Fundação CECIERJ - **Vice Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
Disciplina de Arquitetura e Projeto de Sistemas II
Gabarito – AD1 1° semestre de 2011.

Nome –

Observações:

1. Prova com consulta.

Atenção: Como a avaliação à distância é individual, caso sejam constatadas semelhanças entre provas de alunos distintos, será atribuída a nota ZERO a TODAS as provas envolvidas. As soluções para as questões podem ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser distinta. ALÉM DISSO, às questões desta AD respondidas de maneira muito semelhantes às respostas oriundas dos gabaritos já publicados de ADs de períodos anteriores, será atribuída a nota ZERO, incluindo também cópias diretas e sem sentido de tópicos dos slides das aulas.

Questão 1 [9 pontos]

Considere a situação em que você tenha que gerar documentação para viabilizar a construção de um Sistema de Manutenção e Controle de Contas Correntes e Aplicações Financeiras. Para isso, tome por base as seguintes informações extraídas de dados fornecidos pelo cliente durante uma entrevista, de como a gestão deve ser realizada, para entendimento do problema e dos requisitos do sistema a ser desenvolvido:

- *O sistema suportará um cadastro de clientes, onde cada cliente cadastrado poderá ter várias contas correntes, vários dependentes ligados a ele, e várias contas de poupança;*
- *Cada dependente poderá possuir várias contas de poupança, mas não poderão ter uma conta corrente própria;*
- *Poupança seria uma conta que possui um valor, um prazo de aplicação a uma taxa de juros (definida no vencimento da poupança);*
- *Aplicação Pré-fixada seria uma aplicação de um valor, em um prazo pré-determinado a uma taxa de juros previamente definida;*
- *Tanto a conta corrente quanto a poupança deverão manter um histórico de todas as movimentações de crédito, débito, transferências e aplicações de pré-fixados (pré-fixados apenas para conta corrente);*
- *A conta corrente poderá ter várias aplicações pré-fixadas ligadas a ela.*

Dessa forma, depois da análise das informações retiradas da entrevista, construa os artefatos a seguir, considerando a Análise e o Projeto Orientados a Objetos:

a) [1 ponto] Realize uma descrição do Sistema de Manutenção e Controle de Contas Correntes e Aplicações Financeiras a ser desenvolvido;
b) [1 ponto] Especifique uma lista de requisitos funcionais e não funcionais que atenda às necessidades do cliente com relação ao sistema a ser desenvolvido;
c) [1 ponto] Desenhe um diagrama de casos de uso que satisfaça a esta descrição;
d) [1 ponto] Escolha um dos casos de uso identificados e realize a sua descrição, conforme o *template* da Tabela 1;
e) [2 pontos] A partir do caso de uso escolhido, construa o(s) Diagrama(s) de Sequência e o(s) Diagrama(s) de Estado correspondentes;
f) [1 ponto] Construa o Modelo de Classes conceitual (artefato resultante da fase *Análise Orientada a Objetos*), mantendo-o coerente com as entidades utilizadas.
g) [2 pontos] Explique detalhadamente o que grau de dependência, bem como suas variações, e relacione este conceito com os tipos de domínios de classes de um sistema. Além disso, calcule o grau de dependência de cada uma das classes apresentadas no Diagrama de Classes conceitual construído em (f).

Tabela 1 – *Template* para Descrição de Casos de Uso

Nome:	<definir o nome do caso de uso>
Objetivo:	<descrever o objetivo do caso de uso>
Atores:	<descrever os atores que interagem com o caso de uso>
Pré-condições:	<descrever as pré-condições a serem atendidas para que o caso de uso possa ser executado>
***Trigger*:**	<definir que evento dispara a execução desse caso de uso>
Fluxo Principal:	<descrever o fluxo principal do caso de uso>
Fluxo Alternativo:	<descrever os fluxos alternativos do caso de uso, indicando que evento dispara cada um deles. Cada fluxo deve ser nomeado A1, A2 etc.>
Extensões:	<definir que extensões podem ser executadas>
Pós-condições:	<definir que produto ou resultado concreto o ator principal obterá ao final da execução do fluxo básico>
Regras de negócio:	<listar as regras de negócios que devem ser respeitadas na execução do caso de uso. Cada regra deve ser nomeada RN1, RN2 etc., e ser referenciada em algum fluxo do caso de uso (básico ou alternativo)>

Resposta:

a) O Sistema de Manutenção e Controle de Contas Correntes e Aplicações Financeiras (SM3CAF) automatiza as operações realizadas por uma conjuntos de agências de um banco, cada uma delas descritas por nome, código e endereço. O sistema apresenta módulos para cadastro, alteração, consulta e remoção de cliente e respectivos dependentes, contas correntes e poupanças. O cadastro de cliente exige um conjunto de dados como nome completo, telefone fixo e celular, RG, CPF, senha de acesso ao sistema (i.e., às contas a serem cadastradas), endereço e *e-mail*, além da existência ou não de um ou mais dependentes, descritos pelos mesmos dados de um cliente. O cliente pode possuir uma ou mais contas correntes e contas de poupanças, mas os seus dependentes só podem possuir contas poupanças e estarem nesta situação apenas

quando estiverem associados a um cliente. Uma conta corrente é descrita em termos do número de identificação, do código da agência à qual está vinculada e do valor monetário, ao passo que uma conta de poupança é descrita em termos da variação, do valor monetário, de uma ou mais aplicações, que possuem um prazo de aplicação considerando uma data de vencimento (aniversário) e uma taxa de juros do período. Uma conta corrente pode ter ou não uma ou mais aplicações pré-fixadas relacionadas a ela, de modo que uma aplicação pré-fixada é descrita em termos do valor, e do prazo e da taxa de juro pré-definidos. Tanto a conta corrente quanto a conta de poupança apresentam um histórico que considera as seguintes movimentações: crédito, débito, transferência e aplicações de pré-fixados (esta apenas para conta corrente).

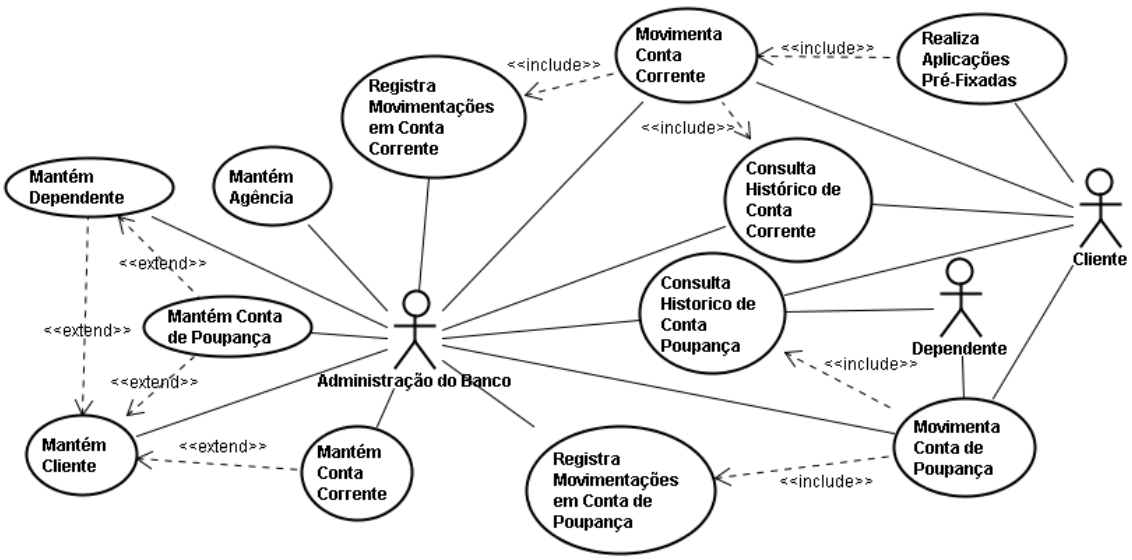
b) Lista de requisitos extraídos da Descrição do SM3CAF:

REQUISITOS FUNCIONAIS	
Nº	**DESCRIÇÃO**
RF1:	O sistema deve manter uma agência bancária. Por manter, entende-se cadastrar, alterar, consultar ou remover uma agência.
RF2:	O sistema deve manter um cliente.
RF3:	O sistema deve manter um dependente.
RF4:	O sistema deve permitir que um cliente tenha dependentes associados, e que dependentes existam apenas quando associados a um cliente.
RF5:	O sistema deve manter uma conta corrente, desde que associada a um cliente.
RF6:	O sistema deve manter uma conta de poupança, desde que associada a um cliente ou dependente.
RF7:	O sistema deve permitir que um cliente tenha uma ou mais contas correntes.
RF8:	O sistema deve permitir que um cliente tenha uma ou mais contas de poupança.
RF9:	O sistema deve permitir que um dependente tenha uma ou mais contas de poupança.
RF10:	O sistema deve permitir que o cliente movimente suas contas correntes. Por movimentar uma conta corrente, entende-se debitar, creditar, transferir para outra conta (corrente ou poupança) ou realizar aplicações pré-fixas a partir de uma conta corrente.
RF11:	O sistema deve permitir que o cliente ou dependente movimentem suas contas de poupança. Por movimentar uma conta de poupança, entende-se debitar, creditar ou transferir para outra conta (corrente ou poupança) a partir de uma conta de poupança.
RF12:	O sistema deve permitir que uma ou mais aplicações pré-fixadas sejam feitas a partir de uma conta corrente.
RF13:	O sistema deve registrar todas as movimentações realizadas em contas correntes. Por movimentações em conta corrente, entende-se débito, crédito, transferência e aplicações pré-fixas a partir de uma conta corrente.

RF14:	O sistema deve registrar todas as movimentações realizadas em contas de poupança. Por movimentações em conta de poupança, entende-se débito, crédito e transferência.
RF15:	O sistema deve permitir que o histórico de uma conta corrente seja consultado, detalhando as suas movimentações em um período de tempo pré-determinado.
RF16:	O sistema deve permitir que o histórico de uma conta de poupança seja consultado, detalhando as suas movimentações em um período de tempo pré-determinado.

REQUISITOS NÃO FUNCIONAIS	
Nº	**DESCRIÇÃO**
RNF1:	O sistema deve ter tempo de resposta menor ou igual a 30 segundos.
RNF2:	O sistema deve estar disponível 24 (vinte e quatro) horas por dia, 7 (sete) dias por semana (24x7), com a possibilidade de paradas pré-programadas (que impeçam o acesso pelos usuários do sistema) para manutenção no mesmo ou na infraestrutura de apoio (servidores, sistemas operacionais, SGBD, servidor web etc.), preferencialmente entre 22h e 6h.
RNF3:	O sistema deve realizar backup diário automaticamente entre 22h e 6h.
RNF4:	Nas funcionalidades que demandarem algum tipo de interação entre o sistema e o usuário (ou seja, na interface humano-computador), todas as falhas do sistema devem ser apresentadas seguindo o mesmo padrão de interface do próprio sistema através da exibição de mensagens apropriadas ao erro ocorrido.
RNF5:	A interface humano-computador deve apresentar ícones e textos sem se tornar cansativa aos usuários mais experientes, apresentando módulos de ajuda aos usuários associados às funcionalidades do sistema.

c) Diagrama de Casos de Uso do SM3CAF:



d) Descrição do Caso de Uso *Mantém Cliente*:

Nome: Mantém Cliente

Objetivo: Realizar cadastro, alteração, consulta ou remoção de clientes na base de dados do sistema.

Atores: Administração do Banco

Pré-condições: A administração do banco deve fazer *login* em uma área restrita do sistema.

***Trigger*:** Uma solicitação para manutenção de um cliente é realizada.

Fluxo Principal:

1. A administração do banco informa ao sistema o nome do cliente e/ou CPF.
2. O sistema verifica a existência do cliente.
3. O sistema exibe as opções "cadastrar" (apenas no caso em que o cliente não esteja cadastrado no sistema), "alterar", "consultar" e "remover cliente".
4. A administração do banco selecionada uma das opções.

Sub-fluxo: Cadastrar cliente

4.1 A administração do banco informa ao sistema os dados do cliente:
- Nome completo
- Telefone fixo
- Telefone celular – *opcional*.
- RG
- CPF
- Endereço
  - Rua, Número, Complemento, Bairro e CEP
  - Cidade, Estado e País
- Endereço eletrônico (e-mail) – *opcional*

4.2 A administração do banco confirma o cadastro das novas informações na base de dados do sistema.

4.3 O sistema gera um código de confirmação do processamento. [a]

Sub-fluxo: Alterar cliente

4.1 A administração do banco altera os dados desejados, dentre aqueles listados no passo 3 do sub-fluxo *Cadastrar cliente*.

4.2 A administração do banco confirma a alteração das informações na base de dados do sistema. [a]

Sub-fluxo: Consultar cliente

4.1 O sistema exibe os dados do cliente.

Sub-fluxo: Remover cliente

4.1 A administração do banco confirma a remoção das informações na base de dados do sistema.

4.2 O sistema verifica as dependências de sistema relacionadas ao cliente. [a]

5. O sistema gera um relatório de atualização.

Fluxo Alternativo:

[a] Uma falha acontece no processamento da solicitação

1 O sistema envia um *e-mail* à administração do banco informando a falha ocorrida.

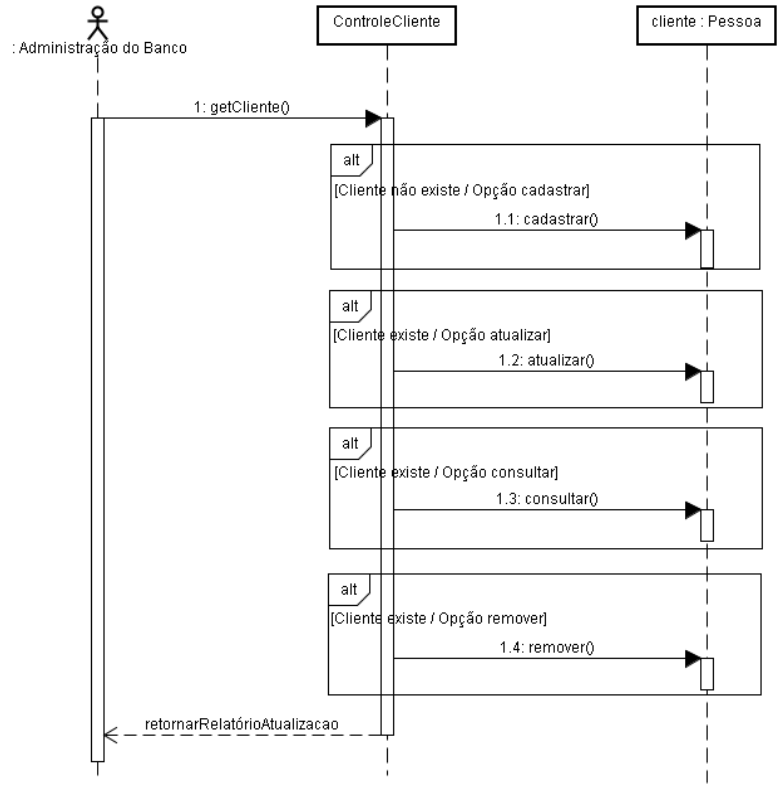
Extensões: Mantém Dependente, Mantém Conta Corrente, Mantém Conta de Poupança.

Pós-condições: O cliente é mantido no sistema com sucesso.

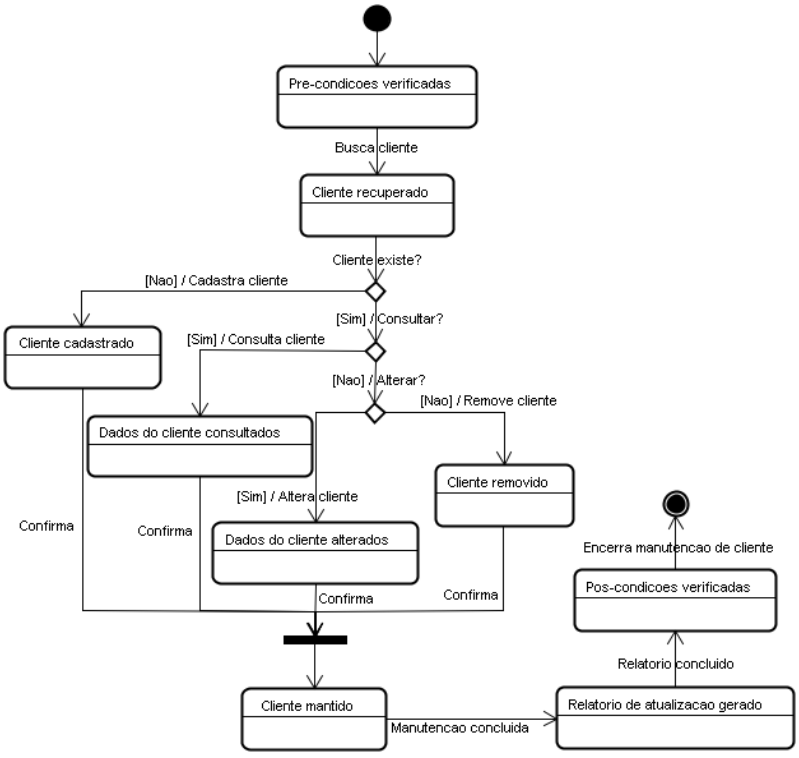
Regras de negócio: RN01 – O sistema deve ser capaz de realizar a manutenção de um cliente na base de dados do sistema.

e) Diagramas do SM3CAF relativos ao Caso de Uso *Mantém Cliente*:

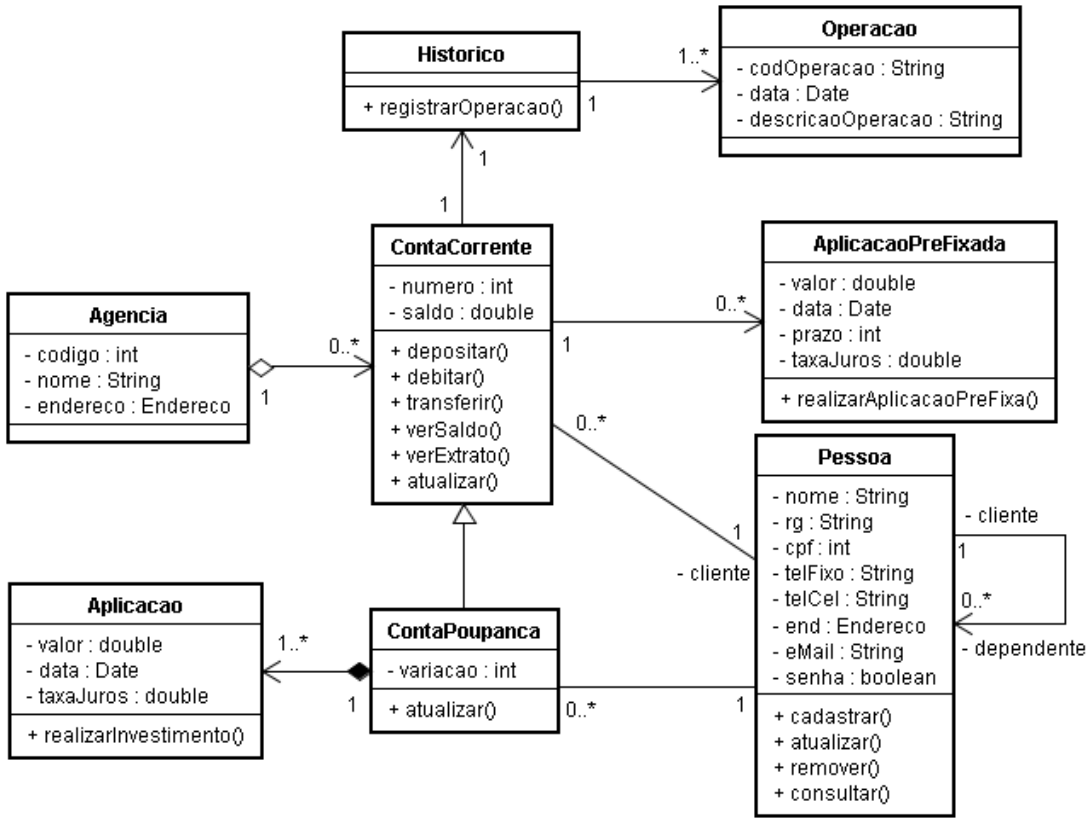
a. Sequência:



b. Transição de Estados:



f) Modelo de Classes Conceitual do SM3CAF:

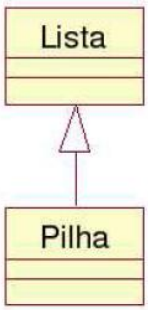


g) Grau de Dependência (GD) é uma métrica semelhante a *Fan-out* (i.e., indica quantos módulos são acessados por um dado módulo) de projeto estruturado e serve para verificar projetos orientados a objetos. Pode ser classificado em dois tipos: (i) *GD direto*, que indica quantas classes são referenciadas diretamente por uma determinada classe; e (iii) *GD indireto*, que indica quantas classes são referenciadas direta ou indiretamente (recursivamente) por uma determinada classe. Para identificar o GD, deve-se verificar se uma classe *A* referencia diretamente outra classe *B*, de maneira que *A* seja subclasse (ou tenha atributo, ou tenha parâmetro do método do tipo, ou tenha variáveis em métodos do tipo, ou ainda chame métodos que retornem valores do tipo) de *B*. Analisando os diferentes tipos de domínios de classes de um sistema, temos que classes de domínios mais altos (negócio e aplicação) tendem a ter alto GD indireto e as classes de domínios mais baixos (arquitetura e base) tendem a ter baixo GD indireto. Para o cálculo de GD na ocorrência de agregação e composição, estamos considerando a navegação no sentido todo → parte. Cálculo do grau de dependência:

CLASSES	GD Direto	GD Indireto
Agencia	1	7
Pessoa	2	6
ContaCorrente	3	6
ContaPoupanca	3	6
Aplicacao	0	0
AplicacaoPreFixada	0	0
Historico	1	1
Operacao	0	0

Questão 2 [1 ponto]

Para o modelo do projeto orientado a objetos desenhado abaixo, explique qual é o perigo detectado e associe-o diretamente com a heurística que apóia o projetista a evitá-lo:



Resposta:

Perigo detectado nos modelos: utilização inadequada da herança (herança forçada).

Heurística que apoia o projetista a evitar este perigo: "Se duas ou mais classes compartilham somente características, não deve ser utilizada herança". *Raciocínio*: herança está fortemente relacionada com o compartilhamento de comportamento (uma pilha não deve ter comportamentos como *inserirNoFinal*, *inserirNoMeio*, *removerDoMeio*, *removerDoFinal* e *alterar*, que são válidos para uma lista). Neste caso, poderíamos adotar como solução a utilização da delegação, a qual permite o compartilhamento baseado em objetos. Dessa forma, este mecanismo permitiria o repasse da mensagem para o outro objeto. O objeto "delegador" do serviço, do tipo da classe `Pilha`, conteria uma referência para o objeto responsável pela execução, no caso aquele do tipo da `Lista`, podendo obter resultados dos métodos da classe `Lista` que a classe `Pilha` referencia. Ou seja, a implementação do método `empilhar()` conteria uma chamada para o `inserir(0)`, da classe `Lista`, e o método `desempilhar()`, uma chamada para o método `remover(0)`, a fim de se manter a disciplina que rege a estrutura de dados pilha: primeiro a chegar é o último a sair (*last in, first out*):