Métodos Avançados de Programação

REVISÃO DE DIAGRAMAS DE CLASSE

DRª. ALANA MORAIS

Aula passada

- Introdução a disciplina
- Apresentação e discussão do plano de aula
- Revisão de Java

Diagramas importantes para entender padrões

- Diagrama de classe (hoje)
- Diagrama de sequência

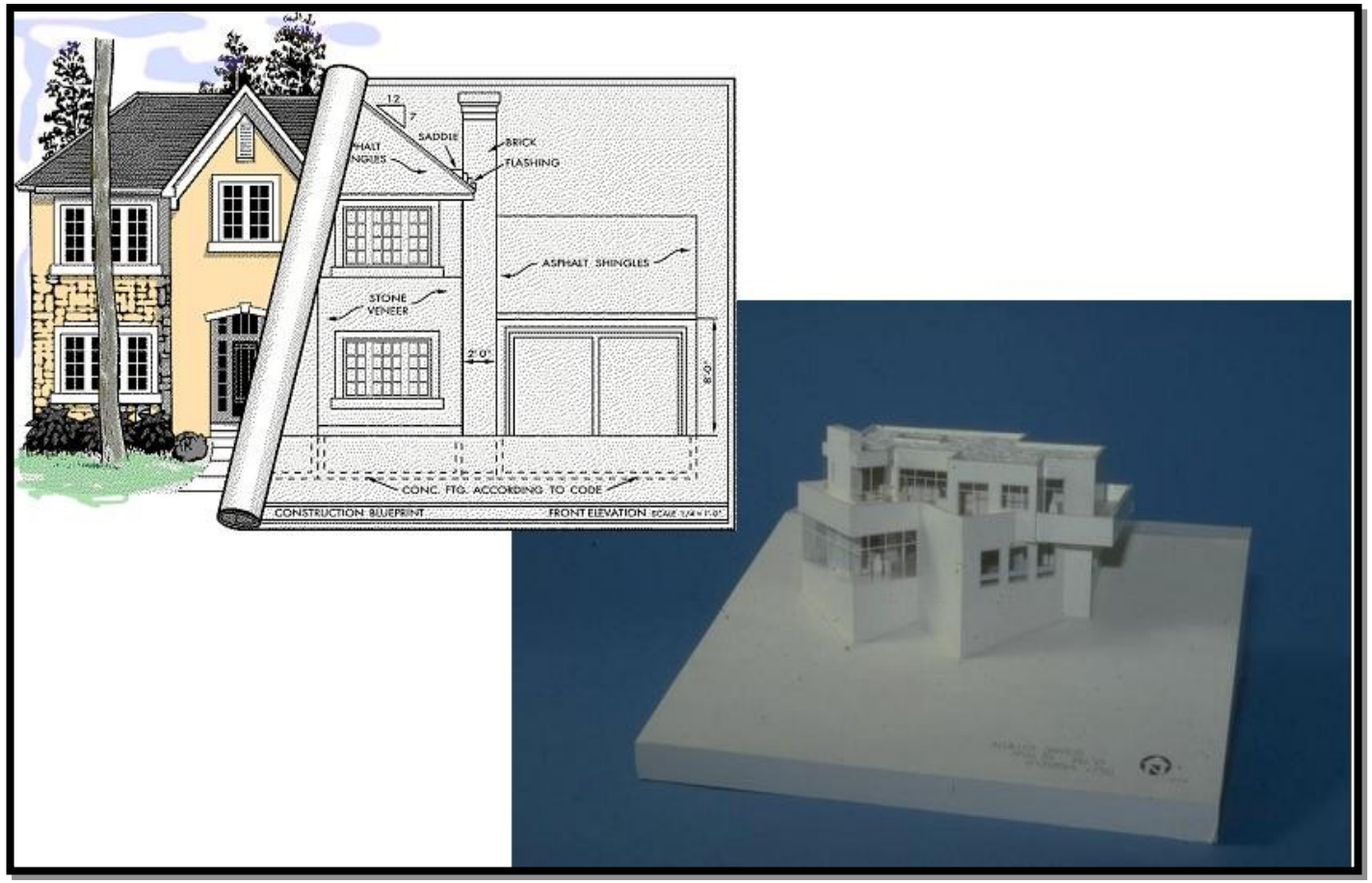
Projeto Orientado a Objetos

- Maneira natural de visualizar o *software*
  - Documentação e comunicação entre a equipe
- Modela o *software* semelhante ao mundo real - usando objetos
- Objetos são modelados em termos de seus atributos e comportamento (métodos)

Por que projetar?

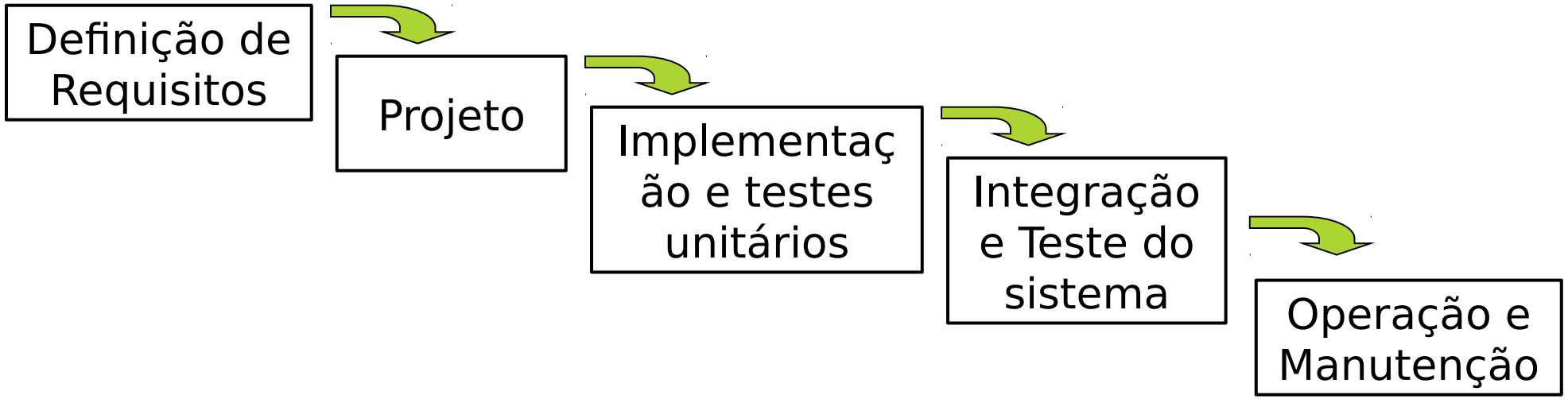
- Tão essencial quanto ter uma planta antes da construção de uma casa
  - Melhora a comunicação entre os membros da equipe
  - A equipe entende melhor o sistema
  - Permite analisar o sistema sobre vários aspectos
  - Facilita a programação e a manutenção
  - Diminui a possibilidade de erros

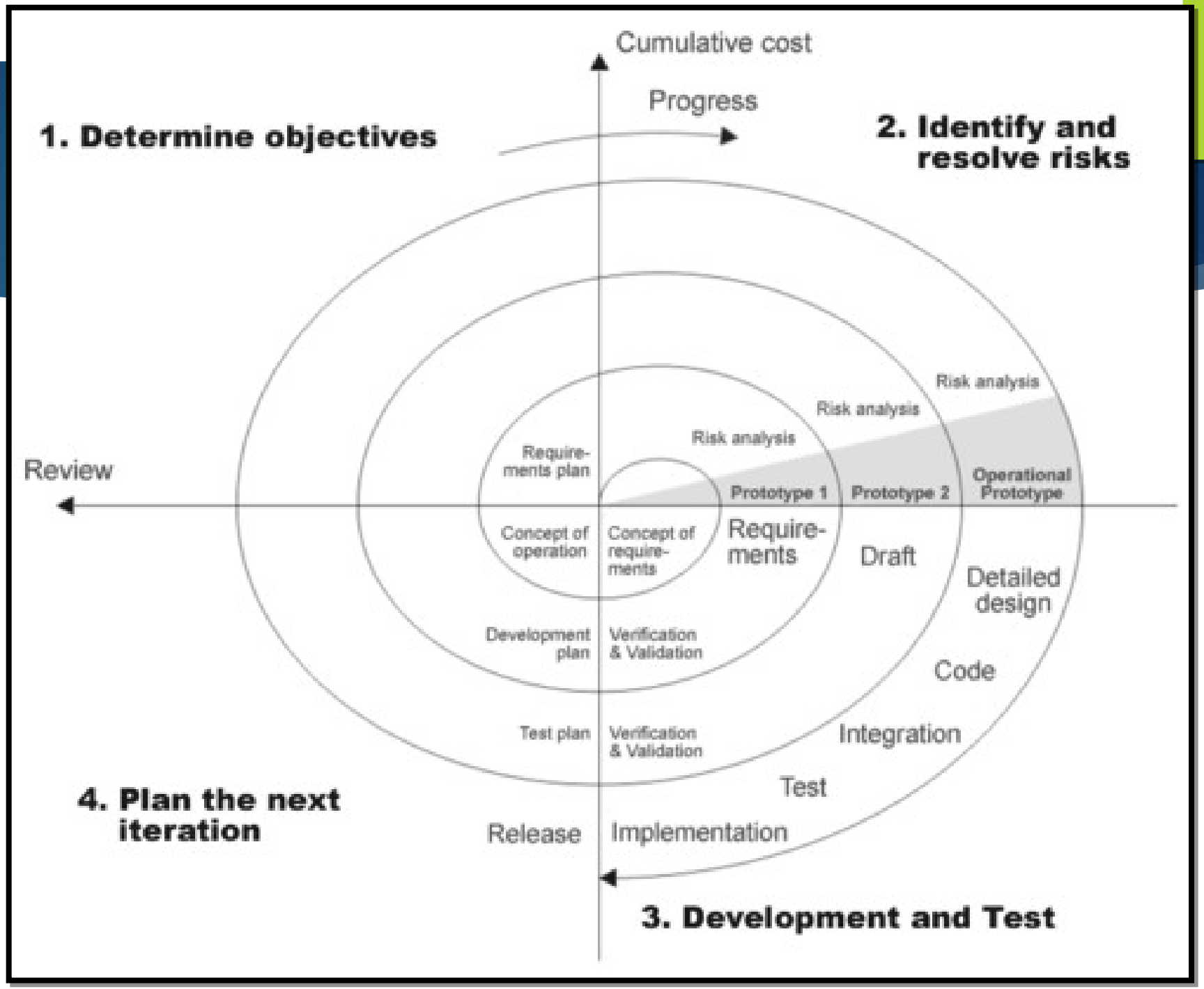
Projetar é Fundamental



Fases do Desenvolvimento

- Modelo Cascata
  - Define atividades sequenciais
  - Outras abordagens são baseadas nesta ideia




Cumulative cost
Progress
1. Determine objectives
2. Identify and resolve risks
Review
Risk analysis
Risk analysis
Risk analysis
Require-ments plan
Prototype 1
Prototype 2
Operational Prototype
Concept of operation
Concept of require-ments
Require-ments
Draft
Detailed design
Development plan
Verification & Validation
Code
Test plan
Verification & Validation
Integration
Test
4. Plan the next iteration
Release
Implementation
3. Development and Test

Linguagem UML

Linguagem UML

- UML (Linguagem de Modelagem Unificada)
- É uma notação gráfica (visual) para projetar sistemas OO
  - Não é uma linguagem de programação
- Define diagramas padronizados
- É extensível
- É complexa
  - Mas usaremos apenas um subconjunto da UML

De onde surgiu ?

- Da união de três metodologias de modelagem
  - Método de Booch - Grady Booch
  - Método OMT - Ivar Jacobson
  - Método OOSE - James Rumbaugh.

UML define Diagramas

- Tipos Principais de Diagramas
  - Estrutural
  - Comportamental

- Objetivos
  - Visualizar o sistema
  - Especificar estrutura e/ou comportamento
  - Guiar e documentar das decisões

Alguns Diagramas UML

- Diagramas Estruturais (Estáticos)
  - Diagrama de Classes
  - Diagramas de Objetos
  - Diagrama de Componentes, etc.
- Diagramas Dinâmicos
  - Diagrama de Caso de Uso
  - Diagrama de Sequência
  - Diagrama de Estados
  - Diagrama de Atividades
  - Diagrama de Colaboração, etc.

Alguns Diagramas UML

- Diagramas Estruturais (Estáticos)
  - **Diagrama de Classes**
  - Diagramas de Objetos
  - Diagrama de Componentes, etc.
- Diagramas Dinâmicos
  - Diagrama de Caso de Uso
  - **Diagrama de Sequência**
  - Diagrama de Estados
  - Diagrama de Atividades
  - Diagrama de Colaboração, etc.

Vamos trabalhar bastante com diagramas de classes nesta disciplina.

Diagrama de Classes

- Diagrama mais utilizado da UML
  - Serve de apoio para a maioria dos outros diagramas
- Define a estrutura das classes do sistema
- Permite a visualização das classes que compõem o sistema
- Representa
  - Atributos e métodos de uma classe
  - Os relacionamentos entre classes
- Apresenta uma visão estática de como as classes estão organizadas
  - Preocupação com a estrutura lógica

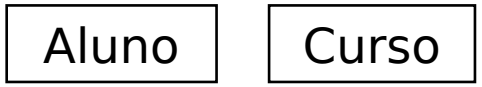
Diagrama de Classes

- O diagrama de classes oferece um ótimo exemplo do tipo de diagrama de estrutura e fornece um conjunto inicial de elementos de notação que todos os outros diagramas de estrutura usam.
- Um diagrama de classes pode oferecer três perspectivas, cada uma para um tipo de observador diferente. São elas:

Conceitual	Especificação	Implementação - *a mais utilizada de todas*
• Representa os conceitos do domínio em estudo. • Perspectiva destinada ao cliente.	• Tem foco nas principais interfaces da arquitetura, nos principais métodos, e não como eles irão ser implementados. • Perspectiva destinada as pessoas que não precisam saber detalhes de desenvolvimento, tais como gerentes de projeto.	• Aborda vários detalhes de implementação, tais como navegabilidade, *tipo* dos atributos, etc. • Perspectiva destinada ao time de desenvolvimento.

Diagrama de Classes

- Em UML, uma classe é representada por um retângulo com o nome da classe



- Habitualmente escreve-se o nome da classe no singular (nome de uma instância), com a 1ª letra em maiúscula
- Para se precisar o significado pretendido para uma classe, deve-se explicar o que é (e não é ...) uma instância da classe
  - Exemplo: “aluno é uma pessoa que está inscrita em um curso ministrado em uma escola. Uma pessoa que esteve no passado inscrita em um curso, mas não está atualmente inscrita em nenhum curso, não é um aluno.”
  - Em geral, o nome da classe não é suficiente para se compreender o significado da classe

Um exemplo de Diagrama de Classes

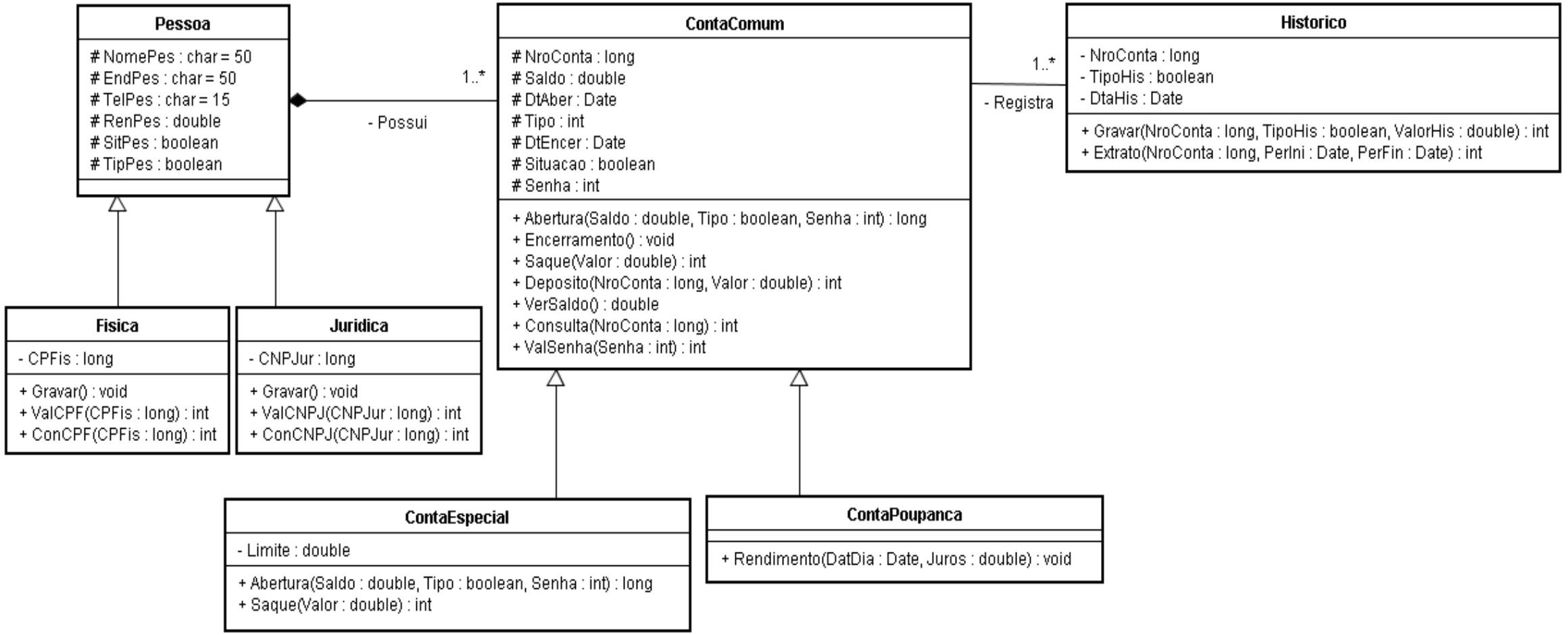
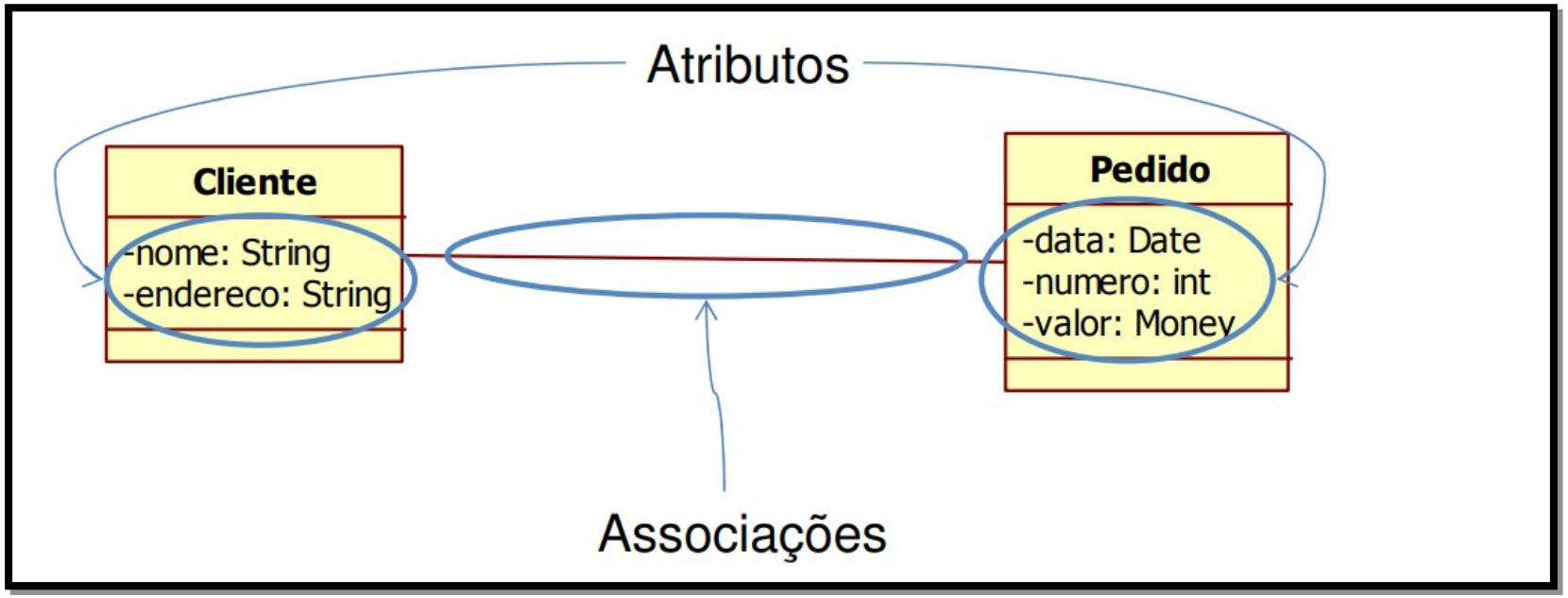


Diagrama de Classes



Atributos

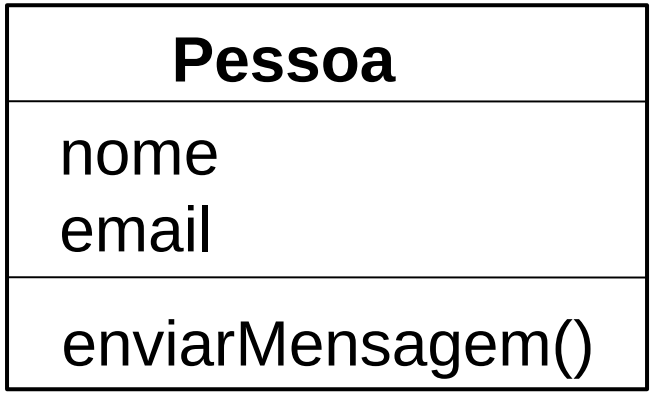
- Permite a identificação de cada objeto de uma classe
- Os valores dos atributos podem variar de instância para instância
- Uma classe não deve ter dois atributos com o mesmo nome
- Atributos devem conter o tipo de dados a ser armazenado
  - byte, boolean, int, double, char, String, etc.

Métodos

- São apenas declarados neste diagrama
  - Diagrama de Classes não define a implementação
- Outros diagramas permitem modelar o comportamento interno dos métodos
  - Diagrama de Sequência
  - Diagrama de Atividades

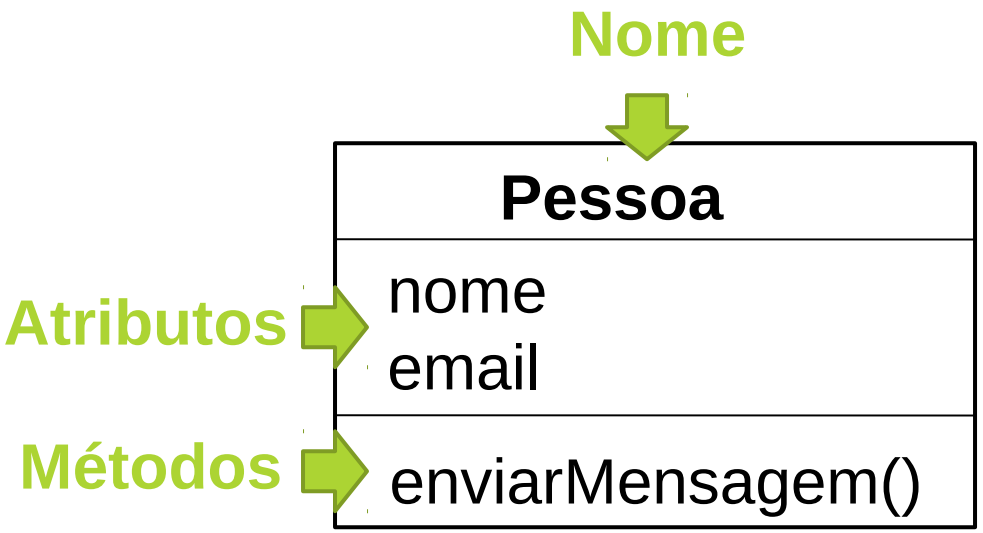
Representação de uma Classe

- Uma classe é representada por um retângulo com três divisões:
  - Nome da Classe
  - Atributos da Classe
  - Métodos da Classe



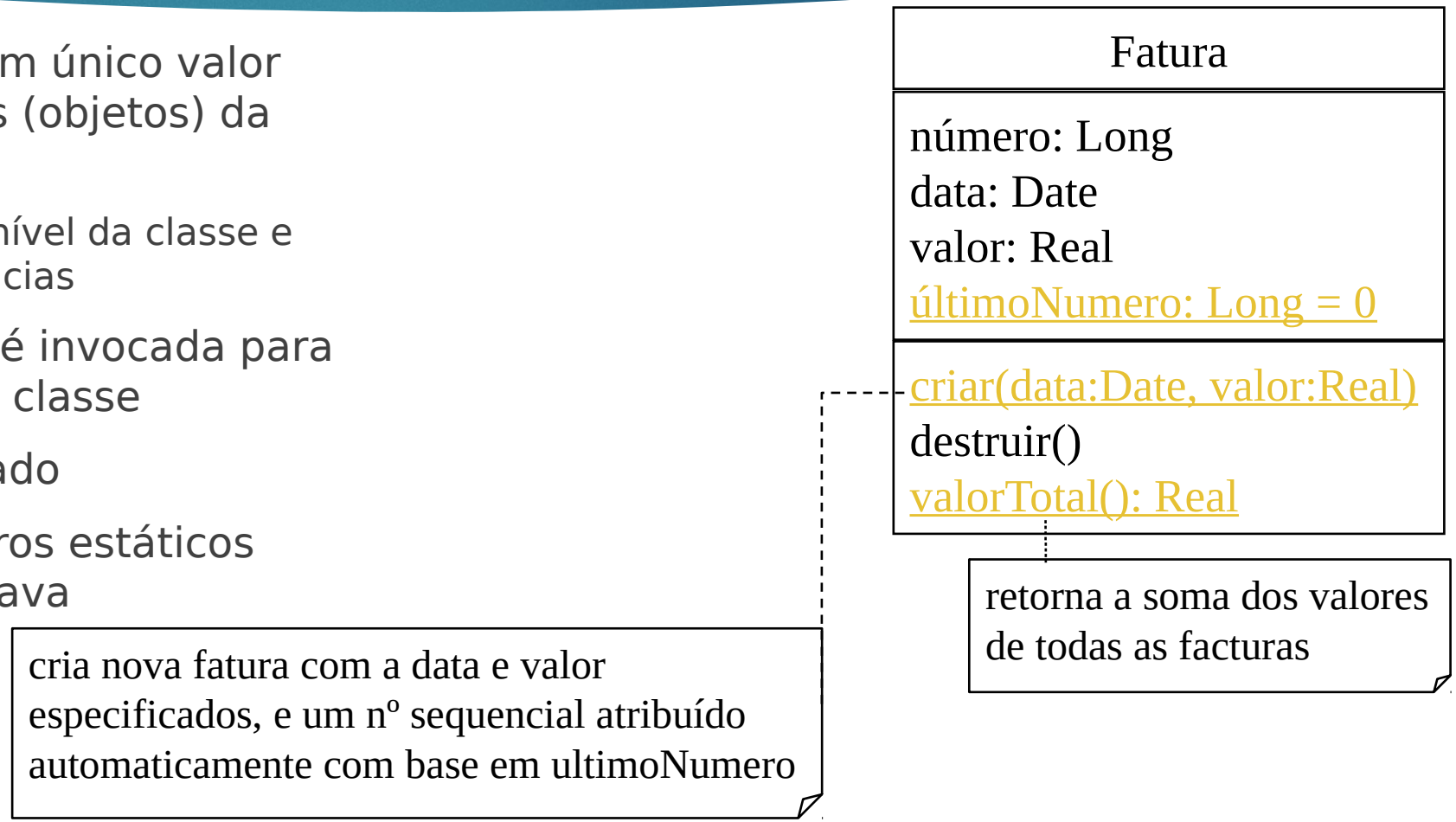
Representação de uma Classe

- Uma classe é representada por um retângulo com três divisões:
  - Nome da Classe
  - Atributos da Classe
  - Métodos da Classe



Atributos e operações estáticos (≠ de instância)

- Atributo estático: tem um único valor para todas as instâncias (objetos) da classe
  - valor está definido no nível da classe e não no nível das instâncias
- Operação estática: não é invocada para um objeto específico da classe
- Notação: nome sublinhado
- Correspondem a membros estáticos (static) em C++, C# e Java

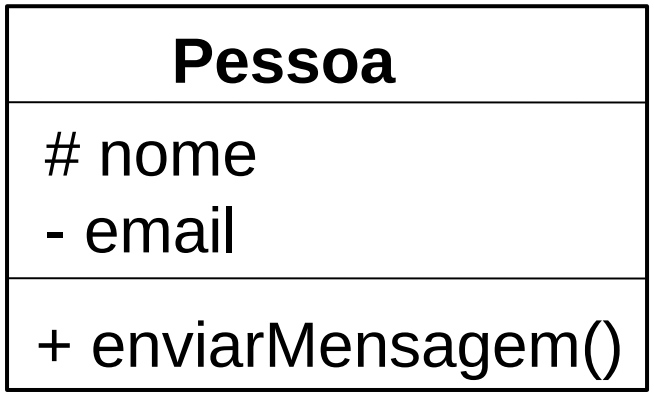


Tipos de visibilidade

- Pública (+)
  - O atributo ou método pode ser utilizado por qualquer classe
- Protegida (#)
  - Somente a classe ou sub-classes terão acesso
- Privada (-)
  - Somente a classe terá acesso

Tipos de visibilidade

- Pública (+)
  - O atributo ou método pode ser utilizado por qualquer classe
- Protegida (#)
  - Somente a classe ou sub-classes terão acesso
- Privada (-)
  - Somente a classe terá acesso



Relacionamentos

- Classes possuem relacionamentos entre elas
  - Compartilham informações
  - Colaboram umas com as outras
- Principais tipos de relacionamentos
  - Associação
  - Agregação / Composição
  - Herança
  - Dependência

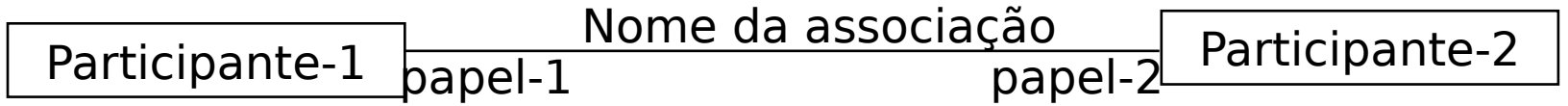
Comunicação entre Objetos

- Conceitualmente, objetos se comunicam através da troca de mensagens.
- Mensagens definem:
  - O nome do serviço requisitado
  - A informação necessária para a execução do serviço
  - O nome do requisitante.
- Na prática, mensagens são implementadas como chamadas de métodos
  - Nome = o nome do método
  - Informação = a lista de parâmetros
  - Requisitante = o método que realizou a chamada

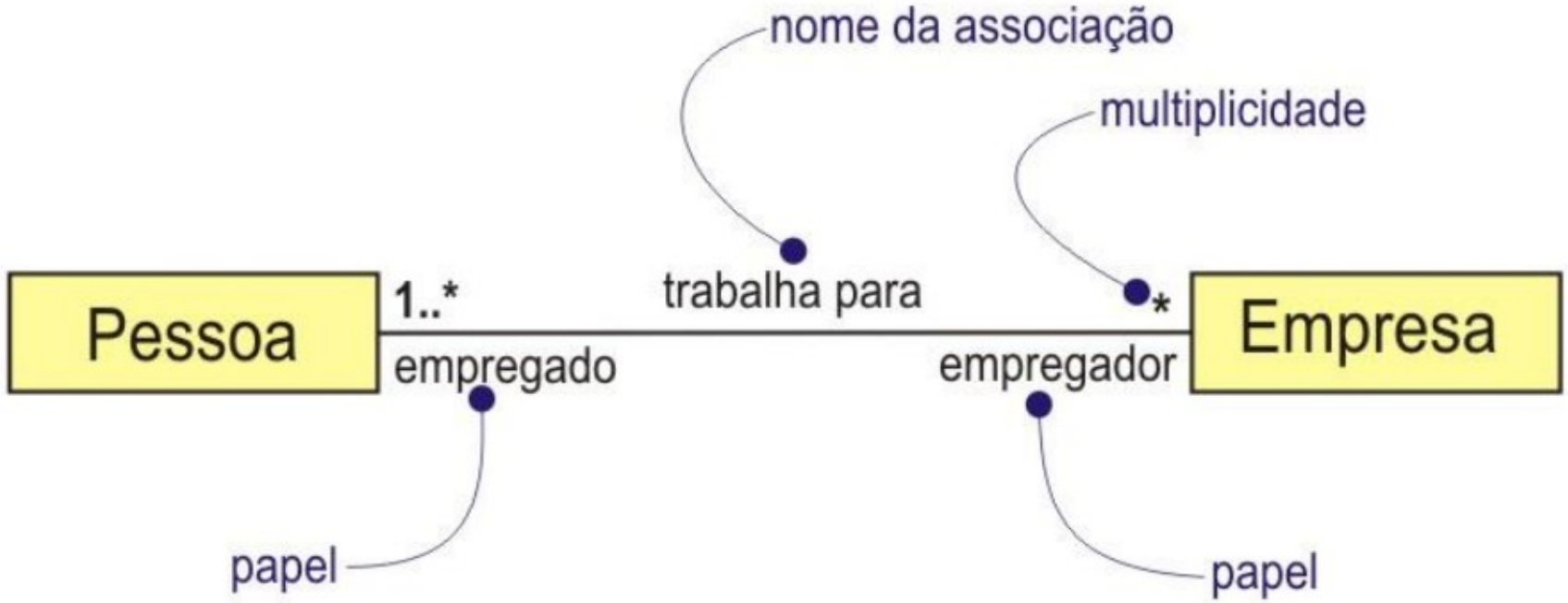
Associações

- Descreve um vínculo entre duas classes
  - Chamado Associação Binária

- Determina que as instâncias de uma classe estão de alguma forma ligadas às instâncias da outra classe



Associações

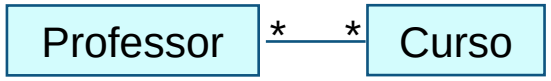


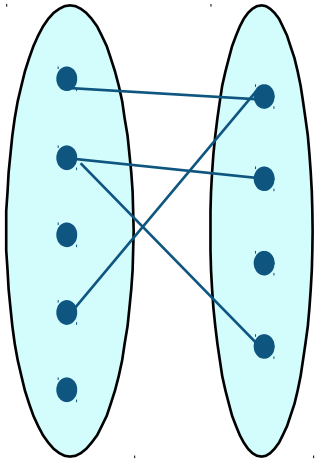
Associações

- Nome: opcional
- Papéis
  - Papéis das classes que estão relacionadas pela associação
  - O papel da classe A é o nome do atributo que a classe B possui que guarda o objetivo da classe A
- Multiplicidades
  - Quantidades de objetos associados a um papel
- Navegabilidade
  - Indica a direção da relação entre as classes

Multiplicidade

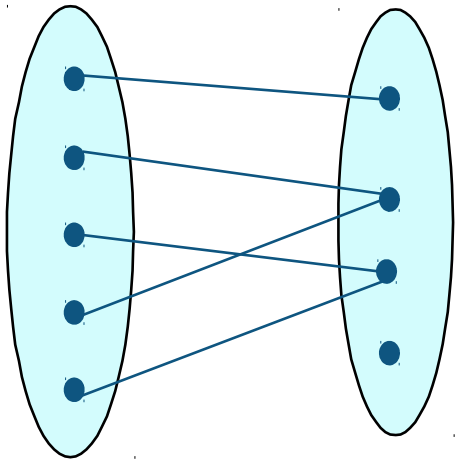
Muitos-para-Muitos



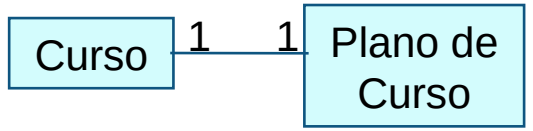


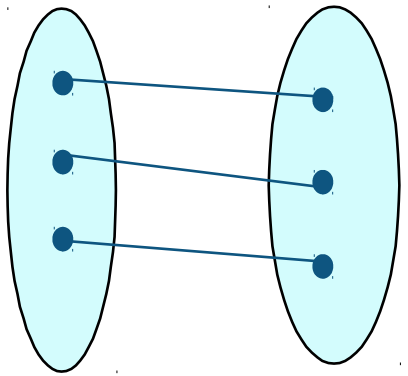
(sem restrições)

Muitos-para-1



1-para-1

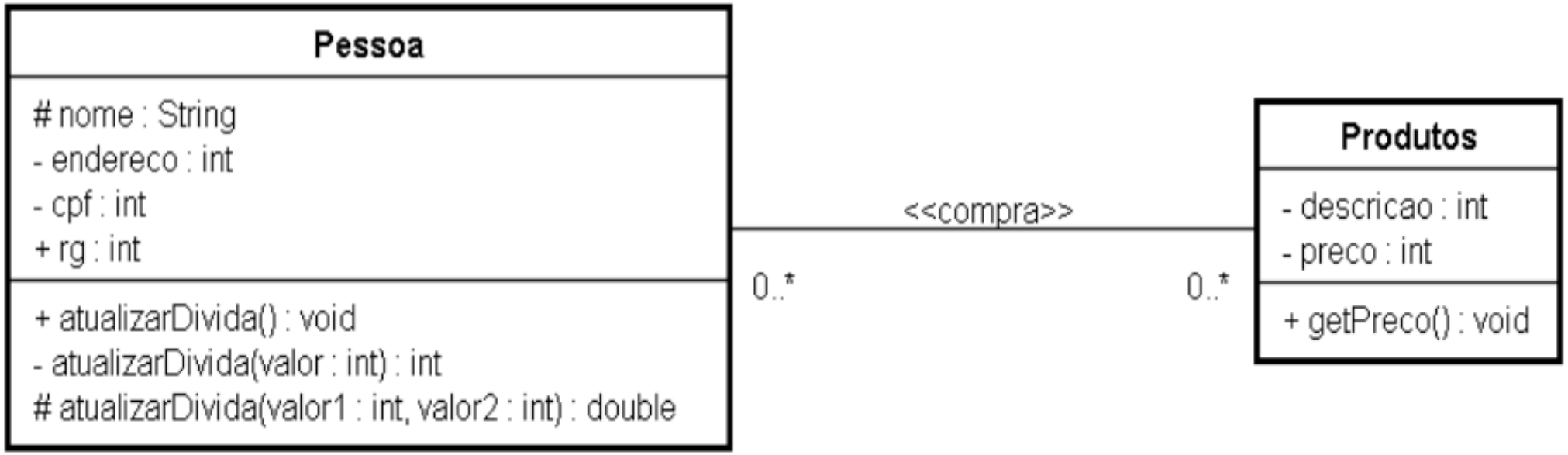




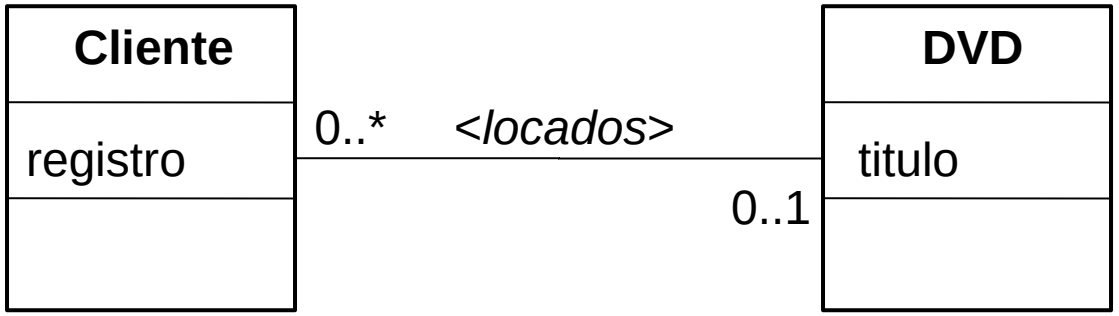
Multiplicidade

0..1	No máximo um. Indica que os Objetos da classe associada não precisam obrigatoriamente estar relacionados.
1..1	Um e somente um. Indica que apenas um objeto da classe se relaciona com os objetos da outra classe.
0..*	Muitos. Indica que podem haver muitos objetos da classe envolvidos no relacionamento
1..*	Um ou muitos. Indica que há pelo menos um objeto envolvido no relacionamento.
3..5	Valores específicos.

Exercício: descreva o diagrama

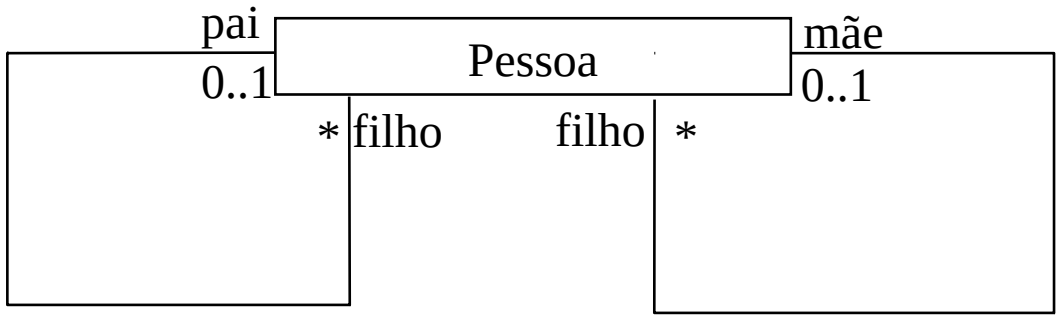


Representação de Associação



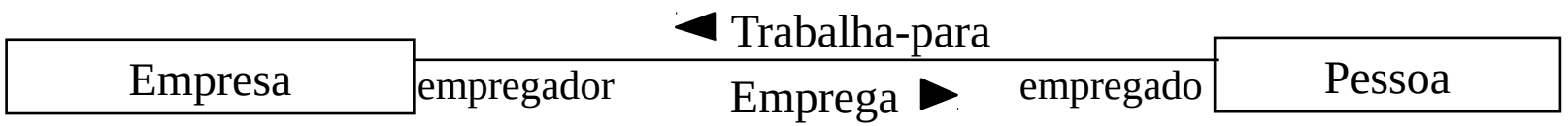
Associação reflexiva

- Pode-se associar uma classe com ela própria (em papéis diferentes)



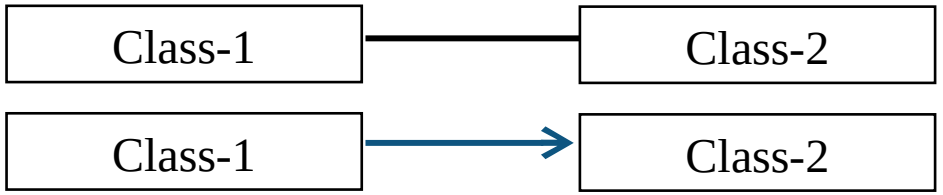
Nome de associação

- A indicação do nome é opcional
- O nome é indicado no meio da linha que une as classes participantes
- Pode-se indicar o sentido em que se lê o nome da associação



Associações unidireccionais

- As associações são classificadas quanto à navegabilidade em:
  - bidirecionais (normal)
  - unidirecionais



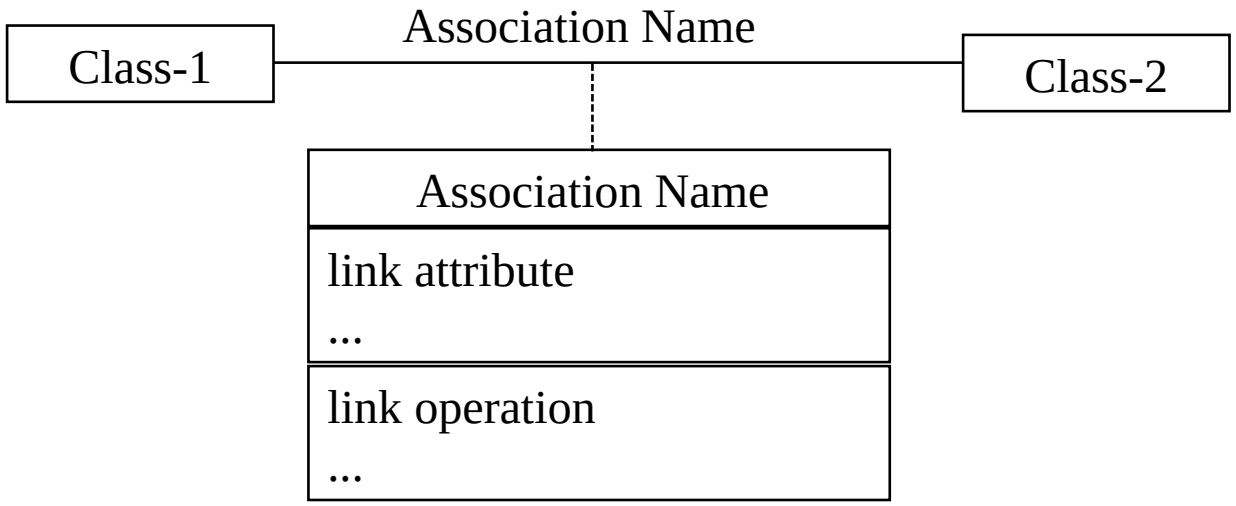
um objeto da classe 1 tem a responsabilidade de dar o(s) objeto(s) correspondente(s) da classe 2 (nível de especificação)

ou

um objeto da classe 1 tem apontador(es) para o(s) objeto(s) correspondente(s) da classe 2 (nível de implementação)

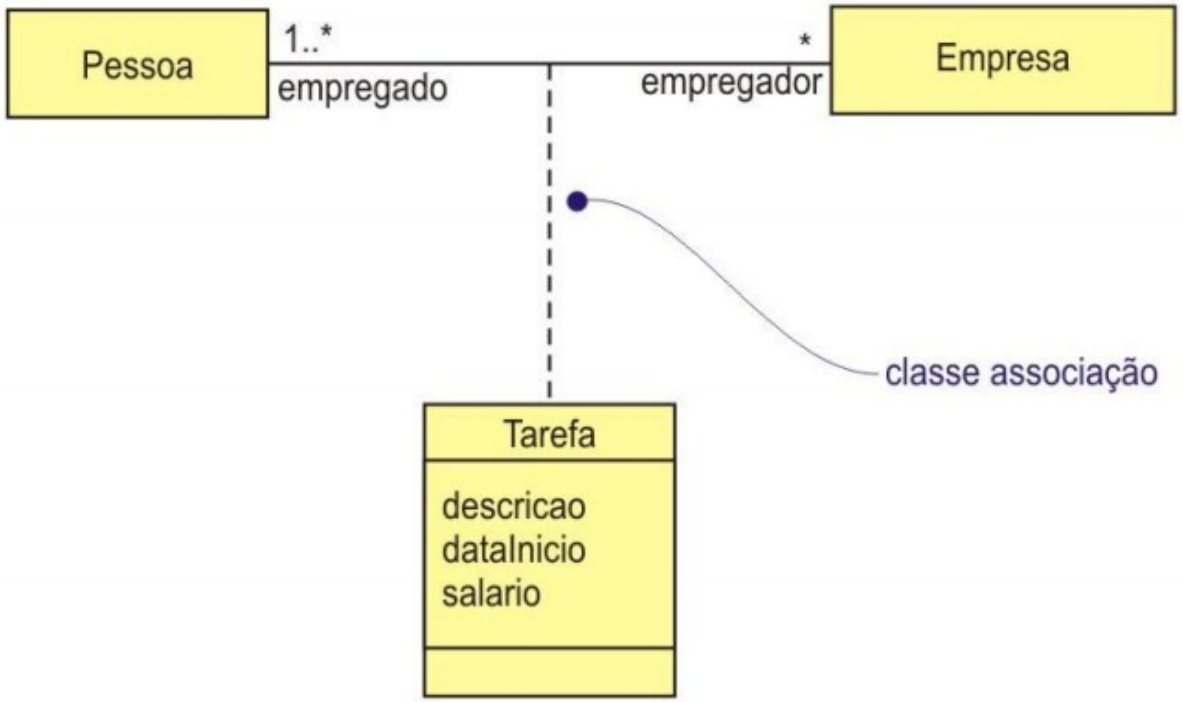
Classe de Associação

- reúne as propriedades de associação e classe
- o nome pode ser colocado na associação ou na classe, mas a semântica é a mesma
- não é possível repetir combinações de objetos das classes participantes na associação



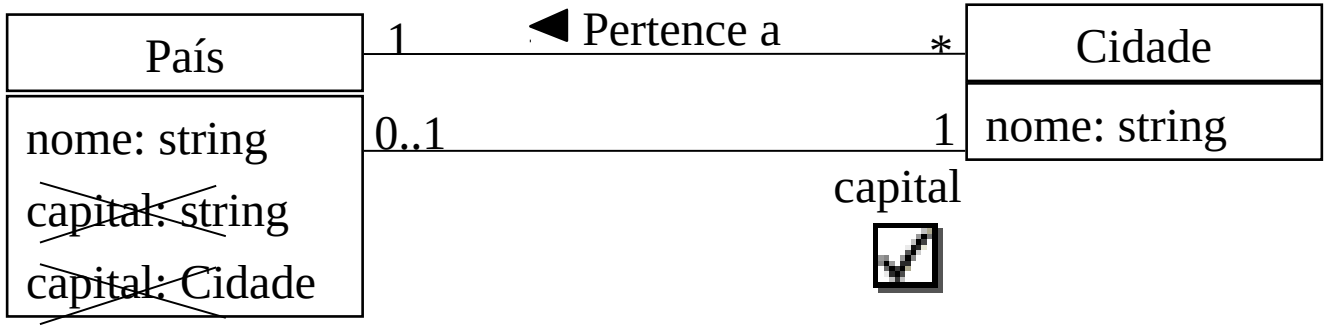
Classe de Associação

- Numa relação de associação entre classes, a associação pode também ter os seus próprios atributos (e eventualmente operações), devendo ser, por conseguinte, modelada também como uma classe.
- Este tipo de classes designa-se por classe-associação



Atributos versus Associações

- Uma propriedade que designa um objeto de uma classe presente no modelo, deve ser modelada como uma associação e não como um atributo
- Exemplo:



Agregação

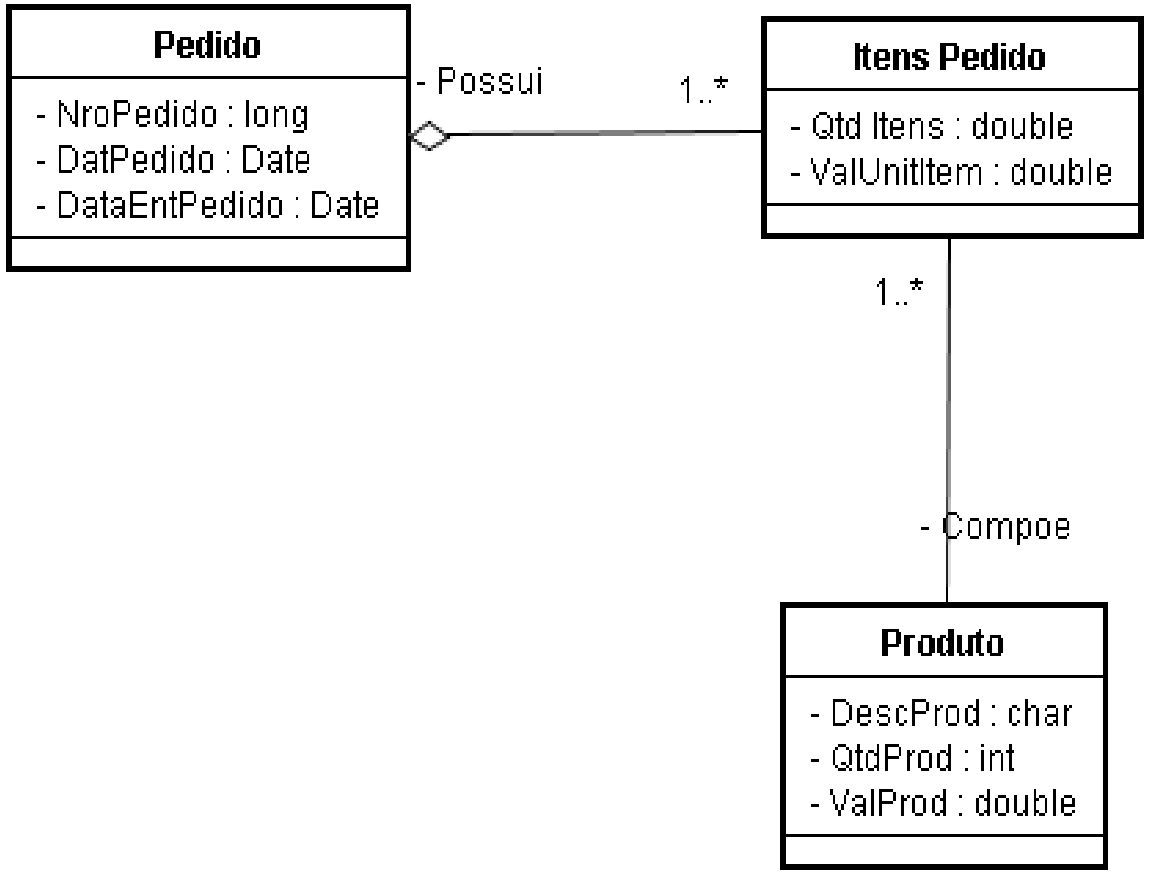
- Tipo especial de associação
- Demonstra que as informações e um objeto precisam ser complementadas por um objeto de outra classe
- Associação Todo-Parte
  - objeto-todo
  - objeto-parte
- O todo existe sem as partes e as partes existem sem o todo

Representação de Agregação

- Um losango na extremidade da classe que contém os objetos-todo
- Agregação é conhecida como “parte-de” ou relacionamento de conteúdo

Exercício

- Descreva com suas palavras o seguinte diagrama de classe:



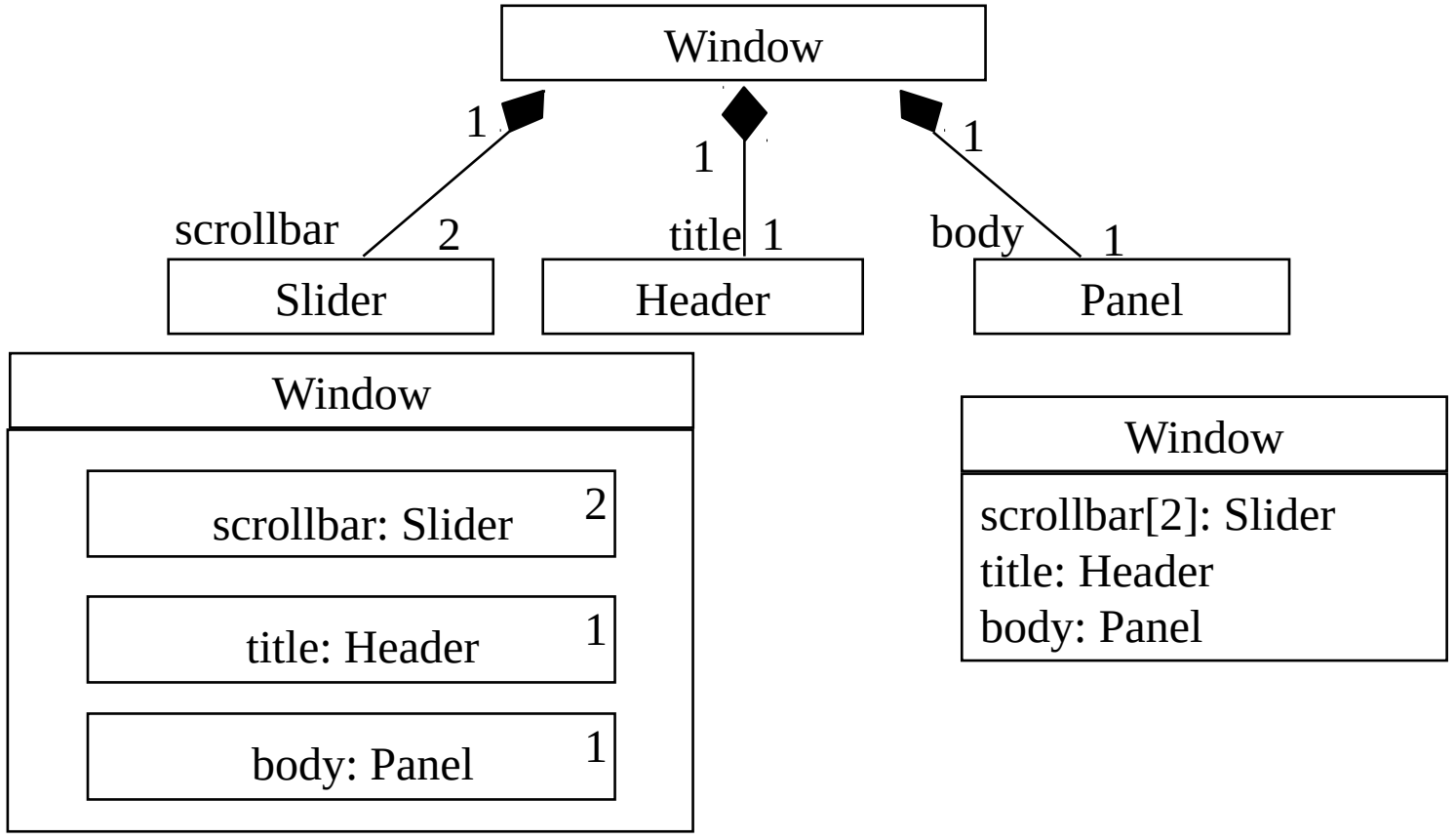
Composição

- Uma variação do tipo agregação
- Representa um vínculo mais forte entre objetos-todo e objetos-parte
- Objetos-parte **têm que pertencer** ao objeto-todo
  - O todo não existe (ou não faz sentido) sem a parte

Representação da Composição

- Um losango preenchido, e da mesma forma que na Agregação, deve ficar ao lado do objeto-todo

Composição: notações alternativas

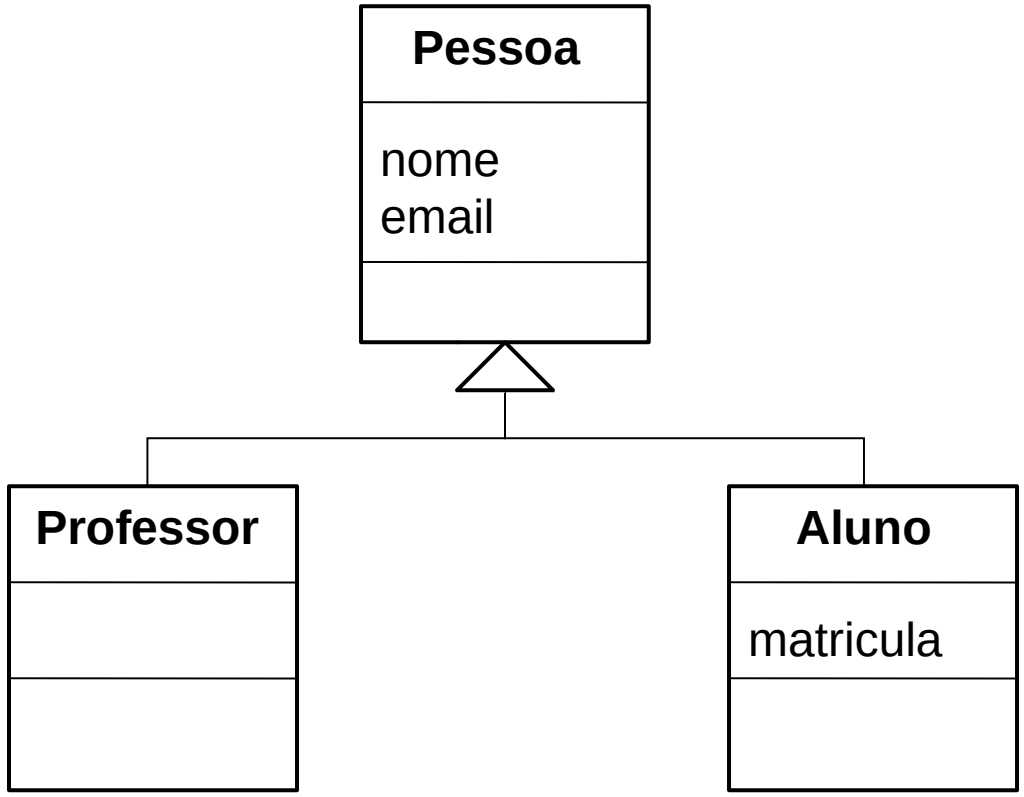


(sub-objetos no compartimento dos atributos)

Especialização / Generalização

- Identificar classes-mãe (gerais) e classes-filhas (especializadas)
- Atributos e métodos definidos na classe-mãe são herdados pelas classes-filhas
- Permite métodos polimórficos
- Classes com características semelhantes

Especialização / Generalização

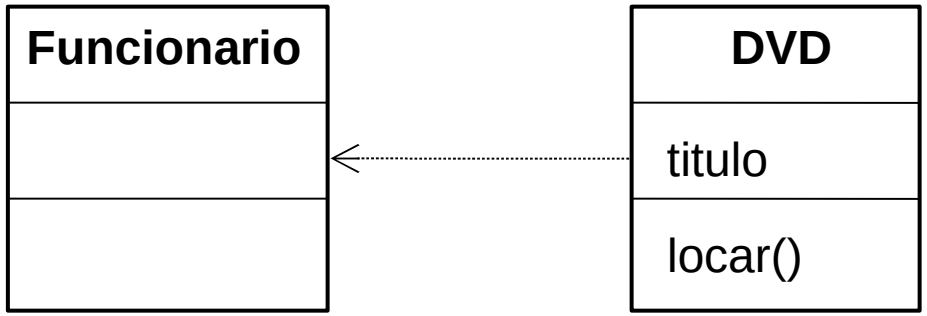


Dependência

- Tipo menos comum de relacionamento
- Identifica um baixo grau de dependência de uma classe em relação a outra

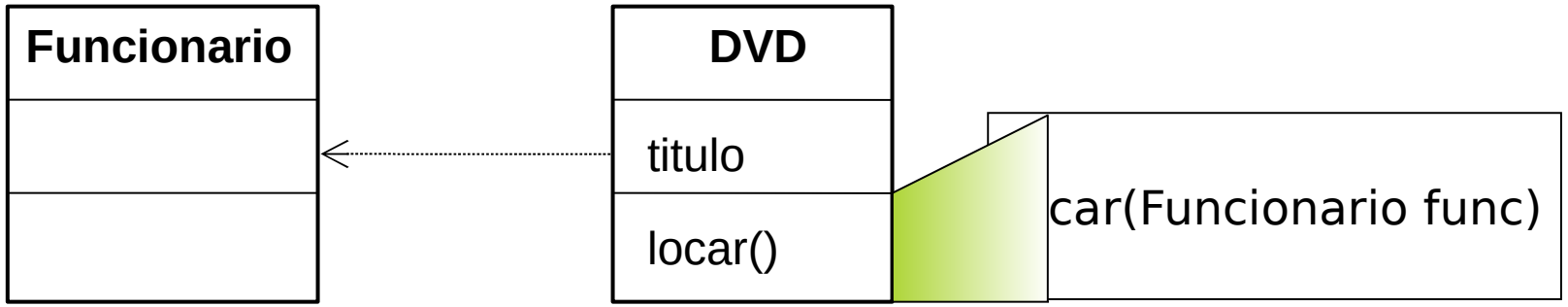
Dependência

- Representado por uma reta tracejada entre duas classes
- Uma seta na extremidade indica o dependente

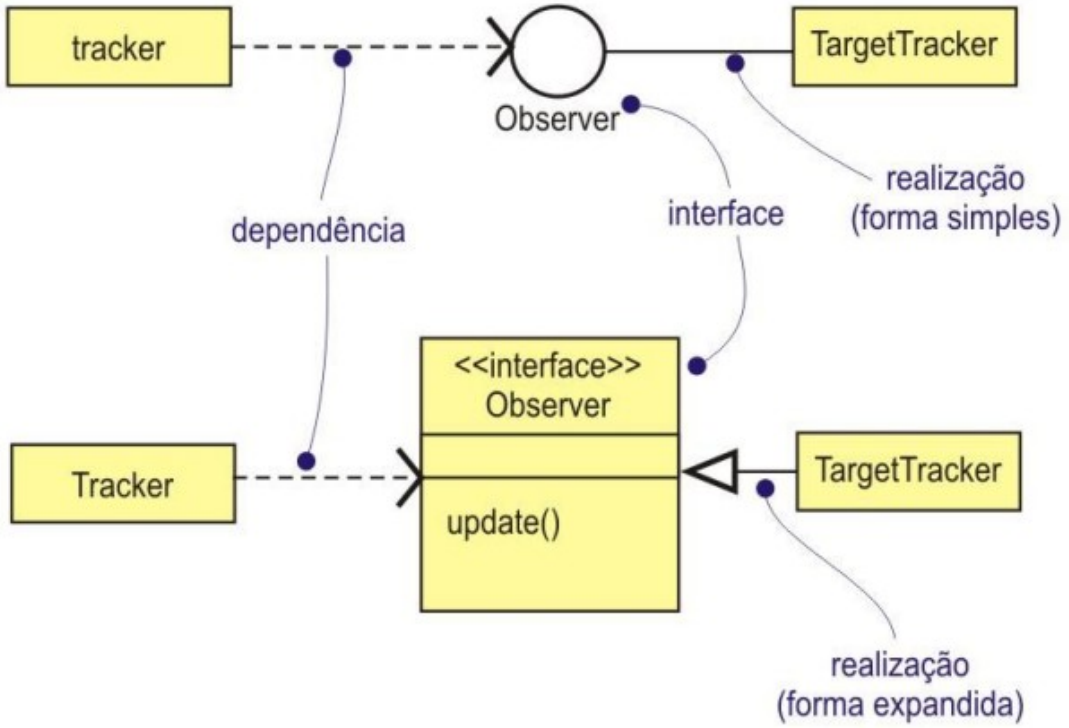


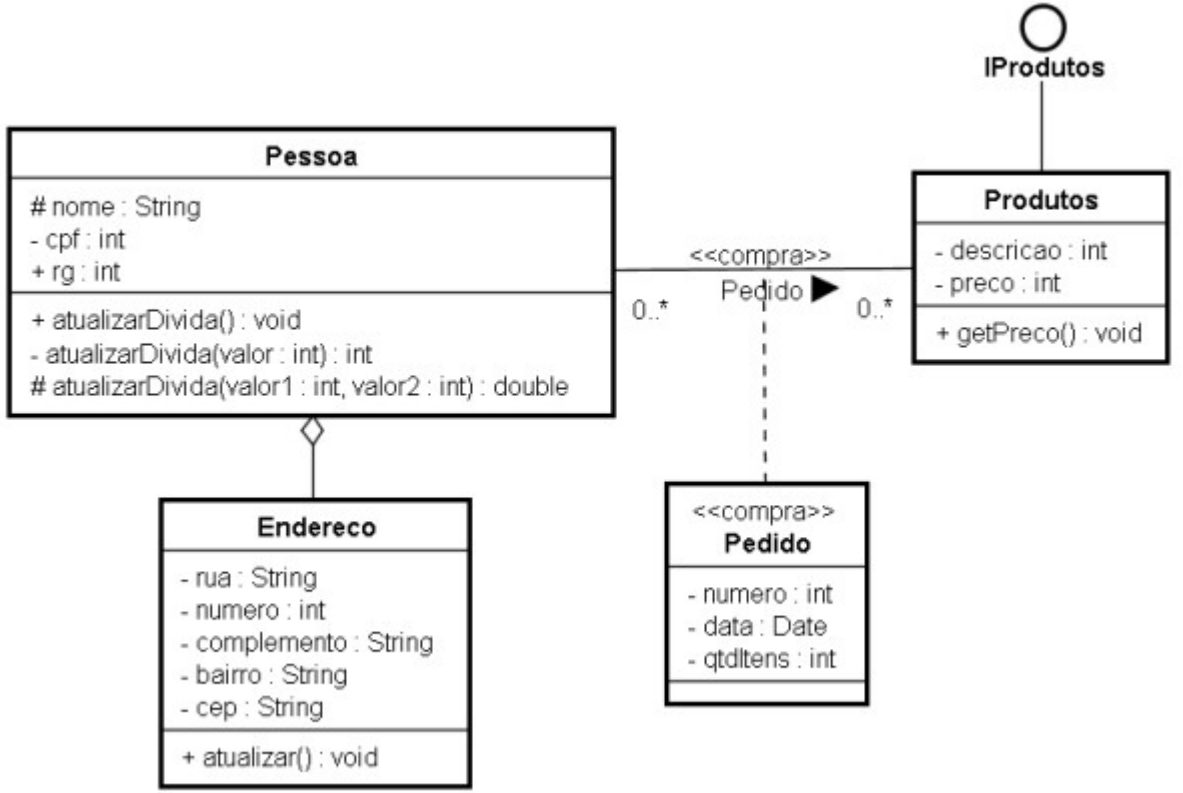
Dependência

- Representado por uma reta tracejada entre duas classes
- Uma seta na extremidade indica o dependente



Interfaces


tracker
Observer
TargetTracker
dependência
interface
realização
(forma simples)
<<interface>>
Observer
Tracker
update()
TargetTracker
realização
(forma expandida)


IProdutos
Pessoa
nome : String
- cpf : int
+ rg : int
+ atualizarDivida() : void
- atualizarDivida(valor : int) : int
atualizarDivida(valor1 : int, valor2 : int) : double
<<compra>>
Pedido
0..*
0..*
Produtos
- descricao : int
- preco : int
+ getPreco() : void
Endereco
- rua : String
- numero : int
- complemento : String
- bairro : String
- cep : String
+ atualizar() : void
<<compra>>
Pedido
- numero : int
- data : Date
- qtdItens : int

Exercício
Questão ENADE

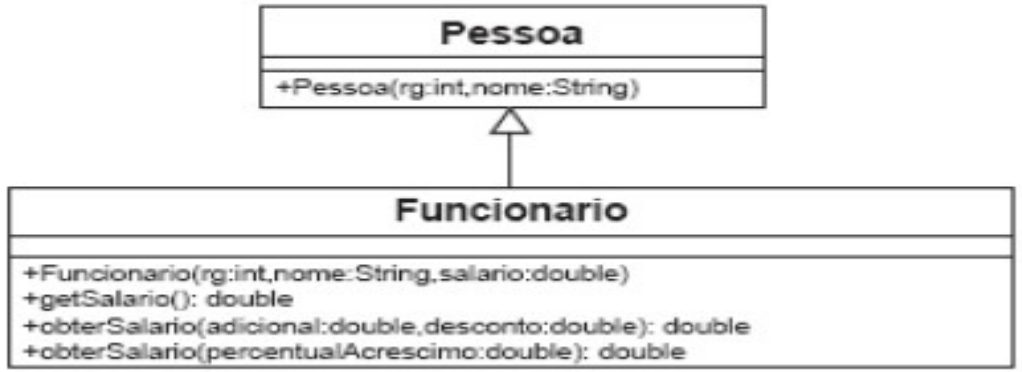
- UML é uma linguagem padrão para desenvolver e documentar projetos de software e permite que desenvolvedores visualizem os produtos de seus trabalhos em diagramas padronizados. Ela surgiu como uma proposta de ser uma linguagem para modelagem de dados que usava diversos artefatos para representar o modelo de negócio e um desses artefatos é o diagrama de classes. (PRESSMAN, R., 2006)
- Se um projeto não tem a documentação apropriada ou se está com a documentação desatualizada, uma opção é a engenharia reversa que possibilita mapear códigos para diagramas UML. A seguir, é apresentado um código na linguagem de programação JAVA.

```
1  package default;
2
3  public class Funcionario extends Pessoa {
4      private double salario;
5
6      public Funcionario(int rg, String nome,
7                         double salario) {
8          super(rg, nome);
9          this.salario= salario;
10     }
11     public double getSalario() {
12         return salario;
13
14     }
15     public double obterSalario(double
16                         percentualAcrescimo) {
17         double salarioReajustado = salario +
18               salario * percentualAcrescimo / 100;
19         return salarioReajustado;
20
21     }
22     public double obterSalario(double
23                   adicional, double desconto){
24         return this.getSalario() + adicional - desconto;
25
26     }
27 }
```

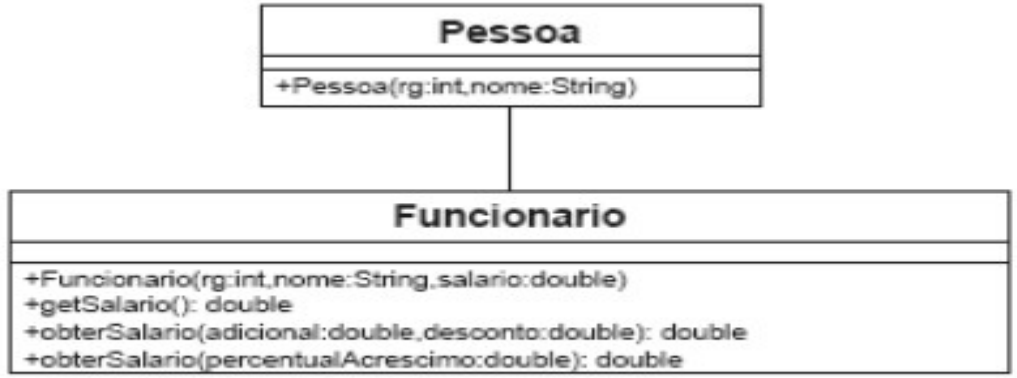
Exercício
Questão ENADE

Utilizando a engenharia reversa nesse trecho de código, o digrama UML de classes correspondente é:

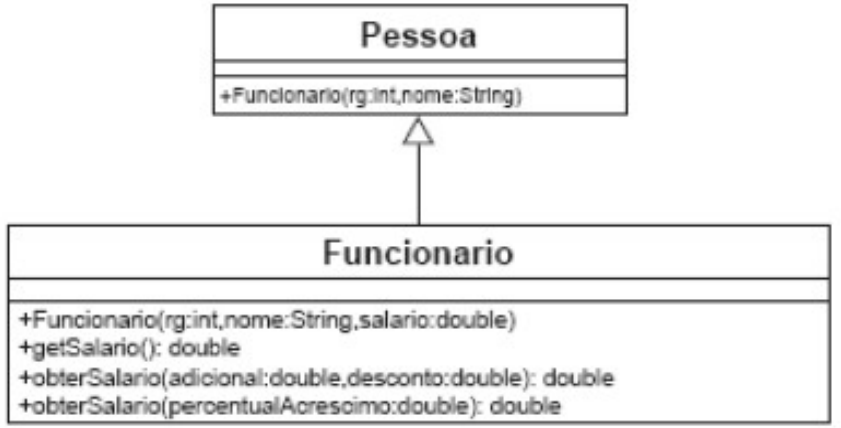
A



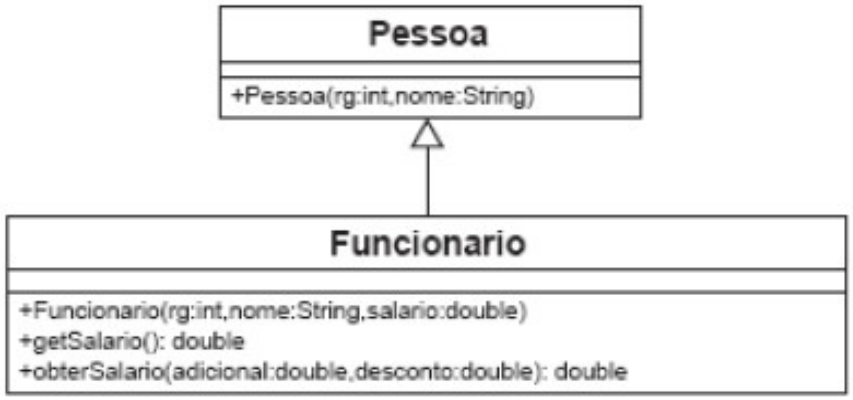
B



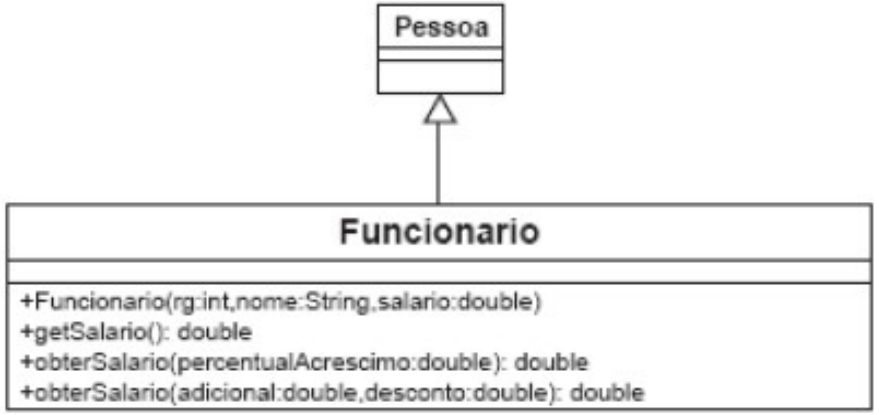
C



D



E



Exercício

- Uma loja De roupas possui um sistema capaz de controlar a venda e o estoque. Cada roupa possui um código de barras, um tamanho e o número de exemplares que a loja possui daquela roupa.
- Os clientes da loja são cadastrados pelo nome
- Faça um diagrama de classe que modele um sistema capaz de respondendo as perguntas abaixo:
  - Quais foram as roupas compradas por um cliente?
  - Quais são os cliente que já compraram uma determinada roupa?
  - Quantos exemplares possuem de uma determinada roupa?

Exercício

- Elabore um diagrama de classes para um sistema de ponto de vendas
  - R01. O gerente deve fazer login com um ID e senha para iniciar e finalizar o sistema;
  - R02. O caixa (operador) deve fazer login com um ID e senha para poder utilizar o sistema;
  - R03. Registrar a venda em andamento – os itens comprados;
  - R04. Exibir a descrição e preço e do item registrado;
  - R05. Calcular o total da venda corrente;
  - R06. Tratar pagamento com dinheiro – capturar a quantidade recebida e calcular o troco;
  - R07. Tratar pagamento com cartão de crédito – capturar a informação do cartão através de um leitor de cartões ou entrada manual e autorizar o pagamento utilizando o serviço de autorização de crédito (externo) via conexão por modem;
  - R08. Tratar pagamento com cheque – capturar o número da carteira de identidade por entrada manual e autorizar o pagamento utilizando o serviço de autorização de cheque (externo) via conexão por modem;
  - R09. Reduzir as quantidades em estoque quando a venda é confirmada;
  - R10. Registrar as vendas completadas;
  - R11. Permitir que diversas lojas utilizem o sistema, com catálogo de produtos e preços unificado, porém estoques separados;

Referências

- Aula do professor Eduardo Figueiredo
- BOOCH, G., RUMBAUGH, J., JACOBSON, I. **UML, Guia do Usuário**. Rio de Janeiro: Campus, 2000.