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 – AD2 2° semestre de 2012.

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 e APs de períodos anteriores, **será atribuída a nota ZERO**, incluindo também cópias diretas e sem sentido de tópicos dos slides das aulas.

Questão 1 [5 pontos]

O padrão *Information Expert* visa atribuir a responsabilidade ao especialista, isto é, à classe que tem a informação necessária para satisfazer a responsabilidade.

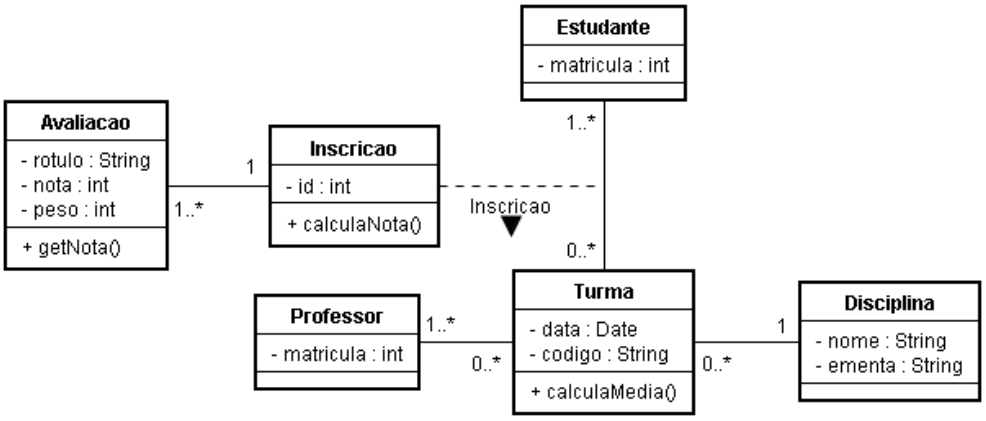
a) [1 ponto] Descreva um exemplo, **diferente daquele visto em aula**, de uma situação onde esse padrão é útil.
b) [2 pontos] Quais são as possíveis **dificuldades** com o uso desse padrão?
c) [2 pontos] Desenhe um ou mais **modelos (classes** e/ou **sequência)** exibindo como o padrão pode ser utilizado na situação descrita em (a).

Resposta:

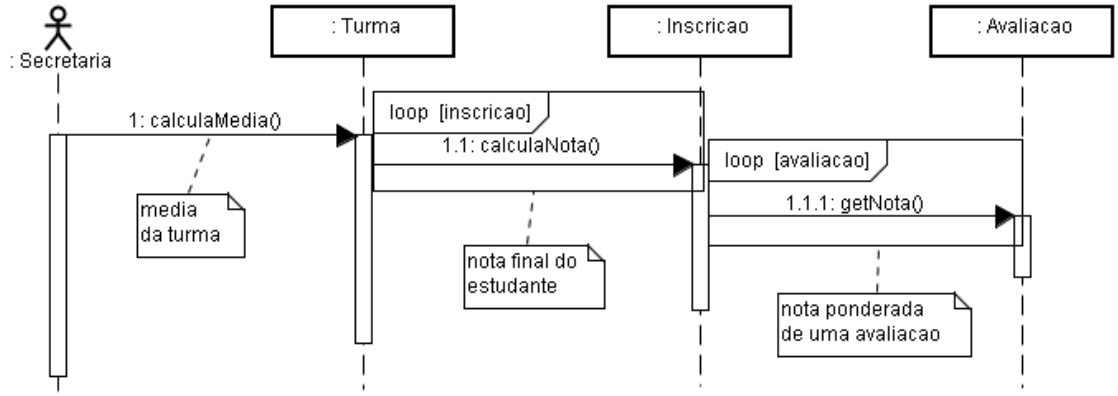
a) Em um sistema de gestão acadêmica, o responsável pelo cálculo da média geral de uma turma de uma disciplina deveria ser a classe `Turma`, considerando o padrão *Information Expert*. Isso se deve ao fato de que deve-se atribuir a responsabilidade ao especialista, ou seja, à classe que tem a informação necessária para satisfazer a responsabilidade (no caso, o cálculo da média geral de uma turma de uma disciplina). Neste caso, pode-se estar interessado em entender como uma disciplina se comporta ao longo dos semestres em determinada universidade, a partir do desempenho global dos alunos em cada uma das suas diferentes turmas.

b) Entre as dificuldades, às vezes, ao definir as responsabilidades, responsabilidades de mais baixo nível não são cobertas, e.g., qual a melhora ou piora de cada aluno na disciplina no decorrer do tempo, considerando as suas turmas. Além disso, a aplicação deste padrão não é desejável em todas as situações. Por exemplo, considere a situação de persistir um objeto de `Estudante` em um banco de dados. De acordo com este padrão, a classe `Estudante` deveria ter esta responsabilidade. No entanto, pela experiência dos programadores, é sabido que esta situação não é trivial ou mesmo desejada, e.g., em aplicações *web*, uma solução seria o *Data Access Layer* assumir esta responsabilidade ao prover métodos como *persisteEstudante()* em alguma classe *controller*.

c) Para o seguinte modelo:



a aplicação do padrão supracitado para o cálculo da média geral de uma turma de uma disciplina em um sistema de gestão acadêmica pode ser visualizado pelo diagrama de sequência abaixo:



Questão 2 [5 pontos]

Em relação a Componentes, responda:

a) [3 pontos] Explique o conceito de componente de forma a incluir pelo menos **cinco** de suas propriedades fundamentais. Além disso, faça um desenho que exemplifique estas propriedades, seja exemplificando-as por meio de um exemplo de componente, seja montando um modelo ou analogia para descrevê-las.

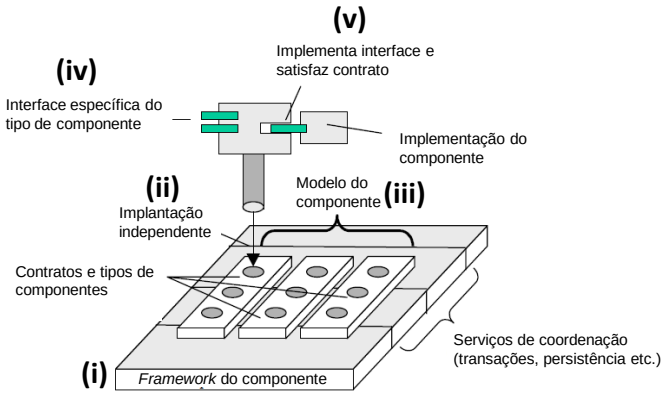
b) [2 pontos] No que se refere ao escopo do projeto de um componente, o que é mais vantajoso ou interessante: *um componente com muitas funcionalidades, ou com poucas funcionalidades*? Argumente a sua resposta **com suas palavras**.

Resposta:

a) Componente é uma unidade de software independente, que encapsula dentro de si seu projeto e implementação, oferecendo serviços por meio de interfaces bem definidas. A seguir, são explicadas as cinco propriedades fundamentais destacadas na ordem em que foram sublinhadas:

i. **Identificação**: componentes devem ser facilmente identificados, ou seja, devem estar armazenados em um único lugar, ao invés de espalhados e misturados com outros artefatos de software ou documentação;
ii. **Autocontido**: característica dos componentes de poderem ser reutilizáveis sem a necessidade de incluir ou depender de outros componentes. Caso haja alguma dependência, todo o conjunto deve ser abstraído como um componente reutilizável;
iii. **Documentação**: A existência de documentação é indispensável para a reutilização. O tipo de componente e a sua complexidade indicarão a conveniência do tipo de documentação;
iv. **Funcionalidade**: componentes têm uma funcionalidade clara e específica que realizam e/ou descrevem. Assim, componentes podem realizar funções ou podem ser simplesmente descrições de funcionalidades (e.g., artefatos do ciclo de vida);
v. **Interface**: componentes devem possuir interfaces claras, que indicam como estes podem ser reutilizados e conectados a outros componentes, devendo-se ocultar os detalhes que não são necessários para a reutilização per si (encapsulamento).

Um desenho que exemplifica estas propriedades montando um modelo ou analogia para descrevê-las é apresentado abaixo[1]:



[1] Maiores informações sobre Engenharia de Software Baseada em Componentes e sobre o modelo apresentado podem ser obtidas em: SANTOS, R.P., 2010, *Brechó-VCM: Uma Abordagem Baseada em Valor para Mercados de Componentes*. Dissertação (M.Sc.), COPPE/UFRJ, Rio de Janeiro, RJ, Brasil.

b) Dependendo do contexto do componente e/ou do projeto e da experiência do desenvolvedor e/ou organização, por exemplo, diversas vantagens (e respectivas desvantagens justificadas sobre cada uma dessas vantagens) para cada uma das escolhas mencionadas poderiam ser apontadas. No entanto, analisar o escopo de um componente requer uma análise de contexto e experiência. Um componente de conversão de moeda para um sistema *web* de uma companhia aérea é bem delimitado em seu escopo, isto é, possui uma funcionalidade e é pequeno em termos da medida de uma métrica como LoC (linhas de código). Por outro lado, um componente de busca e recuperação de voos em sistemas de diversas companhias aéreas pode possuir mais de uma funcionalidade e ser médio ou grande em termos da medida da métrica LoC, no contexto de um sistema de venda de voos. Ou seja, esse componente desempenha a função mais importante deste sistema, com milhares ou milhões de chamadas de outros sistemas ou subsistemas e com vários requisitos não funcionais como disponibilidade e confiabilidade. É bem delimitado em seu escopo e, diante de outros componentes deste sistema (e.g., transação financeira, autenticação de usuários etc.), pode parecer oferecer poucas funcionalidades e até ser pequeno em termos de própria medida mencionada, no contexto em que está inserido. Assim, o vantajoso, ou interessante, é que o componente atenda ao conceito mostrado em seu conceito em (a), apresentando um escopo bem definido em seu projeto e construção.