UML
(Linguagem Modelagem Unificada)

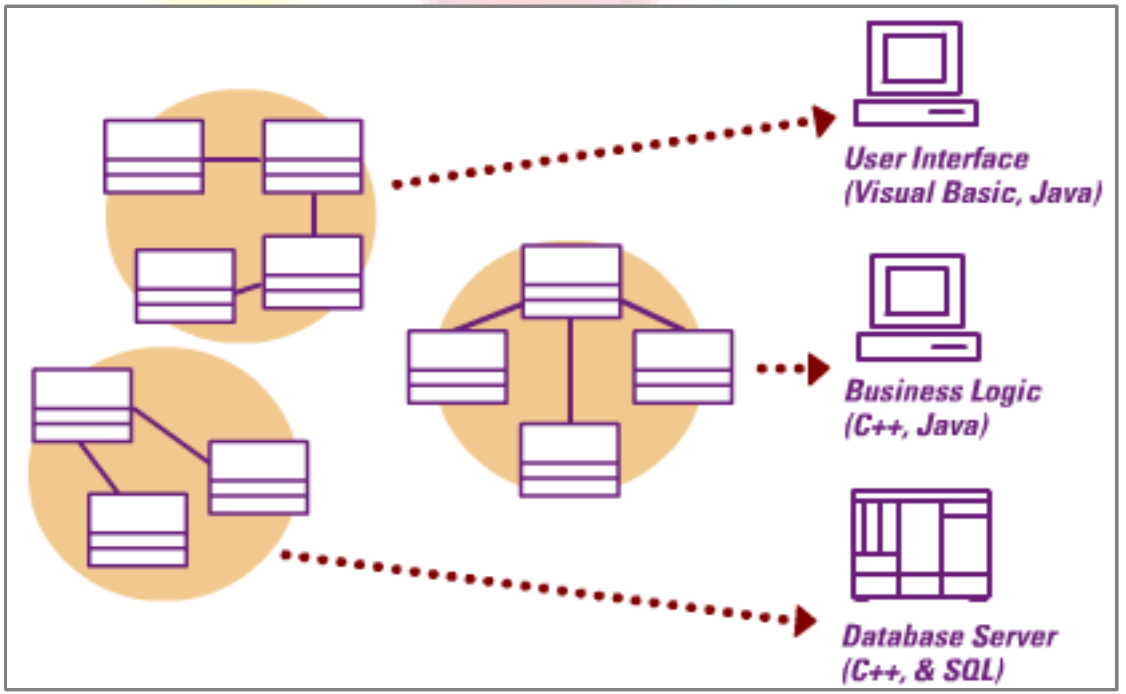
João Paulo Q. dos Santos
joao.queiroz@ifrn.edu.br

Roteiro

- A importância da UML para projetar sistemas.
- Principais características do diagrama de classes e de sequência.

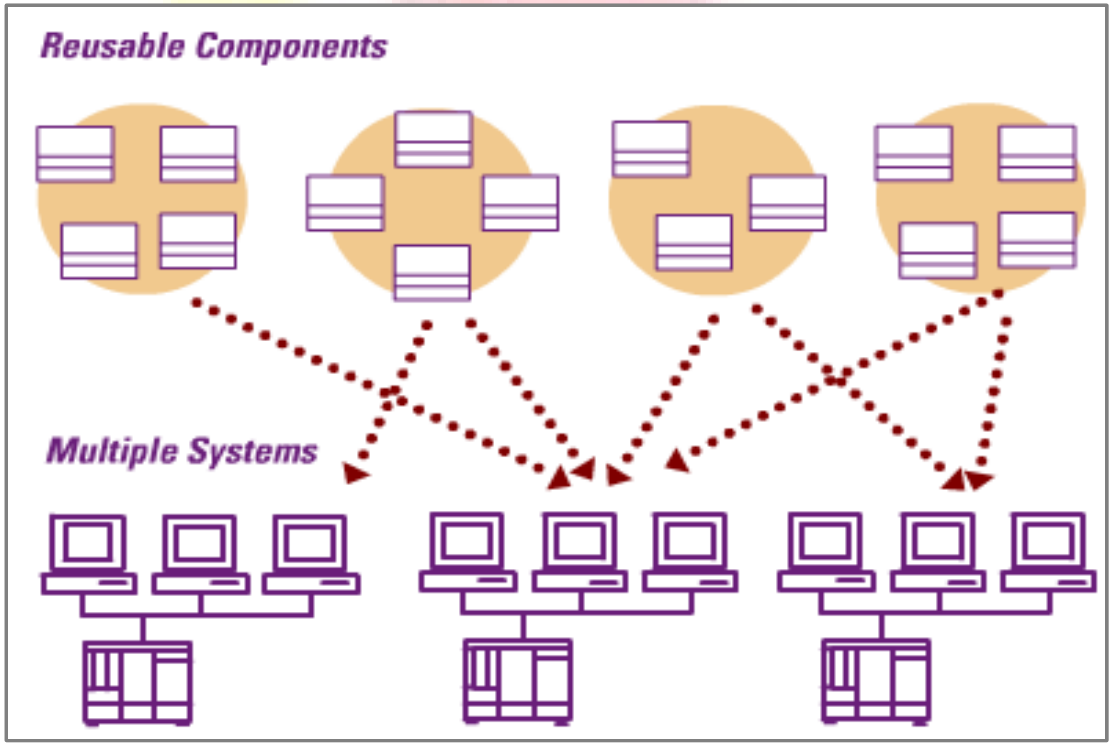
Motivação

- Um modelo dever ser criado independentemente de sua implementação. A qualquer momento uma implementação pode ser trocada por outra sem afetar o modelo.



Motivação

- A utilização de uma modelagem visual facilita a visualização, e, por conseguinte, a criação de um melhor modelo (mais flexível, mais robusto e principalmente mais reutilizável).



UML

- UML é uma linguagem para visualização, especificação, construção e documentação de artefatos de um software em desenvolvimento.
- UML permite modelar:
  1. elementos;
  2. relacionamentos;
  3. mecanismos de extensibilidade;
  4. diagramas.

1. Elementos:
   - estruturais

     classes, interfaces, componentes
   - comportamentais

     interações, máquinas de estado
   - grupos de elementos

     pacotes, subsistemas
   - outros

     anotações

UML

2. Relacionamentos
   - Dependências, Associações, Generalizações, Implementações (realization)
3. Mecanismos de Extensibilidade
   - Estereótipos (<<public>>)
   - Tagged value ({version = 1.1})
   - Regras (constraints)

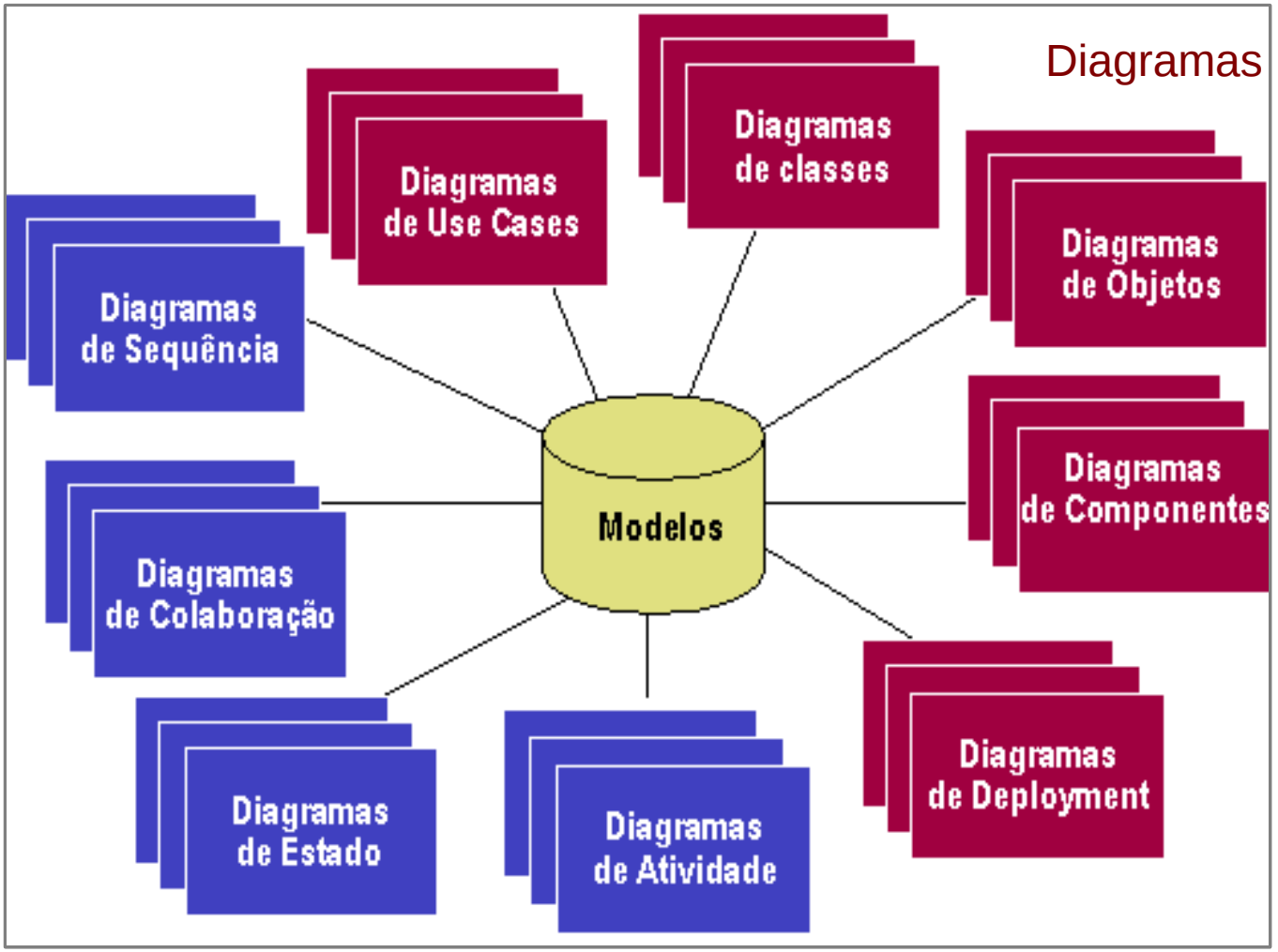
UML

4. Diagramas

- Um modelo é uma descrição completa do sistema em uma determinada perspectiva.
- Um modelo é representado por um ou mais diagramas. Desta forma, um diagrama pode ser visto como uma visão dentro de um modelo.
- Um diagrama pode ser representado de várias formas, dependendo de quem irá interpretá-lo.

UML

Diagramas Estruturais



Diagramas Comportamentais

Diagramas

- Um diagrama provê uma parcial representação do sistema. Ele ajuda a compreender a arquitetura do sistema em desenvolvimento. Nesta seção focaremos a construção destes artefatos que auxiliam nossa interpretação.
- Tipos de diagramas:
  - Estáticos:
    - Diagramas de Use Cases
    - Diagramas de Classes
    - Diagramas de Pacotes
  - Dinâmicos:
    - Diagramas de Interação
      - Diagramas de Sequência
      - Diagramas de Colaboração
    - Diagramas de Estado (*Statechart*)
    - Diagramas de Atividade

Diagrama de classes

- Objetivo

  Descrever os vários tipos de objetos no sistema e o relacionamento entre eles.

- Perspectivas

  Um diagrama de classes pode oferecer três perspectivas, cada uma para um tipo de observador diferente. São elas:

  - Conceitual

    Representa os conceitos do domínio em estudo.

    Perspectiva destinada ao cliente.

  - Especificação

    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.

  - Implementação

    Aborda vários detalhes de implementação, tais como navegabilidade, tipo dos atributos, etc.

    Perspectiva destinada ao time de desenvolvimento.

Diagrama de classes

PERSPECTIVA CONCEITUAL

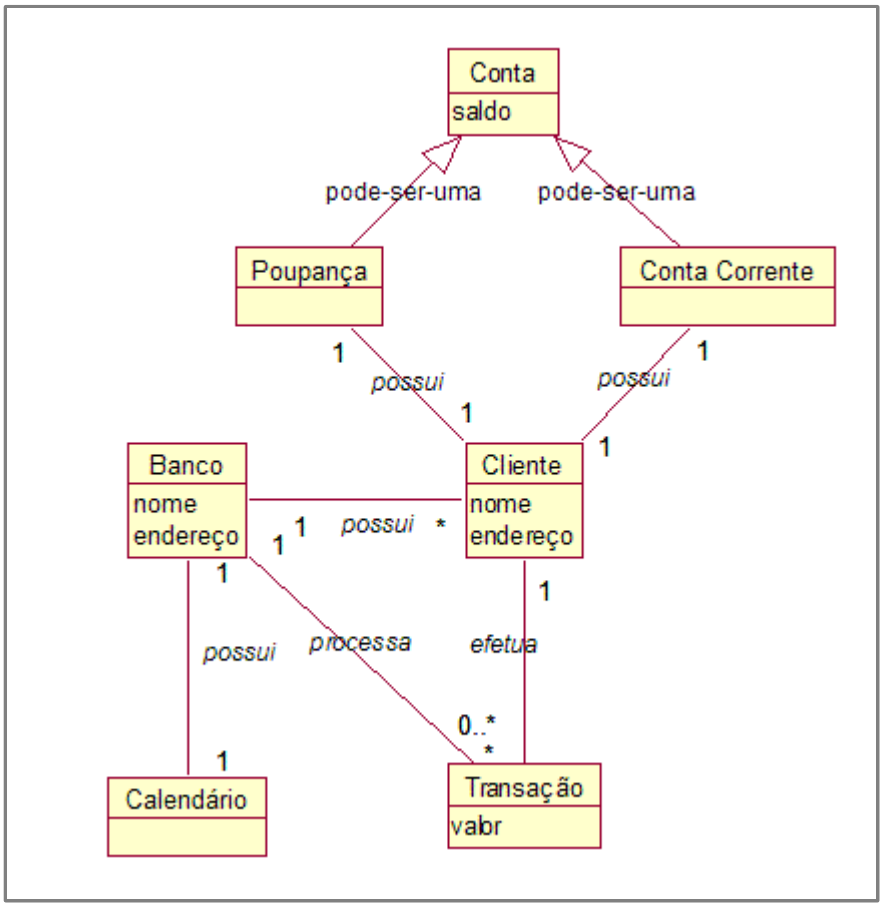


Diagrama de classes

PERSPECTIVA DE ESPECIFICAÇÃO

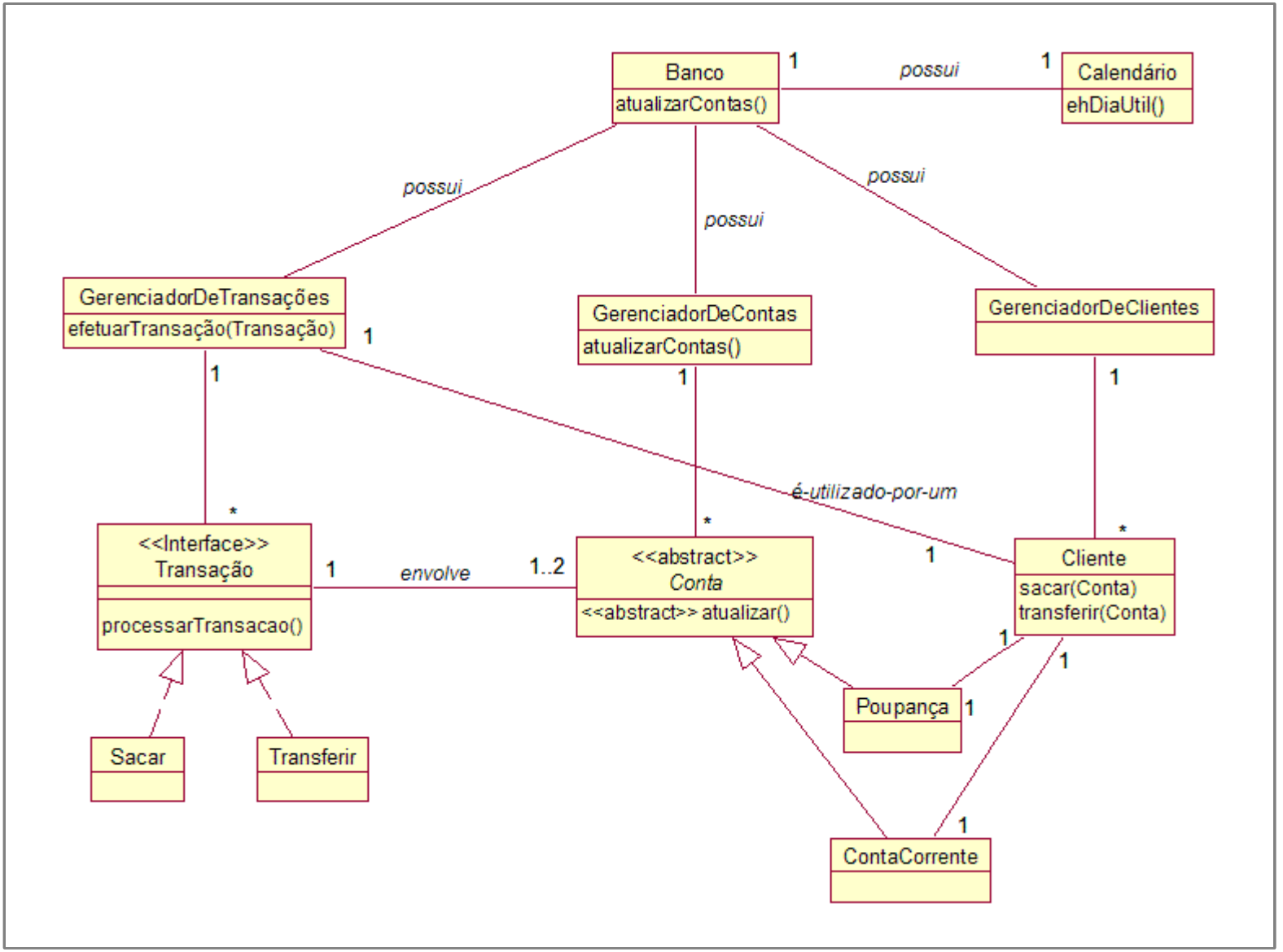


Diagrama de classes

PERSPECTIVA DE IMPLEMENTAÇÃO

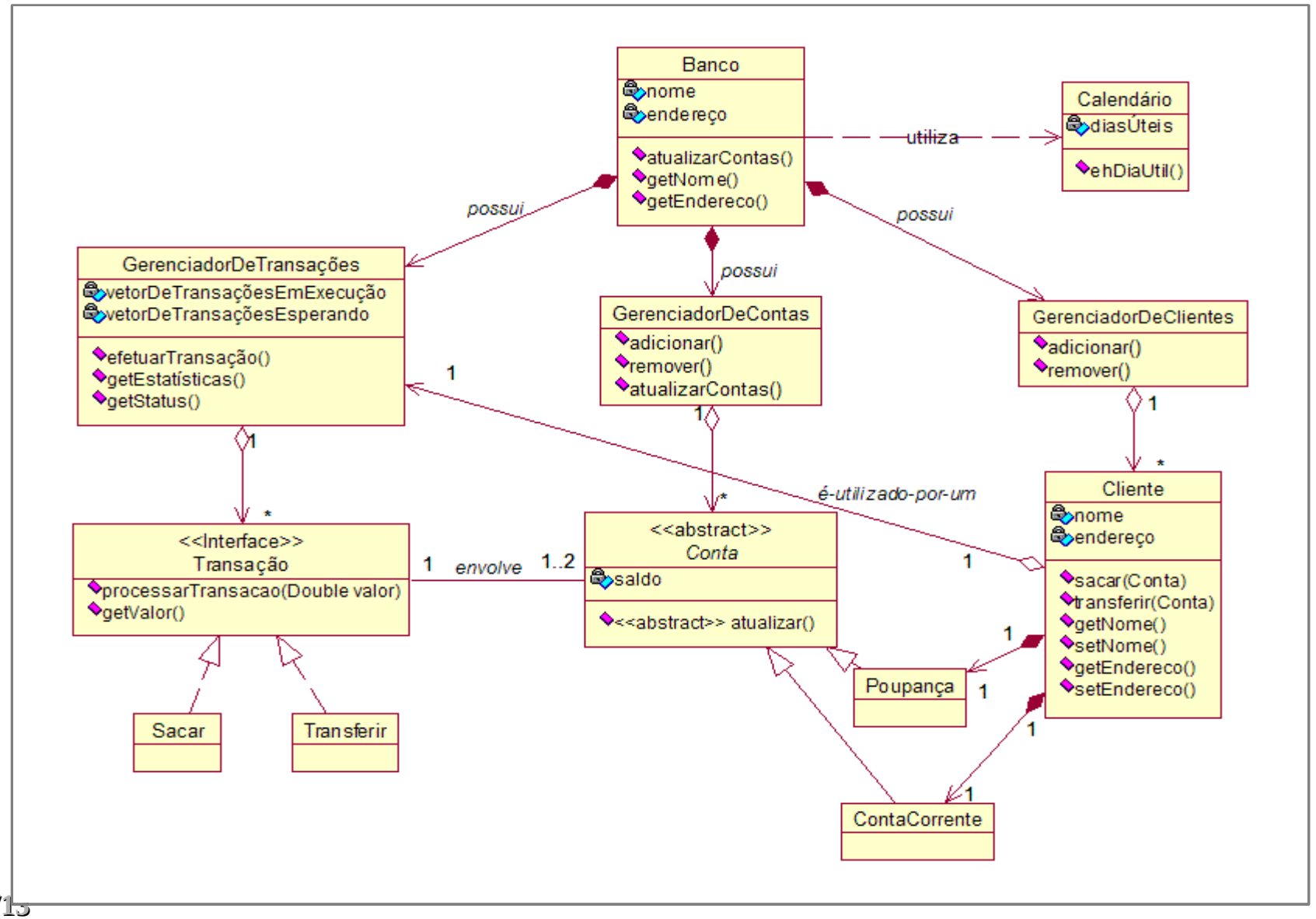


Diagrama de classes

- Um diagrama de classes contém:
  - Entidades
  - Relacionamentos

Diagrama de classes

- Entidades
  - Classe Concreta
    - Uma classe é representada na forma de um retângulo, contendo duas linhas que separam 3 partes. A primeira contém no nome da classe, a segunda os atributos da classe e a última os métodos da mesma.

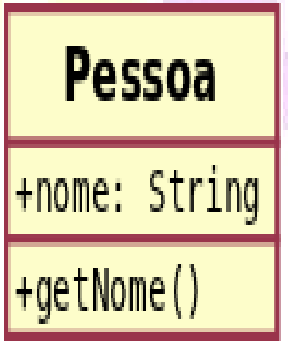


Diagrama de classes

- Classe Abstrata
  - A representação de uma classe abstrata em UML é quase igual à representação de uma classe concreta, a única diferença é o estilo da fonte do nome da classe, que, neste caso, está em itálico.

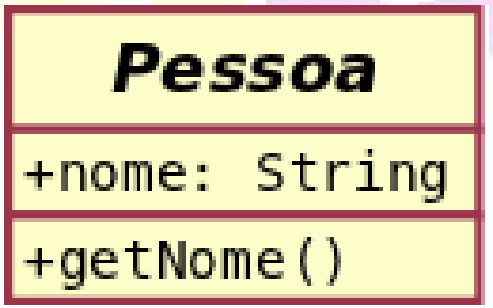


Diagrama de classes

- INTERFACE
  - Perspectivas:
    - Conceitual

      Apenas classes são utilizadas. Neste tipo de perspectiva, uma classe é interpretada como um conceito. Apenas atributos são utilizados.

    - Especificação

      Tanto classes como interfaces são utilizados neste tipo de perspectiva. O foco consiste em mostrar as principais interfaces e classes juntamente com seus métodos.

      Não é necessário mostrar todos os métodos, pois o objetivo deste diagrama nesta perspectiva é prover uma maior entendimento da arquitetura do software a nível de interfaces.

    - Implementação

      Nesta perspectiva, vários detalhes de implementação podem ser abordados, tais como:
      - visibilidade de atributos e métodos;
      - parâmetros de cada método, inclusive o tipo de cada um;
      - tipos dos atributos e dos valores de retorno de cada método.

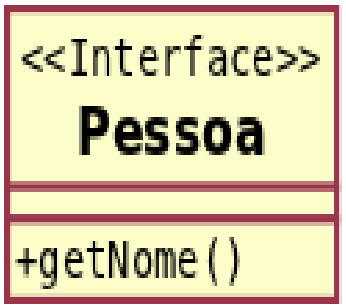


Diagrama de classes

- Relacionamentos
  - Papel

    Descreve o relacionamento.

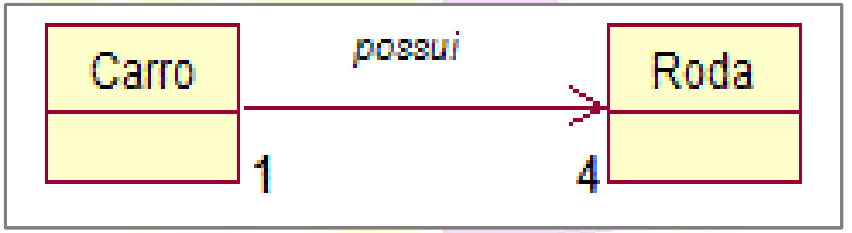


Diagrama de classes

- Multiplicidade (utilizado em todas as perspectivas de forma uniforme)

Notações possíveis

Tipos	Significado
0..1	Zero ou uma instância. A notação n..m indica n para m instâncias.
0..* ou *	Não existe limite para o número de instâncias
1	Exatamente uma instância.
1..*	Ao menos uma instância.

Exemplo:

Diagrama de classes

- Associação (utilizado em todas as perspectivas)

- Perspectiva:
  - Conceitual
    - Define um relacionamento entre duas entidades conceituais do sistema.
  - Especificação
    - Define responsabilidades entre duas classes. Implica que existem métodos que tratam desta responsabilidade.
  - Implementação
    - Permite saber quem está apontando para quem, através da representação gráfica da navegabilidade. Além disto, é possível compreender melhor de que lado está a responsabilidade.

Diagrama de classes

```
public class Empresa {
    private Trabalhador trabalhador;
    public Empresa( ){
    }
    public void setTrabalhador(Trabalhador trabalhador){
         this.trabalhador = trabalhador;
    }
    public Empregado getTrabalhador( ) {
        Return trabalhador;
    }
}
```

```
public class Trabalhador {

    public Trabalhador( ){

    }
}
```

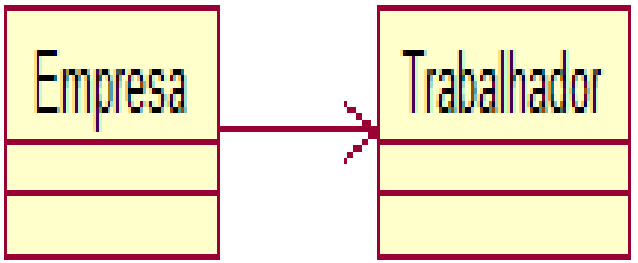


Diagrama de classes

- Herança ou Generalização (utilizado em todas as perspectivas)

  Representação Gráfica

  Perspectiva:

  - Seja B uma generalização (extensão) de A.
- Conceitual

  Considera que B é um subtipo ou um tipo especial de A. O que é válido para A, também é válido para B.
- Especificação

  Ocorre uma herança de interface.
- Implementação

  Ocorre uma herança de implementação.

Diagrama de classes

Exemplo de uma herança de implementação:

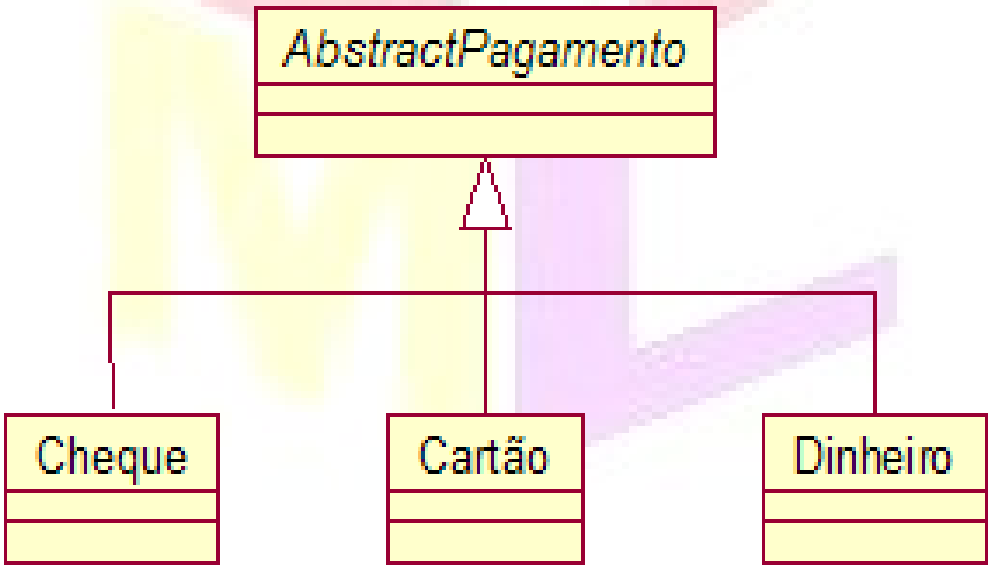
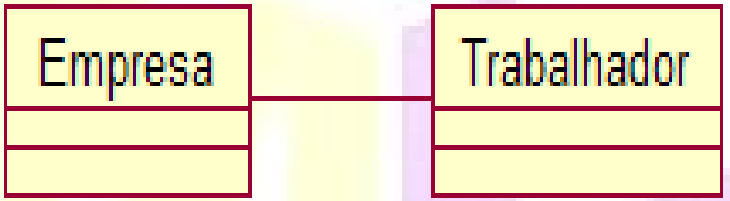


Diagrama de classes

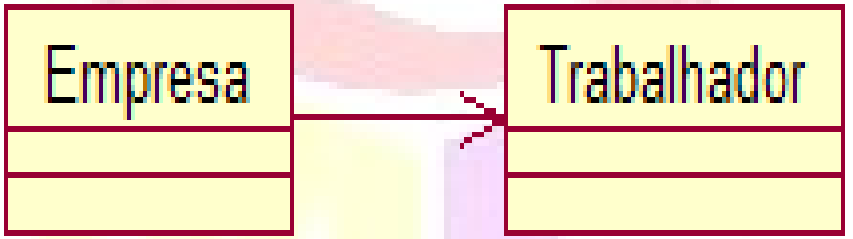
- Navegabilidade (utilizado apenas na perspectiva de implementação)
- Um relacionamento sem navegabilidade implica que ele pode ser lido de duas formas, isto é, em suas duas direções. Ex.:



- Uma empresa possui um trabalhador, como também um trabalhador trabalha em uma empresa.
- Utilizando a propriedade de navegabilidade, podemos restringir a forma de ler um relacionamento. Isto é, em vez de termos duas direções, teremos apenas uma direção (de acordo com a direção da navegação). Ex.:

Diagrama de classes

- Utilizando a propriedade de navegabilidade, podemos restringir a forma de ler um relacionamento. Isto é, em vez de termos duas direções, teremos apenas uma direção (de acordo com a direção da navegação). Ex.:



Uma empresa possui um trabalhador.

Diagrama de classes

- Agregação (utilizado apenas na perspectiva de implementação)
  - Definição
    - Agregação é uma associação em que um objeto é parte de outro, de tal forma que a parte pode existir sem o todo.
    - Em mais baixo nível, uma agregação consiste de um objeto contendo referências para outros objetos, de tal forma que o primeiro seja o todo, e que os objetos referenciados sejam as partes do todo.
    - A diferença entre os relacionamentos de associação e agregação ainda é algo de bastante discussão entre os gurus. De forma geral, utiliza-se agregação para enfatizar detalhes de uma futura implementação (perspectiva de implementação).

Diagrama de classes

Agregação com navegabilidade

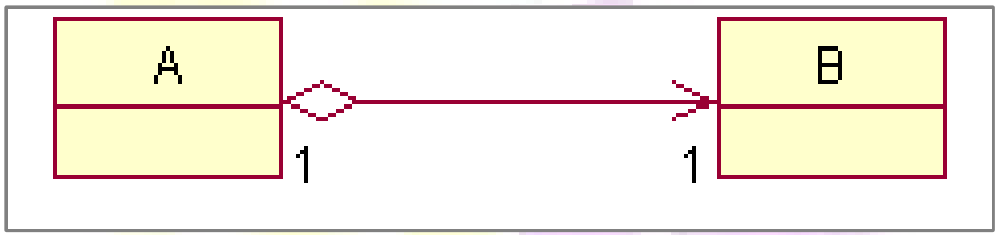


Diagrama de classes

- Composição (utilizado apenas na perspectiva de implementação)
  - Definição
    - Em mais baixo nível, em termos de passagem por parâmetro, seria uma passagem por valor. Enquanto que agregação seria uma passagem por referência.
    - O todo contém as partes (e não referências para as partes). Quando o todo desaparece, todas as partes também desaparecem.

Diagrama de classes

Representação Gráfica

Diagrama de classes

- Implementação (utilizado apenas na perspectiva de implementação)
- Em Inglês: realization
- Definição
  - Utilizado para indicar que uma classe implementa uma interface

Implementação de uma interface representada por um círculo

Implementação de uma interface representada por um retângulo

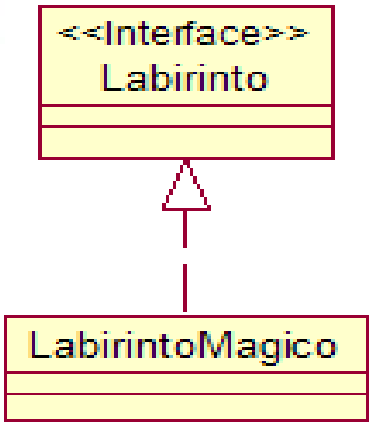
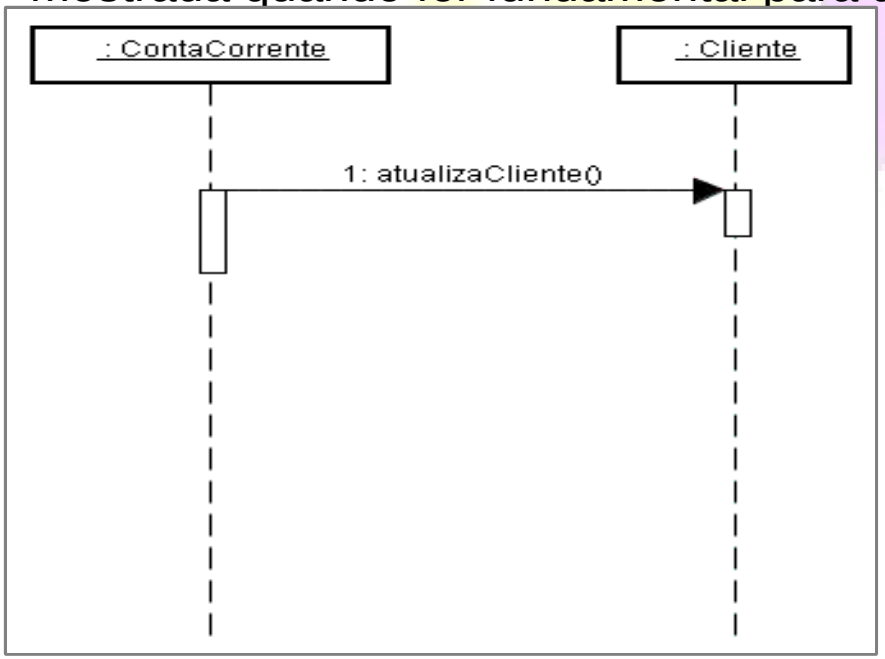


Diagrama de Sequência

- Consiste em um diagrama que tem o objetivo de mostrar como as mensagens entre os objetos são trocadas no decorrer do tempo para a realização de uma operação.
- Em um diagrama de sequência, os seguintes elementos podem ser encontrados:
  - Linhas verticais representando o tempo de vida de um objeto (lifeline);
  - Estas linhas verticais são preenchidas por barras verticais que indicam exatamente quando um objeto passou a existir. Quando um objeto desaparece, existe um "X" na parte inferior da barra;

Diagrama de sequência

- Linhas horizontais ou diagonais representando mensagens trocadas entre objetos. Estas linhas são acompanhadas de um rótulo que contém o nome da mensagem e, opcionalmente, os parâmetros da mesma. Observe que também podem existir mensagens enviadas para o mesmo objeto, representando uma iteração;
- Mensagens de retorno são representadas por linhas horizontais tracejadas. Este tipo de mensagem não é frequentemente representada nos diagramas, muitas vezes porque sua utilização leva a um grande número de setas no diagrama, atrapalhando o entendimento do mesmo. Este tipo de mensagem só deve ser mostrada quando for fundamental para a clareza do diagrama.



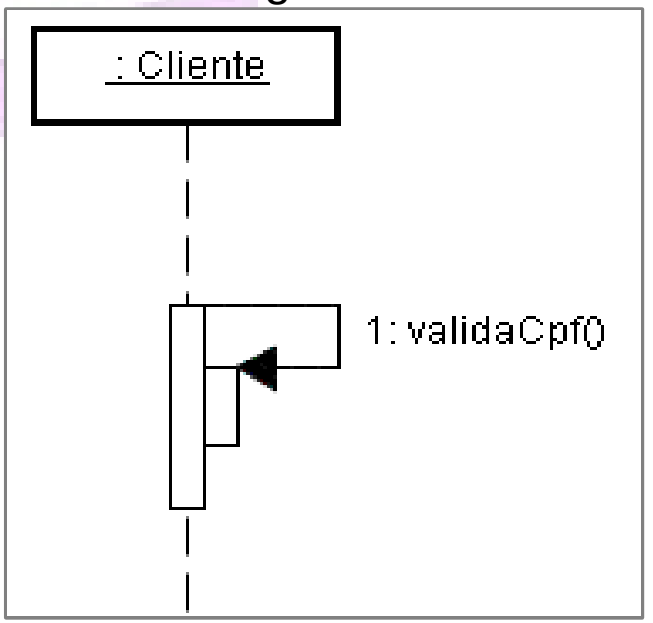
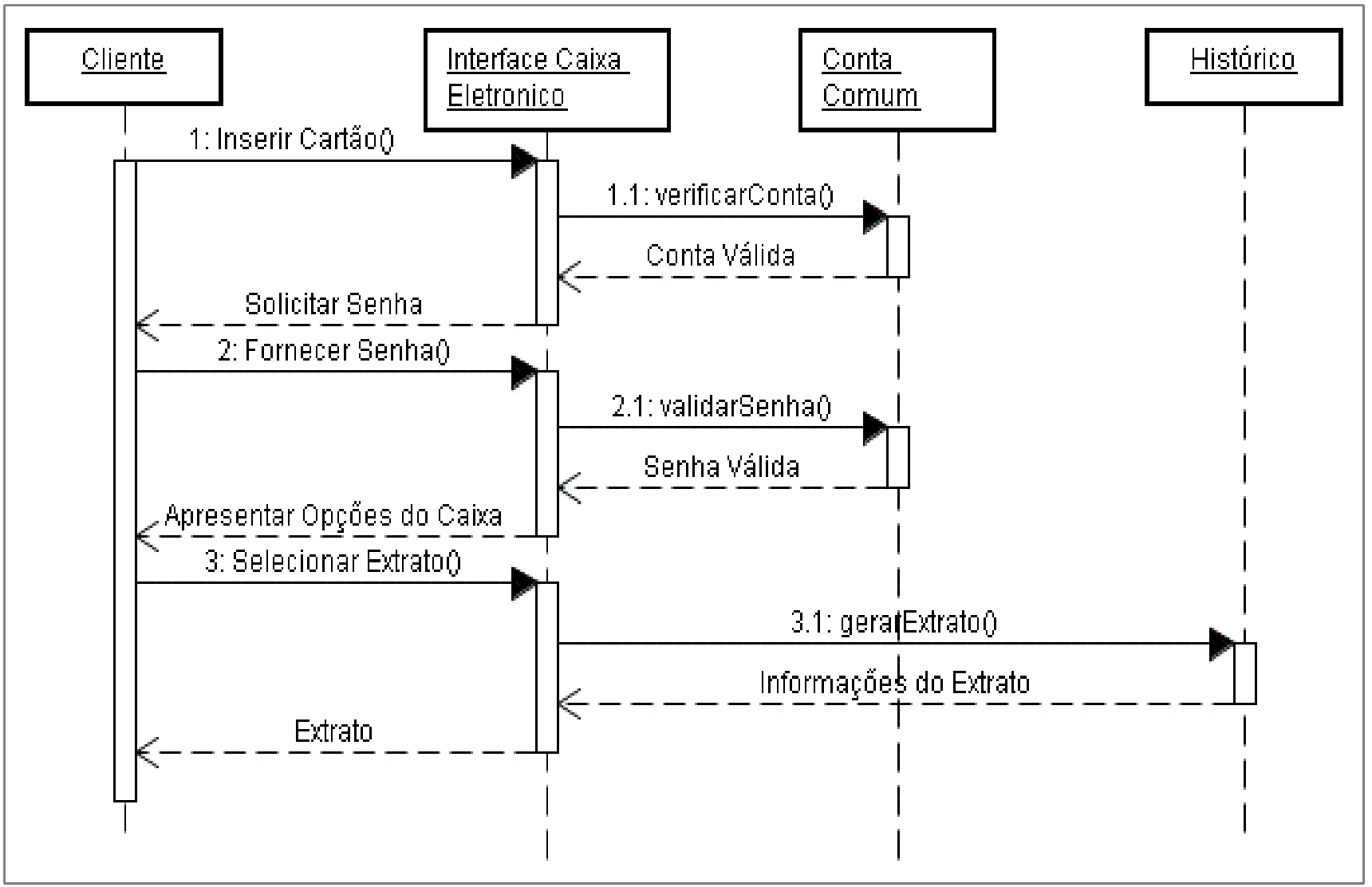


Diagrama de sequência



Dúvidas