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 1° 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 [3 pontos]

O padrão *Facade* visa prover uma interface única para um conjunto de interfaces de um subsistema, facilitando o seu uso.

a) [1 ponto] Descreva um exemplo, **diferente daquele visto em aula**, de uma situação onde esse padrão é útil.
b) [1 ponto] Quais são os benefícios alcançados com o uso desse padrão?
c) [1 ponto] Desenhe um modelo de classes exibindo como o padrão pode ser utilizado na situação descrita.

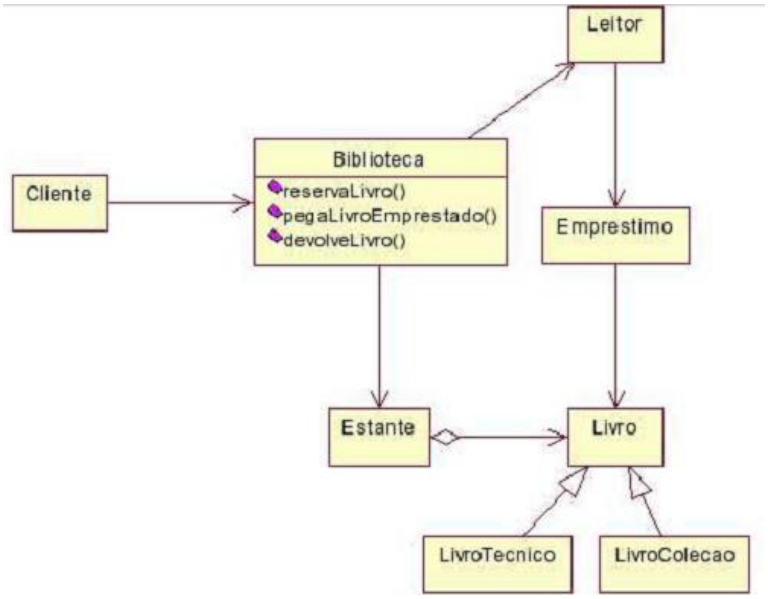
Resposta:

a) Um possível exemplo de situação onde esse padrão é útil pode ser verificado no projeto da interação entre um sistema cliente com um sistema de controle de biblioteca. Neste caso, visando prover uma interface única para um conjunto de interfaces de um subsistema, um conjunto de classes (e.g., `Estante`, `Leitor`, `Empréstimo` e `Livro`) deve se comportar como um componente, de maneira que o cliente deva falar com uma única interface (i.e., `Biblioteca`). Dessa forma, operações como `reservaLivro()`, `pegarLivroEmprestado()` e

`devolveLivro()` estariam concentradas na classe `Biblioteca`, facilitando seu uso, compreensão e manutenção.

b) Dentre os benefícios alcançados com a utilização do padrão *Facade*, pode-se citar: (i) separa a responsabilidade da criação complexa em objetos auxiliares coesos; (ii) oculta a lógica de criação potencialmente complexa; e (iii) permite a introdução de estratégias de gerenciamento de memória para a melhoria do desempenho, como o uso de *cache* ou reciclagem de objetos.

c) Modelo de classes correspondente à situação apresentada em (a):



Questão 2 [3 pontos]

Dê a sua avaliação sobre o acoplamento nos problemas abaixo e indique se é uma situação desejável.

(a) [1,5 ponto] **CENÁRIO A**: um sistema onde todos os objetos herdam da classe `ObjetoPersistente`, para que seja possível implementar um mapeamento objeto-relacional automático;
(b) [1,5 ponto] **CENÁRIO B**: um sistema onde as classes têm somente atributos primitivos e métodos, sem nenhuma associação ou herança;

Resposta:

a) Uma subclasse é fortemente acoplada à sua superclasse. Por isso, deve-se considerar cuidadosamente qualquer decisão de derivar de uma superclasse, pois é uma forma muito forte de acoplamento. O cenário apresentado é funcional, ou seja, não apresenta uma situação incorreta. No entanto, seguindo a ideia do padrão *Low Coupling*, a desvantagem desse uso de subclasses é que ele acopla fortemente objetos do domínio a um

determinado serviço técnico e mistura diferentes interesses arquiteturais. Isso faz com que esses objetos estejam atrelados e dependam de um mecanismo de persistência específico. Em caso de descontinuidade do projeto desse mecanismo ou evolução não cuidadosa, ou ainda de manutenção corretiva, o sistema como um todo sofrerá seus reflexos (e.g., quando da necessidade de migração para outro mecanismo de persistência, esses objetos do domínio demandarão adaptações, tornando-se oneroso para o projeto do sistema). Por outro lado, uma vantagem estaria na herança automática de comportamento de persistência, buscando facilitar o tratamento deste interesse. Enfim, esse cenário é representativo de um caso de alto acoplamento.

b) Esse cenário representa o caso extremo de aplicação do padrão *Low Coupling*, no qual não há acoplamento entre classes. Isso não é desejável, porque contradiz uma metáfora central da tecnologia de objetos: um sistema é composto de objetos conectados que se *comunicam* por mensagens. Se esse padrão for aplicado em demasia, levará a um projeto ruim, com poucos objetos ativos, sem coesão, imprecisos e complexos, que fazem todo o serviço, e com muitos objetos passivos, com acoplamento zero, que agem como simples repositórios de dados. Ou seja, um grau de acoplamento entre classes é normal e necessário para criar um sistema orientado a objetos no qual as tarefas são executadas por uma colaboração entre objetos conectados. Dessa forma, este cenário não é funcional.

Questão 3 [4 pontos]

Considerando as arquiteturas *web* responda:

a) [2 pontos] Quais os seus componentes fundamentais? Explique um canário típico de comunicação em um sistema *Web* que apresenta arquitetura em três camadas.
b) [2 pontos] Selecione dois estilos arquiteturais para *Web* e explicite suas características, vantagens e limitações.

Resposta:

a) Os componentes fundamentais de arquiteturas *web* são: HTML (páginas *web* estáticas), componentes que rodam no cliente (e.g., Applets e ActiveX), scripts que rodam no cliente (e.g., JavaScript), scripts que rodam no servidor (e.g., PHP, ASP, JSP), componentes que rodam no servidor (e.g., Servlets e EJB) e banco de dados.

Um cenário típico de comunicação em um sistema *web* que apresenta arquitetura em três camadas poderia ser: inicialmente, o cliente no *browser* solicita um serviço ao servidor usualmente após o preenchimento de um formulário HTML; a seguir, o servidor interpreta a solicitação na camada de aplicação; se necessário, a camada de aplicação se comunica com a camada de armazenamento para acessar dados; assim, a camada de aplicação redireciona o fluxo para a camada de apresentação a fim de que o servidor construa uma página de resposta; por fim, o servidor retorna a página de resposta.

b) *Thin Client*: representa um estilo fundamentado na mínima utilização dos recursos do cliente, exigindo mínima capacidade de seu navegador. Algumas de suas características

são: (i) estilo mais apropriado para aplicações *Internet*; (ii) baixo controle da configuração do navegador cliente; e (iii) toda a lógica de negócio reside no servidor. Suas vantagens: (i) independência total de plataforma ou navegador; e (ii) concentração da lógica de negócio no servidor. Apesar disso, apresenta limitações: (i) interface com o usuário limitada pela linguagem HTML; e (ii) maior número de transações com o servidor.

AJAX: consiste em um estilo baseado na utilização de *scripts* de cliente de forma assíncrona, permitindo a carga parcial de páginas. Algumas de suas características são: (i) permite a carga parcial de dados em uma página e (ii) combina *JavaScript* e XML. Uma vantagem está na carga parcial e assíncrona de dados para o cliente, ao passo que uma limitação se refere à dependência (limitada) com o navegador.