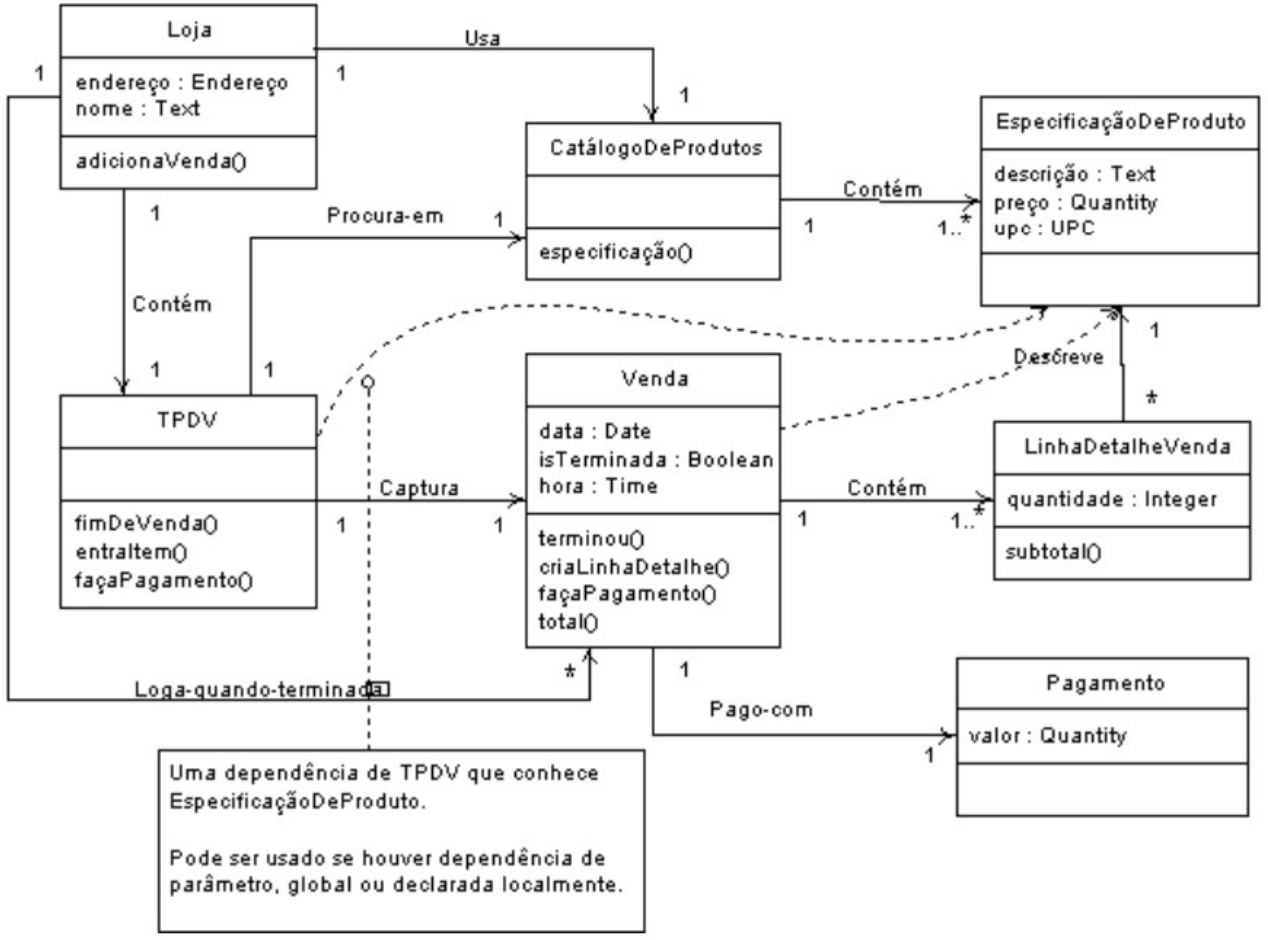
Exemplo: Terminal de Ponto de Venda (TPV)

5) Identificação de dependências (diagramas de interação)



Exemplo: Terminal de Ponto de Venda (TPV)

5) Identificação de dependências (diagramas de interação)



Projeto detalhado
Diagrama de classes (de projeto)

Bruna Diirr

brunadiirr@ic.uff.br