
INSTITUTO
VIANNA JUNIOR

INSTITUTO VIANNA JÚNIOR

Análise OO

Partindo do Caso de Uso e chegando ao Diagrama de Classes

Professor: Camillo Falcão

Passo a passo

- 1) À partir dos casos de uso, identifique os conceitos mais importantes do sistema. Esses conceitos são conhecidos como abstrações chaves.

- 2) Crie diagramas de sequência com base em responsabilidade.

- 3) Preencha e refine as classes.

Diagrama de Sequência baseado em responsabilidades

- Uma leitura de cada cenário do caso de uso deve ser feita e, durante essa leitura, devemos identificar as classes de fronteira, controle e entidades envolvidas.
- A identificação das abstrações chaves (feita anteriormente) auxilia a identificação das classes de entidades.
  - As abstrações chaves e as classes de entidades estão relacionadas às regras de negócio.
  - As abstrações chave são transformadas em classes de entidade e isso pode ajudar o processo de entendimento das regras de negócio.

Diagrama de Sequência baseado em responsabilidades

- Existe um motivo para iniciarmos criando um diagrama de sequência ao invés de iniciarmos com um diagrama de classes:
  - Enquanto estamos visualizando as interações entre ator e sistema, é possível transformar essas interações em mensagens trocadas por classes do diagrama de sequência.
  - Dessa forma, ao interpretarmos o passo a passo do caso de uso como mensagens trocadas entre classes, as classes do diagrama de sequência vão aparecendo gradualmente durante a leitura do caso de uso.

Preenchendo e refinando as classes

- Para iniciar o seu diagrama de classes, basta que cada classe diferente criada em seus diagramas de sequência seja criada em seu diagrama de classes.
  - Cuidado: ao tratar cada cenário isolado, é possível que classes com responsabilidades muito parecidas tenham sido criadas em diagramas de sequências diferentes. Dessa forma, você deve tratar esses problemas e criar uma classe única para essas classes com mesma responsabilidade.

Preenchendo e refinando as classes

- Analise as mensagens trocadas entre as classes dos diagramas de sequência. Essas mensagens representam o relacionamento entre as classes.
- Você deve refinar esses relacionamentos identificando qual o seu tipo. Você deve ter um bom entendimento do caso de uso para identificar esse tipo de relacionamento corretamente. Exemplo: como saber se determinado relacionamento é uma agregação ou uma composição sem conhecer em detalhes o caso de uso?

Preenchendo e refinando as classes

- Quando uma mensagem é criada somente para trafegar informações entre classes, transforme essa mensagem em um atributo da classe responsável por fornecê-las.
  - Existem exceções: se uma classe A necessita de acessar uma classe B para fornecer uma informação, então provavelmente a responsabilidade de guardar essa informação em forma de atributo é da classe B.

GRASP: o padrão Controller

- O padrão *Controller*, ou Controlador, define a classe responsável por tratar um acontecimento externo ao sistema e que desencadeia alguma ação do mesmo. Este é o padrão responsável por tratar eventos do sistema.
- Como atribuir essa responsabilidade?
  - Criando um controlador de caso de uso.
    - Uma classe que trata todos os eventos de um caso de uso.
  - Criando um controlador fachada.
    - Uma classe que controla todos os eventos do sistema.

GRASP: o padrão Controller

- O estereótipo “control” que utilizamos em alguns de nossos diagramas é utilizado para representar as classes *controller*.
- As classes de fronteira (ou *boundary*) não devem ter a responsabilidade de manipular eventos do sistema, elas devem apenas delegar essa tarefa para o controlador.
  - Isso aumenta a nossa capacidade de reuso, pois estamos criando interfaces “plugáveis” para o sistema.
    - Se a camada do sistema que faz interface com o usuário (classes de fronteira) não se responsabiliza por resolver os eventos gerados, então ela pode ser trocada facilmente.
      - Isso é possível devido a essa classe ter um baixo nível de acoplamento!

Exemplo

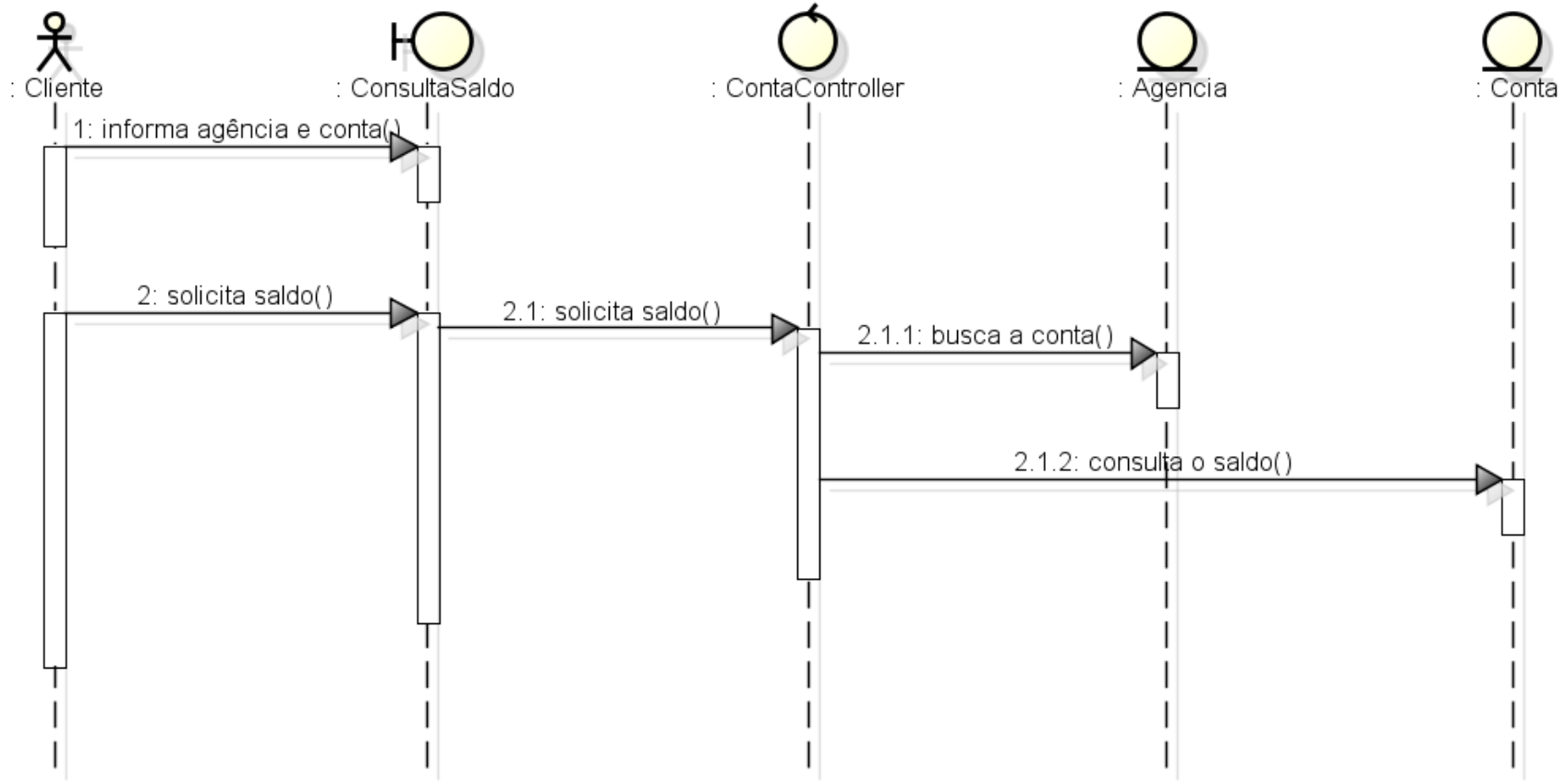
- Caso de uso Consultar Saldo:
  - Fluxo Base:
    1. O cliente informa a agência e a conta;
    2. O cliente solicita o saldo;
    3. O sistema solicita a senha;
    4. O cliente informa a senha;
    5. O sistema informa o nome do cliente, a agencia, a conta e o saldo atual.

Exemplo

- Ao analisar o caso de uso, é possível identificar algumas abstrações chave:
  - Cliente: pessoa ou empresa que utiliza os serviços do banco;
  - Conta: representação dos valores monetários do cliente no banco. Pode ser do tipo corrente ou poupança;
  - Agência: representação de um conjunto de contas.

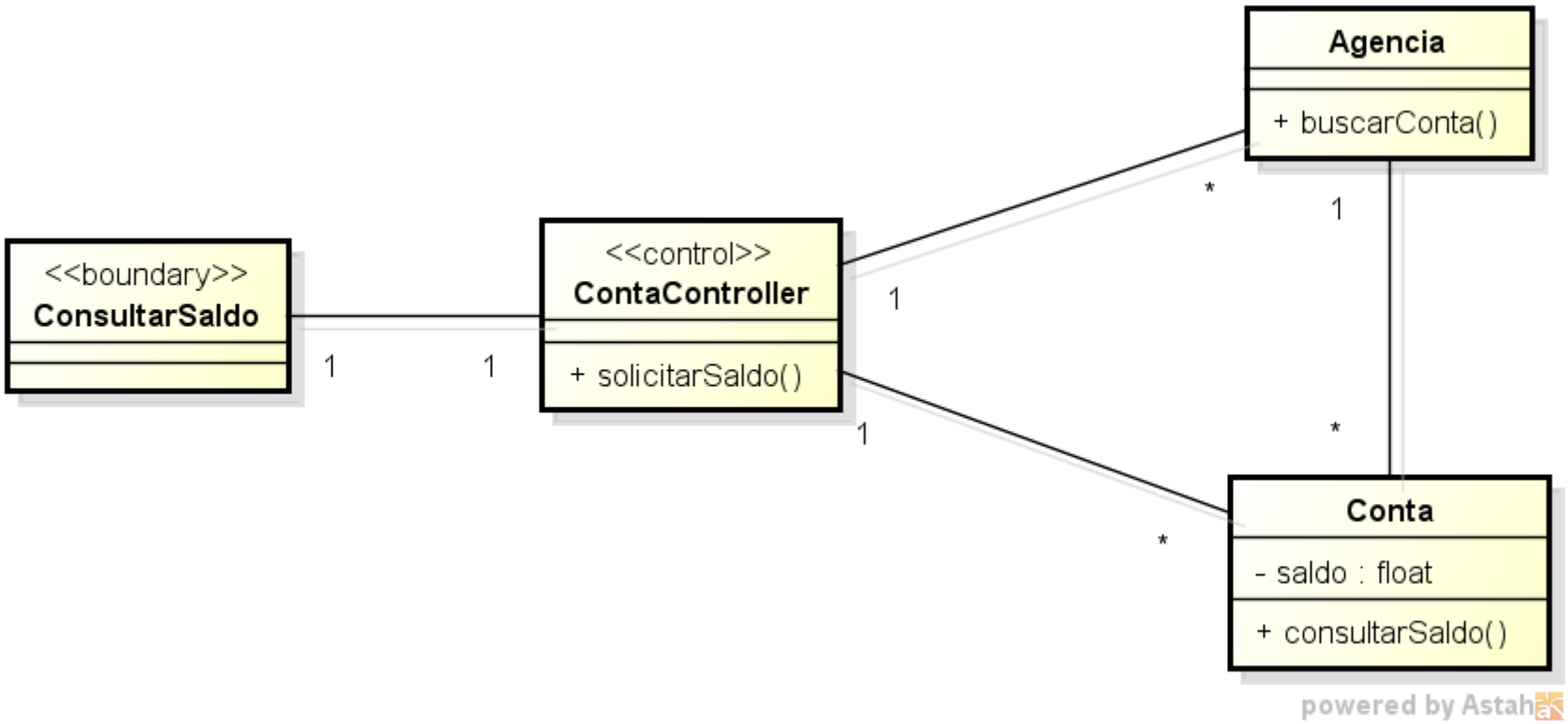
Exemplo

- Diagrama de Sequência extraído do caso de uso:



Exemplo

- Diagrama de Classes extraído do caso de uso:



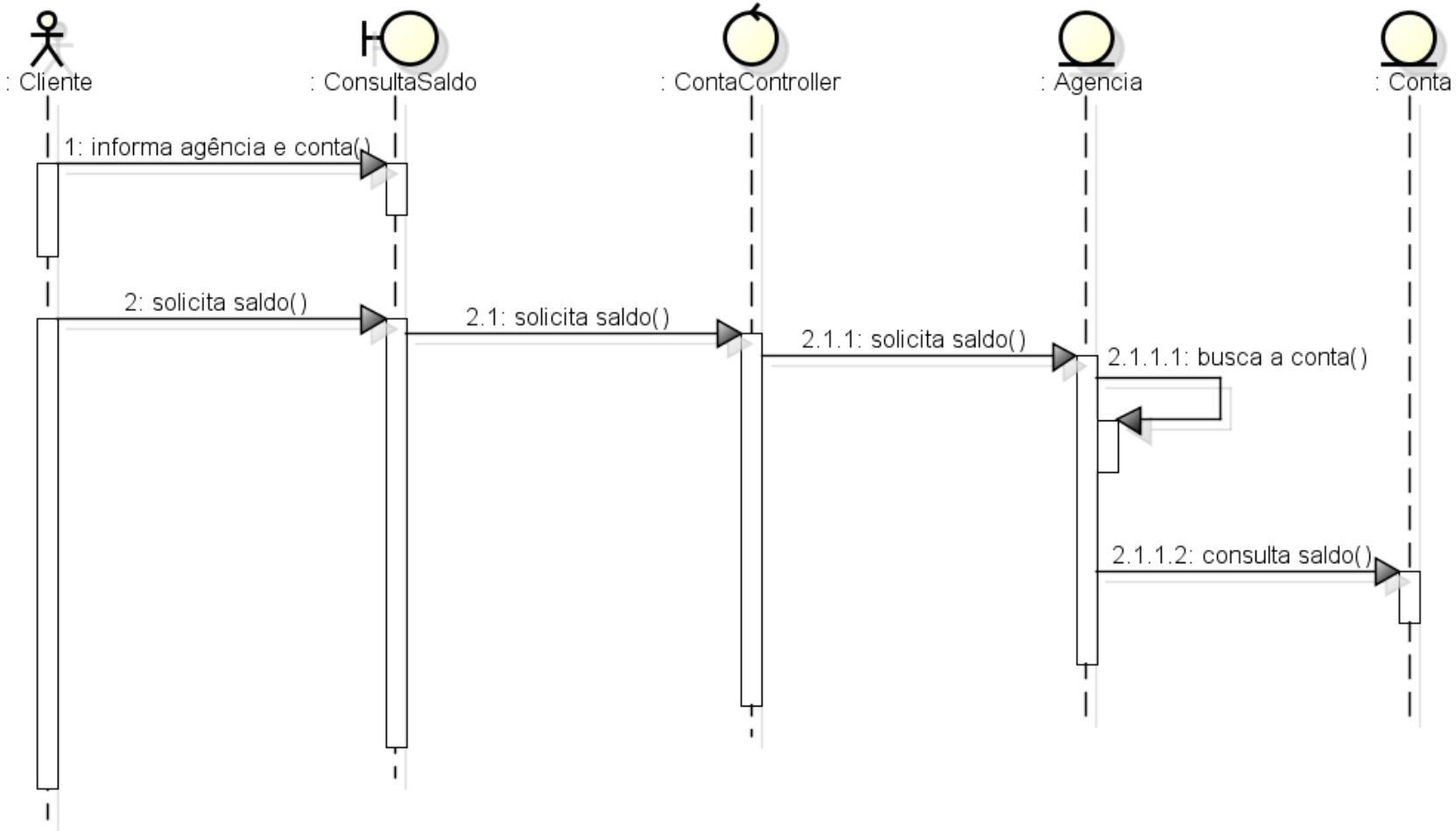
Exemplo

- Utilizando GRASP, é possível chegar a um refinamento nas responsabilidades das classes e também a uma identificação mais precisa dos atributos e operações envolvidos.

- Esse refinamento pode nos levar à descoberta de novas classes, atributos ou operações.

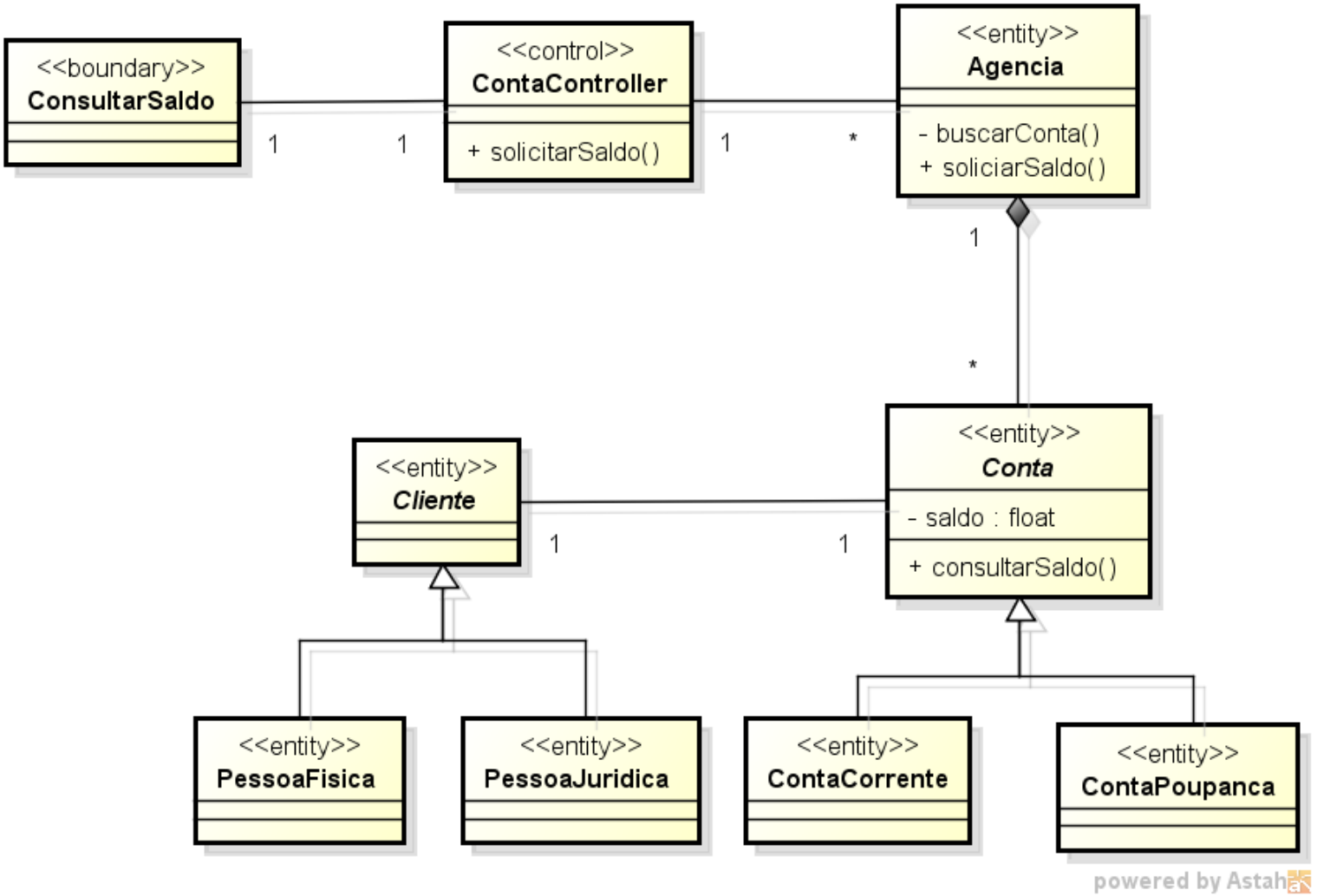
Exemplo

- Diagrama de Sequência após refinamento:



Exemplo

- Diagrama de Classes após refinamento:



Exemplo

- Explicações sobre os refinamentos utilizados:

  1) A classe ContaControl tinha dependência direta para as classes Agencia e Conta. Ela se responsabilizava por buscar a conta pedindo-a para a classe Agencia e por buscar diretamente o saldo contido na classe Conta.

  2) Pelo princípio do padrão Especialista, quem deveria se responsabilizar pelo acesso às informações da classe Conta é a classe Agencia. Afinal, a agência é dona de suas contas e essas contas não existem sem sua agência. Sendo assim, criamos uma composição entre as classes Agencia e Conta e também eliminamos o acesso direto da classe ContaControl à classe Conta.

Exemplo

- Explicações sobre os refinamentos utilizados:

3) Após essas modificações, diminuímos o acoplamento entre as classes, visto que ContaControl não está mais acoplada diretamente à Conta. Então usamos também o padrão Baixo Acoplamento.

Exemplo

- Explicações sobre os refinamentos utilizados:

  4) Analisando o caso de uso e as abstrações chave, percebemos algumas coisas, como:

  - A conta pode ser do tipo corrente ou poupança;
  - A definição da abstração chave Cliente mostra o cliente como uma pessoa ou uma empresa.

  Comparando essas informações com o diagrama de classe, vemos que precisaremos melhorar duas coisas: a coesão da classe Conta e tratar o conceito de negócio cliente.

Exemplo

- Explicações sobre os refinamentos utilizados:

  5) A classe Conta está se responsabilizando tanto pelas particularidades da conta corrente quanto pelas da conta poupança. Para melhorar sua coesão e assim respeitar o padrão Coesão Alta, fizemos o seguinte:

  - Tornamos a classe Conta abstrata, responsabilizando-a apenas pelo comportamento e informações que forem comuns tanto a conta corrente quanto a conta poupança;
  - Criamos as classes concretas ContaCorrente e ContaPoupança, filhas de Conta, para tratar de questões específicas sobre conta corrente e conta poupança.