Análise e Projeto de Sistemas

1 - Prototipagem

2 - Modelagem de Classes de Análise

Prototipagem

- A ***prototipagem*** é uma técnica aplicada quando:
  - há dificuldades no entendimento dos requisitos do sistema
  - há requisitos que precisam ser mais bem entendidos.
- A construção de ***protótipos*** utiliza ambientes com facilidades para a construção da interface gráfica.
- Procedimento geral da prototipagem:
  - Após o LR*, um protótipo é construído para ser usado na *validação*.
  - Usuários fazem críticas...
  - O protótipo é então corrigido ou refinado
  - O processo de revisão e refinamento continua até que o protótipo seja aceito.
  - Após a aceitação, o protótipo é descartado ou utilizado como uma versão inicial do sistema.

*LR - Levantamento de Requisitos

Prototipagem

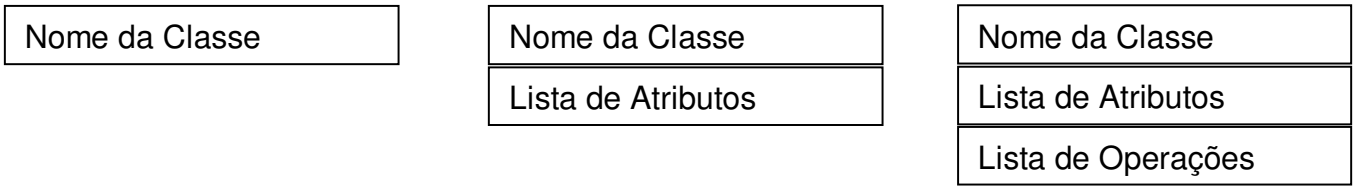
- Note que a prototipagem NÃO é um substituto à construção de modelos do sistema.
  - A prototipagem é uma técnica *complementar* à construção dos modelos do sistema.
  - Mesmo com o uso de protótipos, os modelos do sistema devem ser construídos.
  - Os erros detectados na validação do protótipo devem ser utilizados para modificar e refinar os modelos do sistema.

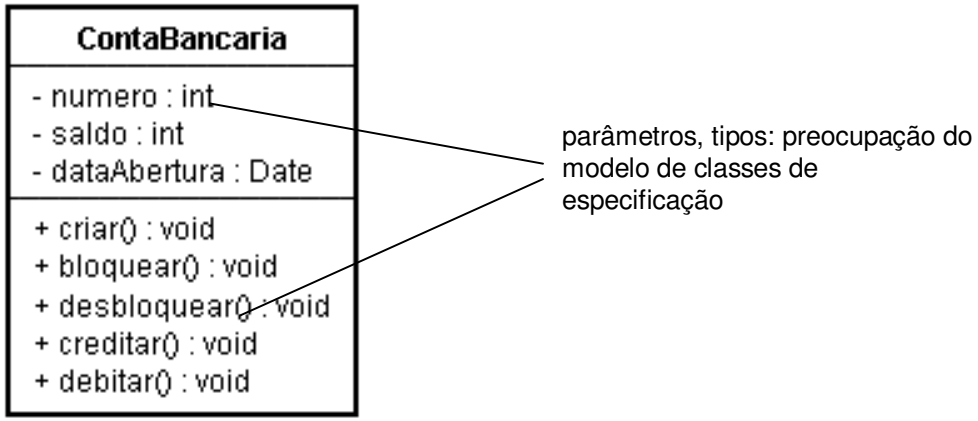
Modelagem de Classes de Análise

- **Visão externa: visão de casos de uso**
- **Visão interna: colaboração entre os objetos**
  - **aspecto dinâmico (iterações / estados) – cap. 7 e 10**
  - **aspecto estrutural estático**
    - **representa a estrutura das classes de objetos e as relações entre elas).**
    - **Não representam informações sobre como os objetos se interagem no decorrer do tempo.**
- **Modelo de Classes**
  - **Diagrama de classes + Descrição textual associada;**
  - **Geralmente é incrementado durante as iterações;**
  - **Níveis de Abstração: Domínio → Especificação → Implementação**
    - **Domínio → Fase de Análise | sem restrições de tecnologia | representa domínio do negócio.**
    - **Especificação → Fase de Projeto | extensão do mod.domínio: add detalhes específicos da solução de SW escolhida. Add novas classes | Nivel alto de abstração.**
    - **Implementação → Fase de Implementação | extensão do mod. espec | Implementa as classes em alguma linguagem OO | Nível baixo de abstração.**
- **Nomenclatura:**
  - **usar camelWords... Classes. atributosDeClasses | operacoesDeClasses.**
  - **Pode-se manter siglas inalteradas. Ex: atributo CPF.**

Modelagem de Classes de Análise

- **Diagrama de Classes:**

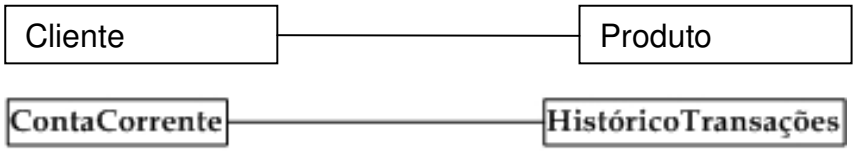




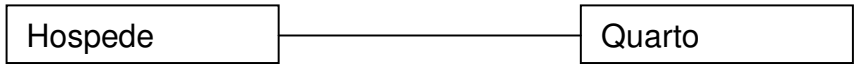
- **Relacionamentos entre objetos (associação)**
  - **representados no modelo de classes, mas referem-se às relações de um objeto (instância) com outro, durante a execução do sistema.**

Modelagem de Classes de Análise

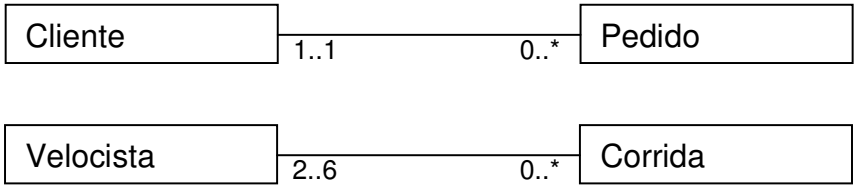
- **Relacionamentos entre objetos (associação)**
  - **Ex:**
    - **Domínio de Vendas: um *cliente* compra *produtos***

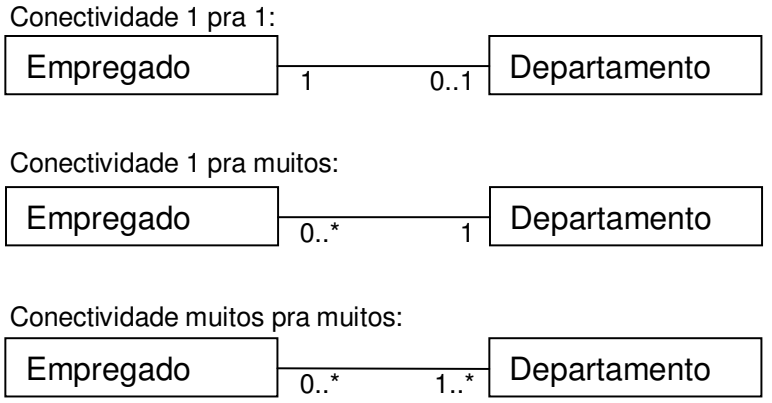


    - **Domínio Hotelaria: em um hotel, há vários *hóspedes*, que ocupam *quartos***



  - **Multiplicidades:**
    - **representam limite superior e inferior da quantidade de objetos aos quais um outro objeto pode estar associado.**
      - **1** **Apenas 1 (idem 1..1)**
      - **0..*** **Zero a muitos (idem *)**
      - **1..*** **Um a muitos**
      - **0..1** **Zero ou um**
      - **x..y** **Intervalo Específico**
      - **intervalos em conjunto são admitidos (ex: 1, 3, 5..9, 11)**





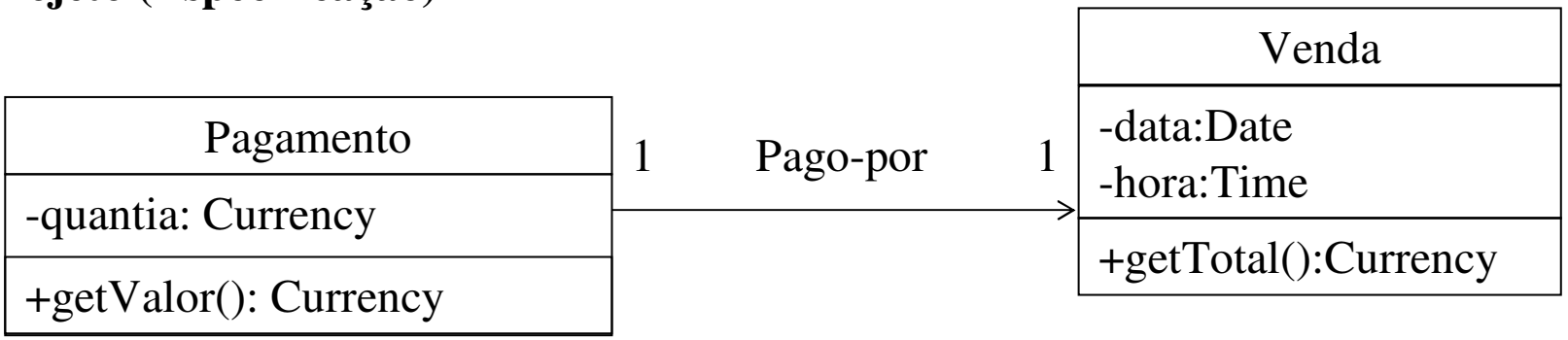
  - **Nome de associação, direção de leitura e papéis** (usar com bom-senso)

Modelagem de Classes de Análise

- **Relacionamentos entre objetos (associação)**

Análise

Projeto (Especificação)



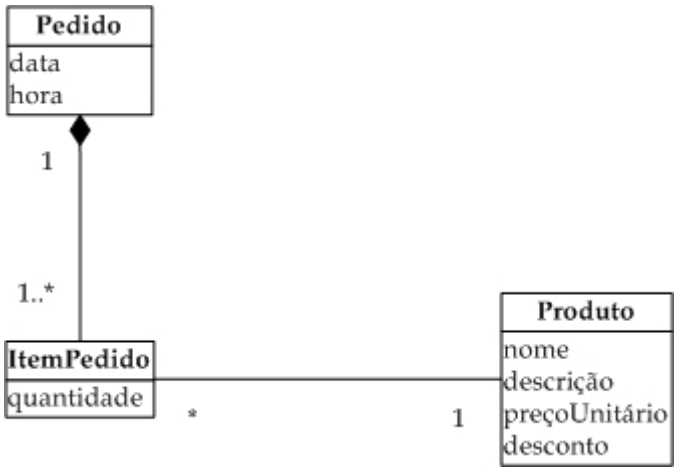
Modelagem de Classes de Análise

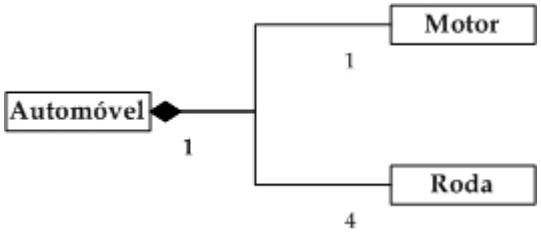
- **Agregações e composições**
  - **Casos especiais de associações (podem ser substituídas por associações – a diferença é só semântica).**
  - **Representam conexões entre objetos que guardam relação todo-parte entre si**
  - **Um objeto está contido no outro**
  - **Perguntas:**
    - **X *tem um ou mais* Y?**
    - **Y *é parte de* X ➔**
    - **X é todo e Y é parte**
  - **Propagam comportamento: Um comportamento aplicado ao *Todo* é aplicado às *Partes***
  - **Não são simétricas: Se A é parte de B, B não pode ser parte de A.**

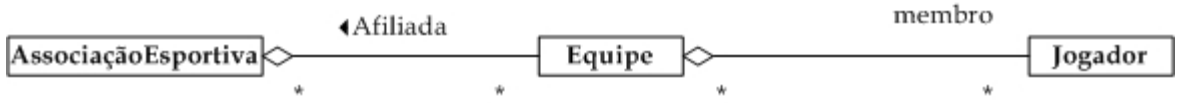
  - **Composição:**
    - **Agregação não compartilhada (objetos parte pertencem a um único todo)**
    - **Destruir um objeto todo implica destruição de todos os objetos parte.**

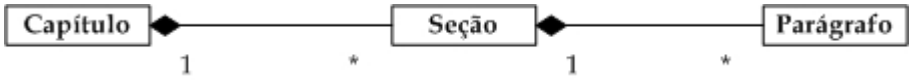
Modelagem de Classes de Análise

- **Agregações e composições**
  - **EXEMPLOS**



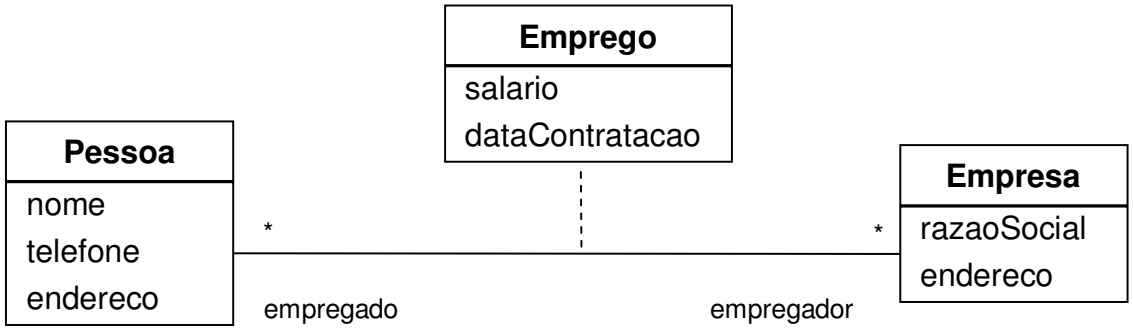






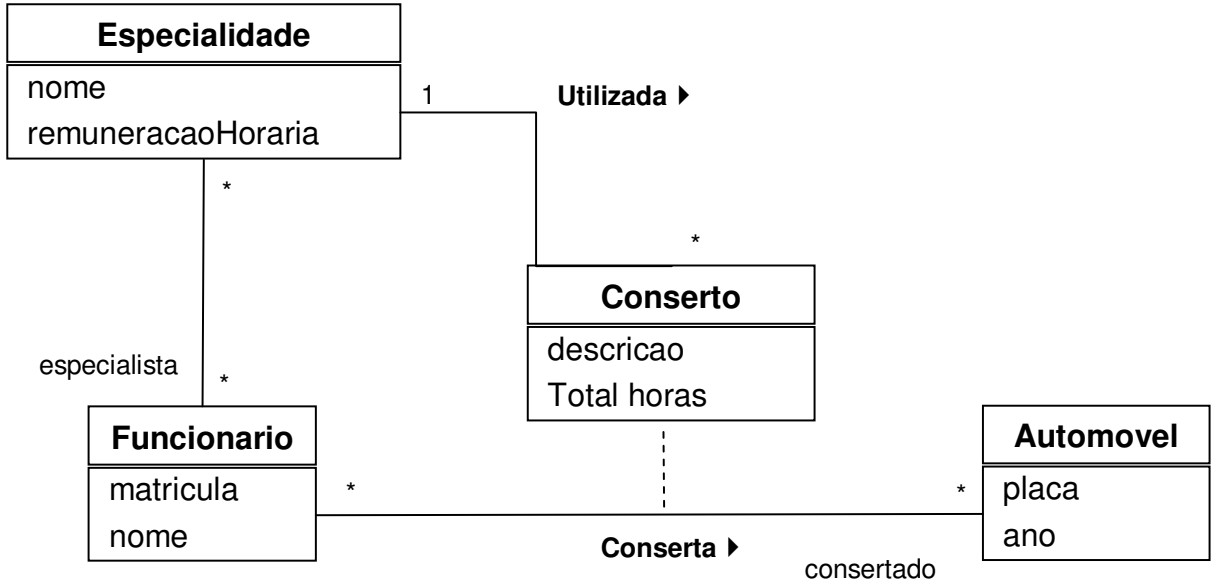
Modelagem de Classes de Análise

- **Classes Associativas**
  - **Ligadas a associações, e não em outras classes**
  - **Mais frequentes em conectividades muitos-para-muitos**
  - **Usadas quando há necessidade de manter informações sobre a associação**

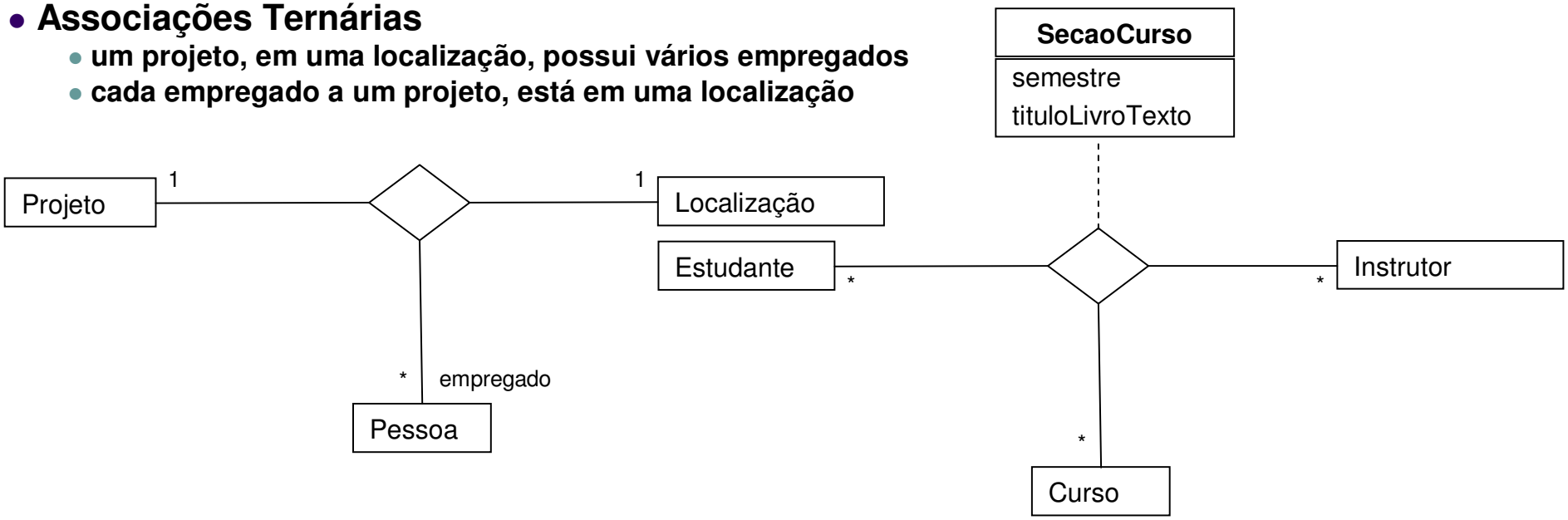


Modelagem de Classes de Análise

- **Classes Associativas**
  - **é um elemento híbrido:**
    - **classe + associação**

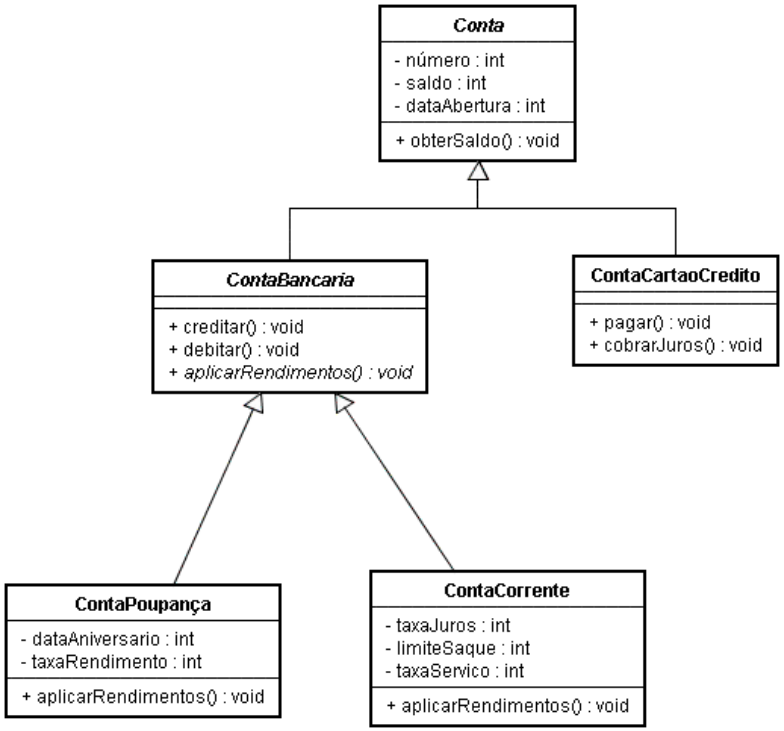


- **Associações Ternárias**
  - **um projeto, em uma localização, possui vários empregados**
  - **cada empregado a um projeto, está em uma localização**



Modelagem de Classes de Análise

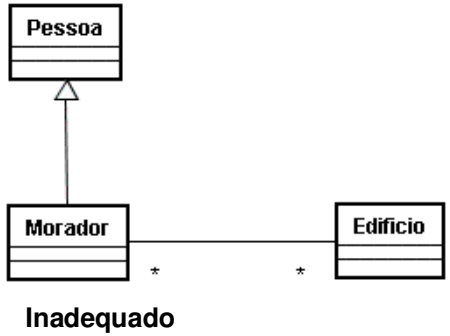
- **Generalizações e Especializações**
  - **O modelo de classes pode representar relacionamentos entre classes, não só entre objetos.**
  - **Mamífero (mais genérico) → Ser Humano (menos genérico)**
  - **Superclasse possui propriedades herdadas por subclasses**



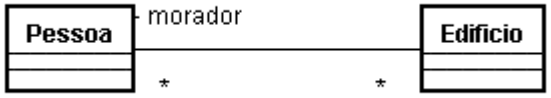
- **Propriedades:**
  - **Transitividade**
    - **Uma classe herda tanto relacionamentos de sua superclasse imediata quanto de superclasses não imediatas (classes em nível mais alto da hierarquia).**
    - **Uma instancia de classe C também é instância de todos os ancestrais de C**
    - **ContaPoupança possui 2 atributos próprios e mais 3 herdados de Conta**
  - **Assimetria**
    - **Se A é uma generalização de B, então B não pode ser generalização de A. Não pode haver ciclos.**

Modelagem de Classes de Análise

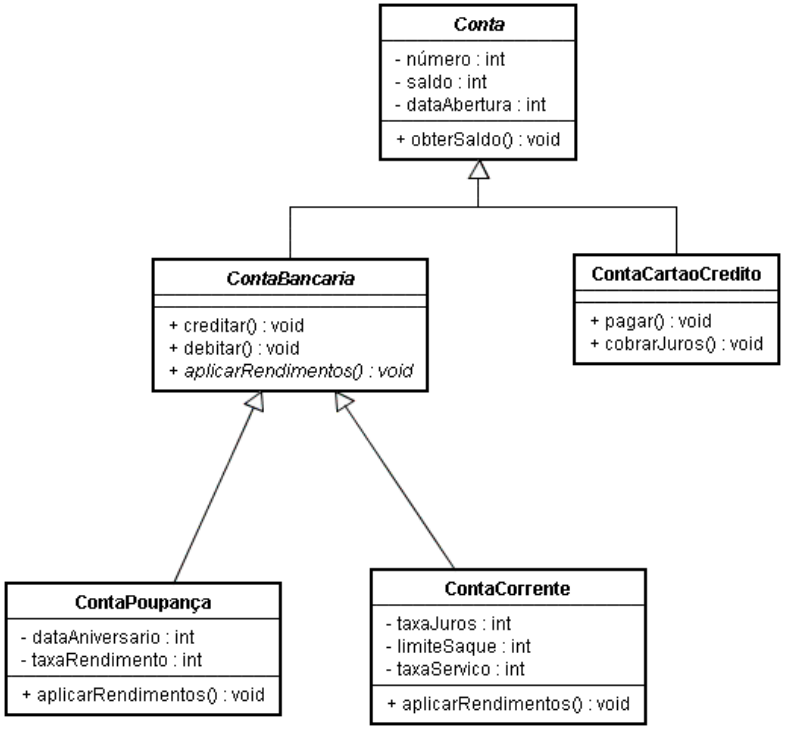
- **Refinando o modelo de classes com gen/espec**
  - **É inadequado o uso de gen/espec onde nem todas as propriedades e comportamentos da superclasse fazem sentido para a subclasse!**
  - **Deve-se evitar diagramas com hierarquias de herança com mais de 3 níveis;**
  - **Papéis e subclasses não devem ser confundidos! O modelador deve evitar a criação de subclasses em situações que podem ser resolvidas através da utilização de papéis.**



Inadequado

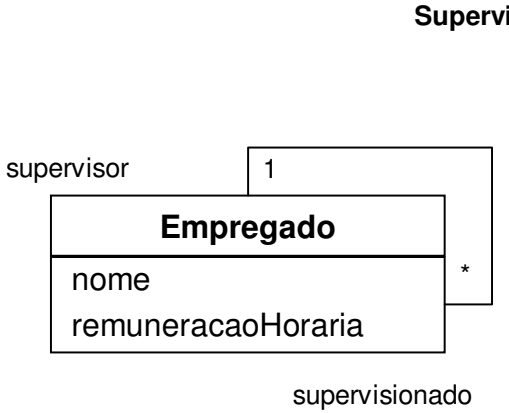


Adequado



Modelagem de Classes de Análise

- **Auto-Associação**
  - **Associação reflexiva**
  - **objetos da classe empregado associam-se a outros objetos da mesma classe**



- **IDENTIFICAÇÃO DAS CLASSES**
  - **Método dirigido a dados: busca conceitos → modelo conceitual**
  - **Método dirigido a responsabilidades: busca comportamentos**
    - **Um objeto deve cumprir obrigações... Algumas sozinho, outras com colaboração de outros.**

Modelagem de Classes de Análise

Categorias de responsabilidades

- Categorias de objetos de acordo com o tipo de responsabilidades (*fazer + conhecer*) a eles atribuídos
- Objetos de entidade, objetos de controle e objetos de fronteira

- **Objetos de Entidade**
  - É um repositório para alguma informação manipulada pelo sistema
  - Representam conceitos do domínio do negócio
  - Normalmente persistentes
  - Normalmente há várias instências co-existindo no sistema
  - Atores não tem acesso direto a estes objetos
  - Se comunicam com o exterior por outros objetos
  - Normalmente participam de vários casos de uso e têm ciclo de vida longo
  - Responsabilidades de *fazer* típicas:
    - Informar valores de seus atributos a objetos de controle
    - Realizar cálculos simples, normalmente com colaboração de objetos de entidade associados através de agregações
    - Criar e destruir *objetos-parte* (considerando que ele seja um *objeto-todo* de uma agregação).
  - Responsabilidades de *conhecer*:
    - normalmente fornecidas por atores quando da criação destes objetos
    - ex: nome, telefone, endereço do cliente
- **Objetos de Fronteira**
  - Fazem a comunicação direta do sistema com os atores
  - Podem realizar interface com o usuário (atores humanos), interfaces com outros sistemas e se comunicar com dispositivos atrelados ao sistema
  - Responsabilidades de *fazer*:
    - Notificar aos objetos de controle os eventos gerados externamente ao sistema.
    - Notificar aos atores o resultados de interações entre os objetos internos
  - Responsabilidades de *conhecer*:
    - para interação humana: informação manipulada através da interface c/ usuário
      - protótipos podem ajudar a identificar esses atributos
    - para comunic. com outros sistemas: propriedades de uma interface de comunicação
  - Durante identificação das classes, abstrair o comportamento dos objetos de fronteira.

Modelagem de Classes de Análise

- **Objetos de Controle**
  - **Ponte de comunicação entre Objs Fronteira e Objs Entidade**
  - **Controlam a lógica de execução correspondente a um caso de uso**
  - **Decidem o que o sistema deve fazer quando ocorre um evento externo relevante**
  - **Ativos e com vida curta: geralmente existem durante a realização de um caso de uso**
  - **Responsabilidades de *fazer*:**
    - **Realizar monitorações, a fim de responder a eventos externos (gerados por objetos de fronteira)**
    - **Coordenar a realização de um CDU* através do envio de mensagens a objetos de fronteira e de entidade**
    - **Assegurar que as regras de negócio estão sendo seguidas corretamente**
    - **Coordenar a criação de associações entre objetos de entidade**
  - **Resp. *conhecer*: de uso local**

- **UMA BOA MODELAGEM DE CLASSES DEVE SEPARAR A INFORMAÇÃO CONTIDA NO DOMÍNIO DO NEGÓCIO (OBJETOS DE ENTIDADE), AS VÁRIAS MANEIRAS DE VISUALIZAR UMA INFORMAÇÃO (OBJETOS DE FRONTEIRA) E A LÓGICA DA APLICAÇÃO (OBJETOS DE CONTROLE).**

*CDU - Casos de uso