Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Arquitetura e Projeto de Sistemas II
Gabarito da AP3 2015/2

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

Questão 1 (2 pontos)

Em relação ao processo unificado, responda com suas palavras:

(a) O que é desenvolvimento iterativo, desenvolvimento evolutivo e desenvolvimento ágil (1 ponto)?

Resposta: Desenvolvimento iterativo consiste em decompor o projeto em mini-projetos, de forma que cada mini-projeto possa ser apresentado ao usuário. Desenvolvimento evolutivo consiste em aceitar que o conhecimento do usuário sobre os requisitos do software evolui a cada iteração. Por fim, desenvolvimento ágil consiste em acolher essas mudanças nos requisitos de forma rápida e flexível para que o software se mantenha aderente às necessidades dos usuários.

(b) Como se relacionam as fases de Concepção, Elaboração, Construção e Transição com as atividades de Análise, Projeto, Programação e Testes? Justifique a resposta com exemplos (1 ponto).

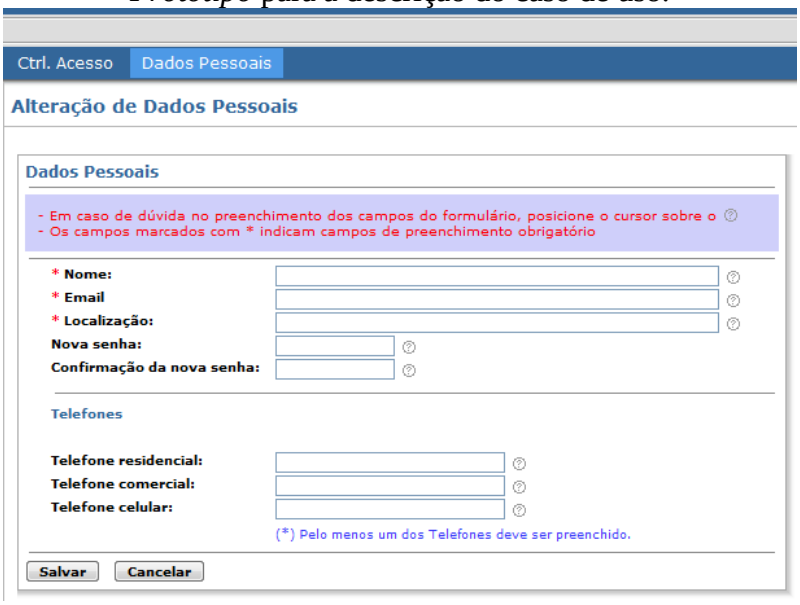
Resposta: As fases englobam iterações com propósitos semelhantes. Como iterações usualmente contém as atividades de Análise, Projeto, Programação e Testes em intensidades distintas, cada uma das fases têm maior ênfase em um das atividades, apesar de contemplar também as demais. Por exemplo, a fase de Concepção enfatiza a Análise,

apesar de ter Projeto, Programação e Teste em menor intensidade. Por outro lado, a fase de Elaboração enfatiza o Projeto, apesar de ter as atividades de Análise, Programação e Teste em menor intensidades. O mesmo vale para Construção e Transição em relação a Programação e Testes, respectivamente.

Questão 2 (2 pontos)

Com relação à especificação funcional de sistemas utilizando casos de uso, com base no protótipo e no template fornecidos, descreva o caso de uso que permitirá aos usuários de um sistema a manutenção dos dados pessoais.

Protótipo para a descrição do caso de uso.



Template para a descrição do caso de uso.

Nome:	<definir o nome do caso de uso>
Objetivo:	<descrever o objetivo do caso de uso>
Atores:	<listar os atores que interagem com o caso de uso>
Trigger:	<definir que evento dispara a execução desse caso de uso>
Fluxo Principal:	<descrever passos numerados do fluxo principal do caso de uso>
Fluxo Alternativo	<descrever os passos dos fluxos alternativos do caso de uso, indicando que evento dispara cada um deles. Cada fluxo deve ser nomeado <Numero do fluxo principal>.<Numero do fluxo alternativo>. Exemplo: 3.1, 3.2, 4.1.
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 passos do fluxo principal e/ou alternativo.

Resposta:

Nome:	Manter Dados Pessoais
Objetivo:	Permitir aos usuários de um sistema realizar a manutenção de seus dados pessoais
Atores:	Usuário
Trigger:	O ator seleciona a opção 'Dados Pessoais' do menu.
Fluxo Principal:	1. Sistema apresenta uma tela contendo os seguintes campos para edição [RN1] [RN2]: - Nome - E-mail - Localização - Nova senha - Telefone residencial - Telefone comercial - Telefone celular Além das opções 'Salvar' e 'Cancelar'. 2. O ator preenche os campos e seleciona a opção 'Salvar'. 3. O sistema armazena os dados e apresenta a mensagem 'Dados atualizados com sucesso' e o caso de uso é encerrado.
Fluxo Alternativo	2.1 O ator seleciona a opção 'Cancelar'. 2.1.1. O sistema apresenta a mensagem 'Operação Cancelada' e o caso de uso é encerrado.
Regras de negócio:	RN1. Os nomes nome, e-mail e localização são de preenchimento obrigatório. RN2. Pelo menos um dos telefones deve ser preenchido.

Questão 3 (3 pontos)

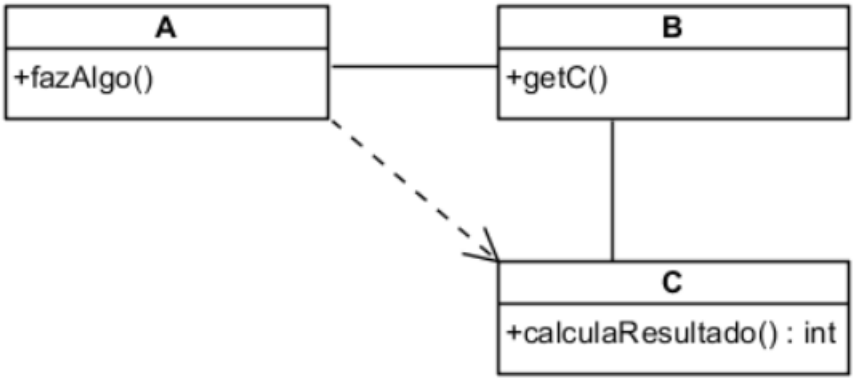
Em relação aos conceitos de projeto orientado a objetos:

(a) A respeito dos princípios 'Baixo Acoplamento' e 'Alta Coesão', comente a seguinte frase: "Baixo Acoplamento e Alta Coesão são o Yin e Yang do projeto de software". (1 Ponto)

Resposta: Em um projeto que possui alta coesão os dados e operações semanticamente relacionados estão na mesma classe, com isto naturalmente o acoplamento é reduzido, já que a classe não precisará se relacionar com outras classes para obter dados ou invocar operações necessários para cumprir suas responsabilidades dentro do sistema. Por outro lado, uma das formas de reduzir o acoplamento é justamente focando na distribuição adequada de responsabilidades, de modo que dados e operações fiquem em unidades fortemente coesas.

(b) Aplique a Lei de Demeter (princípio "não fale com estranhos") para fornecer uma solução que reduza o acoplamento do diagrama de classes abaixo, em que a dependência entra as classes A e C é causada pelo seguinte trecho de código do método público fazAlgo() da classe A:

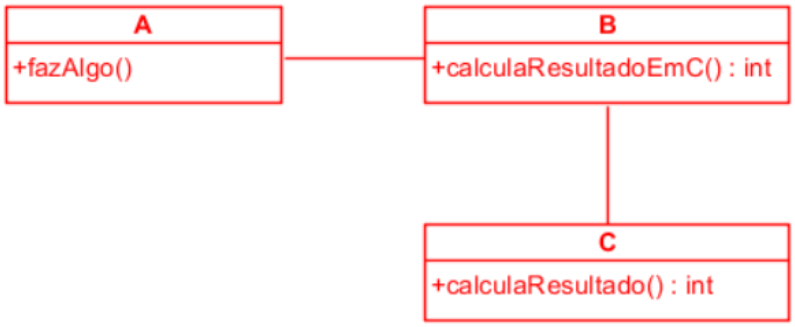
```
public class A {
    private B b;
    ...
    public void fazAlgo(){
        ...
        int resultado = b.getC().calculaResultado();
    }
}
```



Sua solução deverá conter o novo diagrama de classes (1 Ponto) e o novo código do método fazAlgo() da classe A (1 Ponto).

Resposta:

Novo diagrama:



Novo código do método fazAlgo:

```
public class A {
    private B b;
    ...
    public void fazAlgo(){
        ...
        int resultado = b.calculaResultadoEmC();
    }
}
```

```
// Apenas complementando a solução (não foi pedido). O método
// calculaResultadoEmC() da classe B ficará da seguinte forma:

Public class B {
    private C c;
    …
    public int calculaResultadoEmC(){
        return c.calculaResultado();
    }
}
```

Note agora o princípio 'Não fale com estranhos' se aplica, já que a classe A só fala com B e a classe B só fala com C, eliminando a dependência entre as classes A e C e assim reduzindo o acoplamento.

Questão 4 (3 pontos)

Em relação a componentes, responda com suas palavras:

(c) Qual tipo de componente é mais reutilizável: de negócio ou de infraestrutura? Justifique a sua reposta (1 ponto).

Resposta: Componentes de infraestrutura são mais reutilizáveis, pois eles são úteis a diversos domínios. Por outro lado, componentes de negócio são dependentes a um domínio específico.

(d) Em uma situação onde você necessita de um componente que converta reais em dólar via método *Dinheiro real2dolar(Dinheiro real)*, mas você só encontrou um componente genérico chamado "Conversor" que converte qualquer tipo de moeda via método *Dinheiro converte(Dinheiro valor, String moedaOrigem, String moedaDestino)*, qual técnica de adaptação de componentes deveria ser usada? Exemplifique como ficaria essa adaptação (1 ponto).

Resposta: Para essa situação, a técnica de embrulho deveria ser usada. A adaptação ficaria da seguinte forma:

```
public Dinheiro real2dolar(Dinheiro real) {
    return Conversor.converte(real, "Real", "Dolar");
}
```

(e) Supondo que você trabalha em um banco que usa um componente "Operacoes" para os cálculos de operações financeiras. Ainda, suponha que o governo decidiu voltar com a CPMF no valor de 0,4% sobre todas as operações financeiras, e você foi o escolhido por implementar isso no sistema. Ao investigar o componente "Operacoes", que foi feito usando OO, você descobriu um método protegido *Dinheiro recolheImposto(Dinheiro valor)* é responsável por fazer o recolhimento de impostos sobre um dado valor. Esse método é usado por todas as operações

financeiras do componente “Operacoes”. Qual técnica de adaptação de componentes deveria ser usada para fazer com que o componente “Operacoes” passe a recolher CPMF? Exemplifique como ficaria essa adaptação (1 ponto).

Resposta: Para essa situação, a técnica de herança deveria ser usada. A adaptação implicaria na criação de uma classe que herdasse da classe que contém o método *Dinheiro recolheImposto(Dinheiro valor)* sobrescrevendo seu comportamento. O novo comportamento ficaria da seguinte forma:

```
protected Dinheiro recolheImposto(Dinheiro valor) {
    return super.recolheImposto(valor) + 0,004 * valor;
}
```