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 - AP1 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 (2 pontos)

Em relação ao Processo Unificado,

(a) Descreva os seus principais aspectos, explicando cada um dos aspectos.

Resposta:

Desenvolvimento iterativo: consiste em organizar o desenvolvimento em "mini-projetos" – um a cada iteração –, de duração curta e fixa, com atividades de análise, projeto, programação e testes e cujo produto de cada iteração é um software parcial. Possibilita, dessa forma, a verificação constante pelo cliente dos requisitos implementados, com aumento de conhecimento sobre o software e redução de incertezas perante mudanças.

Desenvolvimento evolutivo: as especificações evoluem a cada iteração, com a construção de uma parte de software, de forma que o conhecimento sobre o software aumente. Diante de um cenário de mudanças (requisitos, ambiente e pessoas), a abordagem evolutiva tenta evitar correções, retrabalho e implementações de especificações "desnecessárias" do cliente (requisitos que não são realmente úteis ao software).

Desenvolvimento ágil: fundamenta-se em respostas rápidas e flexíveis a mudanças, com replanejamento contínuo do projeto e entregas incrementais e constantes do software (refletindo tais mudanças). Em relação ao cliente, basta verificar que um dos pontos do manifesto ágil é: "colaboração do cliente vem antes de negociação de contrato".

(b) Descreva os seus benefícios esperados, explicando cada um dos benefícios.

Resposta:

Mitigação de riscos precoce: os casos de uso de maior risco são atacados antes dos demais, possibilitando a diminuição das incertezas.

Visibilidade do progresso: o desenvolvimento iterativo fornece um software parcial a cada iteração.

Envolvimento e comprometimento do usuário: o desenvolvimento ágil visa a proximidade e o comprometimento do usuário com o software que está sendo desenvolvido.

Controle sobre a complexidade: o desenvolvimento iterativo decompõe o projeto em "mini-projetos".

Aprendizado incremental: o desenvolvimento evolutivo assume que o aprendizado aumenta com o passar do tempo, e possibilita que os requisitos sejam refinados.

Menos defeitos: o desenvolvimento iterativo, evolutivo e ágil, com respostas rápidas aos usuários, permite que o produto seja depurado sucessivamente.

Mais produtividade: iterações curtas permitem aumentar o foco da equipe no produto e estabelecer metas de curto prazo, viabilizando o aumento da produtividade.

Questão 2 (3 pontos)

Em relação à Orientação a Objetos,

(a) Qual é o princípio básico da orientação a objetos e qual é o seu objetivo?

Resposta: O princípio básico da Orientação a Objetos (OO) é o encapsulamento e seu objetivo consiste em restringir o escopo da informação para atingir legibilidade, manutenibilidade e reutilizabilidade, congregando abstração visível e implementação escondida a fim de manter uma representação uniforme ao longo de todo o processo de desenvolvimento de software.

(b) Diferencie os conceitos de *classe* e *objeto*.

Resposta: Uma classe é uma representação computacional de entidades ou processos do mundo real. São compostas de atributos (características – informações) e métodos (comportamentos – processos) e instanciam objetos. Por outro lado, um objeto consiste da instanciação de uma classe. Possui um conjunto de serviços (interface) e sua implementação (estruturas de dados – atributos, e implementação de operações – métodos).

(c) É possível aplicar o conceito de polimorfismo sobre atributos? Justifique a sua resposta.

Resposta: Não. Inicialmente, vamos recordar o conceito de polimorfismo: *propriedade derivada do fato de que objetos de diferentes classes podem reagir a uma mesma **mensagem** de forma diferente*. Uma vez que o processo de *troca mensagem* é o paradigma de comunicação seguido em OO, os objetos devem se comunicar apenas por meio de *métodos*, com o intuito de evitar que os dados de um objeto sejam manipulados ou vistos por outro. Somando-se o fato de que atributos representam características/informação relativas a um objeto, o conceito de polimorfismo não teria sentido em ser aplicado a atributos em OO, pois não seria factível que atributos possuíssem "várias formas" dependentes de seu contexto, o que poderia infringir a questão do encapsulamento (princípio básico da OO). Além disso, em conformidade com o conceito de hierarquia de herança, atributos definidos em superclasses são herdados por subclasse de maneira que representem características que respeitem a forma como foram concebidas, ou seja, subclasses podem especializar comportamentos (métodos), mas não características (atributos).

Questão 3 (3 pontos)

Para o diagrama de classes conceitual a seguir,

(a) Determine o grau de dependência direto e indireto para a classe "ItemPedido" em relação às demais classes do diagrama, justificando a resposta via listagem das classes envolvidas em cada uma das contagens de grau de dependência (Lembrete: a navegabilidade nos casos de todo-parte é sempre no sentido do todo para a parte. Ou seja, em situações de composição ou agregação, a parte não conhece o todo. Além disso, a inexistência de setas nas associações representa navegabilidade bidirecional.).

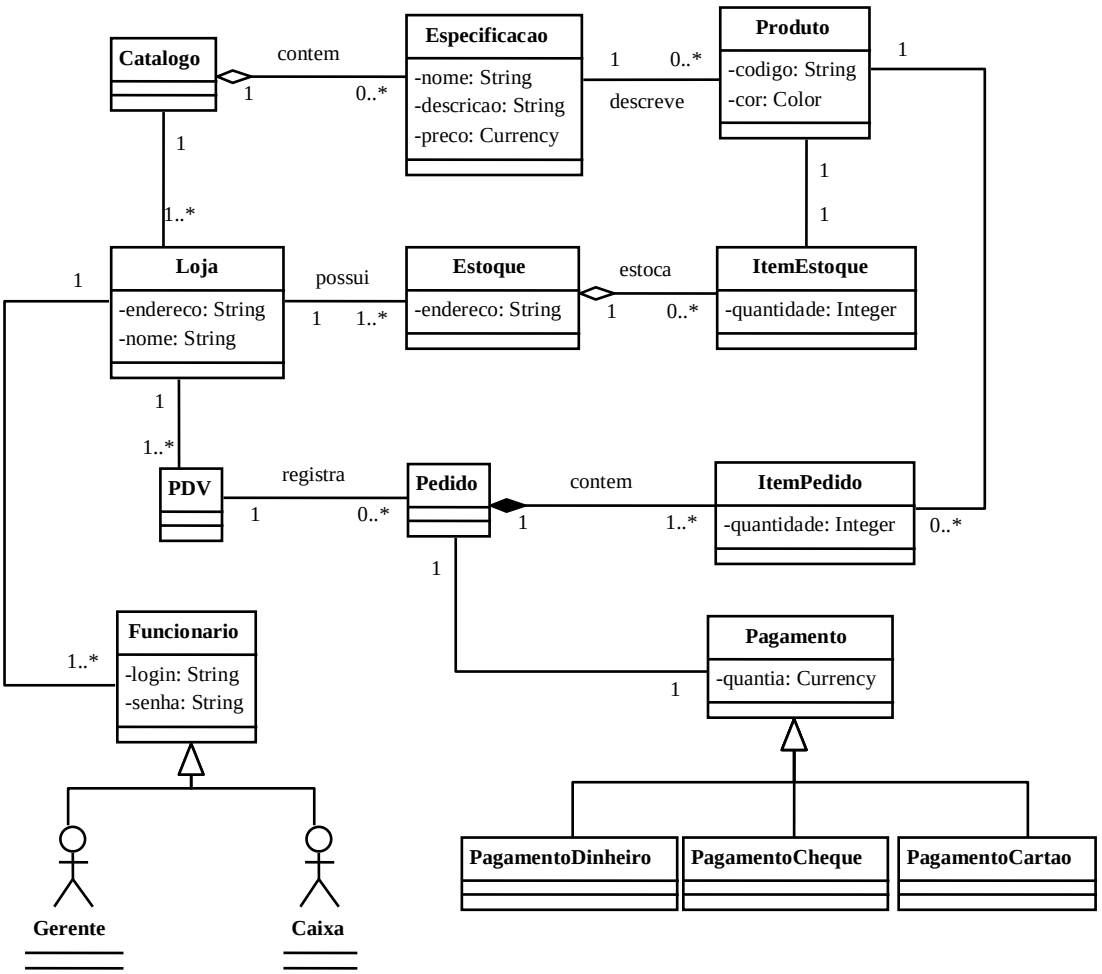
Resposta: Grau de dependência direto para a classe ItemPedido: 1 (classe Produto). Grau de dependência indireto para a classe ItemPedido: 3 (classes Produto, Especificação e ItemEstoque).

(b) Assumindo que o espaço-estado do atributo “quantia” da classe “Pagamento” seja de R$ 50,00 a R$ 500,00, é permitido que a classe “PagamentoCheque” modifique esse espaço-estado para de R$ 0 a R$ 1000,00? Justifique a sua resposta.

Resposta: Não. Tendo em vista que todo PagamentoCheque *é um* Pagamento, e que a variação de espaço-estado em casos de herança deve ser mais restritiva nas subclasses, a modificação proposta é inválida. Ou seja, o intervalo de 0 a 1000 não está dentro do intervalo de 50 a 500.

(c) Para que serve o mecanismo de invariante de classe? Descreva as invariantes da classe “Pagamento”.

Resposta: O mecanismo de invariante de classe serve para garantir que as restrições de espaço-estado sejam respeitadas. As invariantes de classe devem ser válidas para todos os objetos em equilíbrio (sem método em execução) da classe. As invariantes da classe Pagamento são: (1) existir um pedido associado e (2) o atributo quantia ser maior do que zero.



Questão 4 (2 pontos)

Em relação às Heurísticas de Projeto,

(a) Explique o que são e para que servem as heurísticas de projeto.

Resposta: Heurísticas de projeto representam o resultado da explicitação do conhecimento de vários projetistas experientes, ou seja, orientações que apóiam projetistas na tomada de decisões na etapa de projeto de software. Entretanto, as heurísticas não pretendem substituir metodologias e sim auxiliá-las a gerar bons projetos de software – elas são úteis como dicas para projetistas inexperientes na maior parte dos casos, mas não em todos, pois pode haver conflitos mútuos entre algumas delas.

(b) Explique o raciocínio por trás da heurística "A maioria dos métodos de uma classe deve fazer uso da maioria dos atributos na maior parte do tempo".

Resposta: Essa heurística é importante para verificar se a coesão de uma classe está sendo mantida, o que impacta diretamente na existência de um bom projeto, que facilita o entendimento de suas entidades. Além disso, isso significa que a classe é estruturada de forma a não necessitar divisão.