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 1° semestre de 2008.

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 (3 pontos)

Enumere uma vantagem e uma desvantagem para os seguintes estilos arquiteturais:

(a) *Pipes & Filters*

- Vantagens: simplicidade; e facilidades para compor e paralelizar o sistema;
- Desvantagens: baixo desempenho (abstração de dados primitiva e gerência de *buffers*); e difícil interatividade e cooperação entre filtros.

(b) Camadas

- Vantagens: flexibilidade; e reutilização de uma infra-estrutura de computação pré-existente;
- Desvantagens: determinação do número de camadas; e comprometimento do desempenho devido à comunicação entre as camadas.

(c) Orientado a objetos

- Vantagens: permite a organização de programas; e ajuda a manter a integridade dos dados do sistema;

- Desvantagem: devido à granularidade fina dos objetos e a dependência entre eles, a manutenção e a reutilização podem ser dificultadas, pois para modificar ou reutilizar um objeto pode ser necessário conhecer e atuar sobre outros objetos do sistema.

Questão 2 (2 pontos)

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

(a) Caixa-preta

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

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 3 (3 pontos)

Em relação a componentes,

(a) Quais são as suas principais características?

As principais características de componentes são: serem unidades de composição auto-contidas (encapsulamento), terem interfaces bem definidas (contrato), terem grau de maturidade e documentação (qualidade), serem especificados em diferentes níveis (abstração) e obedecerem a restrições (arquitetura/plataforma).

(b) Quais as desvantagens de utilizar interfaces muito específicas ou muito genéricas?

Interfaces muito específicas fazem com que poucas oportunidades de reutilização do componente possam ser identificadas. Por outro lado, com o uso de interfaces muito genéricas, apesar de os componentes se conectarem com facilidade, o protocolo de comunicação pode ser sobrecarregado.

(c) Quais as técnicas existentes para customização?

A customização de componentes pode ser via variabilidade ou adaptação. A customização via variabilidade consiste em utilizar mecanismos predefinidos para customização. Esses mecanismos são: geração, parametrização e interface de configuração. Caso não seja possível customizar via variabilidade, alguma técnica de adaptação pode ser adotada. Exemplos de técnicas de adaptação são: copiar e colar, herança e embrulho.

Questão 4 (2 pontos)

Na orientação a objetos, o padrão *Creator* determina qual classe deve ser responsável pela criação de objetos. De acordo com esse padrão,

(a) Qual classe do diagrama abaixo deveria ser responsável pela criação de objetos da classe Pagamento? Justifique sua resposta descrevendo os benefícios dessa escolha.

A classe Pedido é a mais indicada para ser responsável pela criação de objetos da classe Pagamento. Com essa escolha, o acoplamento não é aumentado, pois a classe Pedido já se relaciona com a classe Pagamento. Uma segunda opção seria atribuir à classe PDV a responsabilidade de criação de objetos da classe Pagamento. Neste caso, a justificativa seria o fato da classe PDV registrar objetos de Pagamento. Contudo, esta decisão aumentaria a acoplamento do sistema, pois PDV passaria a ter dependência para Pagamento.

(b) Faça um diagrama de seqüência mostrando a criação de objetos da classe Pagamento iniciando no momento em que o caixa informa ao sistema o valor pago.