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 – AP3 2° semestre de 2007.

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)

Explique e exemplifique cada um dos seguintes níveis de encapsulamento:

(a) Nível 1: Módulos procedimentais

Resposta: Unidades de encapsulamento que permitam a criação de funcionalidades complexas. Ex.: um procedimento para ordenação escrito em Pascal.

(b) Nível 2: Classes de objetos

Resposta: Unidades de encapsulamento que combinam tanto funcionalidades quanto características. Ex.: uma classe que representa um pedido, contendo atributos referentes a data do pedido, autor e itens, e funcionalidades referentes ao cálculo do total do pedido e emissão de nota fiscal.

(c) Nível 3: Pacotes de classes

Resposta: Unidades de encapsulamento que combinam diversas classes com acessos diferenciados a elas. Ex.: um pacote que contenha classes de manipulação matemática.

(d) Nível 4: Componentes

Resposta: Unidade de encapsulamento que defina de forma contratual interfaces providas e requeridas. Ex.: um componente para conversão de moedas, com interface provida que contenha a funcionalidade de conversão e interface requerida que contenha funcionalidades referentes às taxas do mercado.

Questão 2 (2 pontos)

Explique e exemplifique cada uma das seguintes classificações de interfaces:

(a) Interface com estados ilegais (com vazamento)

Resposta: Contém métodos que deveriam ser privados, pois levam a abstração a um estado ilegal temporariamente. Ex.: método para mover um ponto em uma classe Quadrado.

(b) Interface com estados inapropriados

Resposta: Permite o acesso a estados que não fazem parte da abstração do objeto. Ex.: método para recuperar o N-ésimo item de uma fila.

(c) Interface com comportamento ilegal

Resposta: Permite uma troca de estados ilegal para a abstração em questão. Ex.: método para adicionar um elemento na N-ésima posição de uma fila.

(d) Interface com comportamento replicado

Resposta: Fornece mais de uma forma de se obter um mesmo comportamento. Ex.: método que eleva um número ao quadrado e método que eleva um número a uma determinada potência.

Questão 3 (2 pontos)

Considerando um sistema de venda de livros para web:

(a) Desenhe uma arquitetura de 3 camadas para esse sistema.

Resposta:

(b) Explique e exemplifique cada uma das camadas.

Resposta: A camada de armazenamento é responsável por persistir os dados manipulados pela aplicação. Um exemplo tradicional de implementação dessa camada é a utilização de um sistema de gerenciamento de banco de dados relacional. A camada de aplicação é o local aonde residem as regras de negócio, ou seja, o código executável referente às funcionalidades da aplicação. Um exemplo de implementação dessa camada é a utilização de Servlets em uma arquitetura Java. A camada de apresentação é responsável por fazer interface com o usuário, apresentando e coletando informações. Um exemplo dessa camada é a utilização de JSP em uma arquitetura Java. Vale ressaltar que em uma arquitetura em 3 camadas, a camada de Aplicação é usualmente independente da camada de apresentação.

Questão 4 (2 pontos)

Explique e exemplifique cada um dos seguintes tipos de *frameworks*:

(a) Caixa-preta

Resposta: Nos *frameworks* caixa-preta, o *framework* recebe um conjunto de parâmetros previamente estabelecidos que representa o comportamento específico da aplicação. Este fornecimento de parâmetros é feito por protocolo, através da interface de cada classe do *framework* e, por isso, esta forma de especialização requer que seja conhecida apenas esta interface. Como exemplo, podemos pensar em um sistema de leitor de código de barras, que necessita ser parametrizado com valores como o número de faixas a serem capturadas para que, após essa configuração, o sistema gere um *drive* para determinado tipo de hardware. Dessa forma, não é necessário conhecer o funcionamento interno do sistema para integrá-lo a outros sistemas, mas apenas configurar seus parâmetros para que ele possa ser executado.

(b) Caixa-branca

Resposta: Por outro lado, uma segunda forma de especialização é representada pelos *frameworks* caixa-branca, onde o comportamento específico da aplicação é inserido na arquitetura genérica, somando-se métodos às subclasses de uma ou mais classes do *framework*. Cada método somado a uma subclasse deve estar de acordo com as convenções da respectiva superclasse. Os *frameworks* que fazem uso deste processo de especialização obrigam quem os utiliza a conhecer os seus dispositivos internos. Como exemplo, podemos pensar em um sistema básico que informatize uma rede de supermercados. Este sistema, para ser utilizado, deve ser especializado para a rede de supermercados que deseja utilizá-lo, com a implementação de funcionalidades mais específicas (e.g., módulo de vendas ser especializado para efetuar um balanço geral semanal e comunicar os resultados via e-mail ao gerente de cada supermercado, além da comunicação tradicional já realizada para o gerente

de rede) e com a configuração de quaisquer opções ou parâmetros (taxas de juros para pagamentos de compras à prazo, bonificações incorporadas ao salário dos funcionários, etc.).

Questão 5 (2 ponto)

Explique cada uma das seguintes visões arquiteturais:

(a) Visão lógica

Resposta: Sua principal contribuição é oferecer um retrato estático das classes fundamentais e seus relacionamentos. Dá enfoque aos aspectos funcionais do sistema. A visão lógica é composta por diagramas de classes que apresentem as principais abstrações do sistema: as principais classes e pacotes e os relacionamentos entre eles.

(b) Visão de desenvolvimento

Resposta: Seu principal objetivo é mostrar como o código é organizado em pacotes e bibliotecas. Os pacotes são considerados conjuntos de componentes – unidade de código fonte que serve como um bloco de construção física do sistema – logicamente relacionados, de forma que cada um dos componentes esteja em um único pacote. A visão de desenvolvimento pode incluir novos pacotes para tratarem a funcionalidade de baixo nível.

(c) Visão de processos

Resposta: A visão de processos demonstra as atividades e tarefas realizadas pelo sistema. Focaliza a decomposição do sistema em processos, de forma a exibir a alocação de componentes executáveis a processos e atualizar o diagrama de componentes para apresentar os processos aos quais os componentes são alocados. Outro ponto importante é que a visão de processos dá um enfoque a atributos não funcionais do sistema como disponibilidade, desempenho e confiabilidade.

(c) Visão física

Resposta: A visão física mostra os processadores, dispositivos e ligações no ambiente de operação. Mapeia os processos em unidades de processamento, considerando que um processo é uma linha de controle executada em uma unidade de processamento e que sistemas grandes ou distribuídos podem ser quebrados em diversos processos. Considera tanto requisitos (e.g., capacidade de resposta, desempenho e tolerância a falhas) como peculiaridades de implementação (e.g., necessidade de processadores específicos). Diagramas de implantação (*deployment diagrams*) mostram as unidades de processamento e seus processos.