

Conceitos Básicos de O.O

- 1) Defina os principais conceitos de orientação a objetos.
- 2) Enumere as diferenças entre programação orientada a objetos e programação estruturada.
- 3) Diferencie uma classe abstrata de uma interface.
- 4) Porque é desejável programar com foco em interfaces?

UML

- 1) Quando é apropriado modelar utilizando-se de composição? Por exemplo, por que não utilizar a notação de composição da UML para mostrar que um cachorro é composto de altura, peso, cor e data de nascimento?
- 2) Um pão fatiado é constituído de fatias de pão. A associação entre pão e suas fatias corresponde a uma composição ou uma agregação?
- 3) Esboce um diagrama de agregação de objetos para um capítulo de livro com a seguinte estrutura: um capítulo compreende diversas seções, cada uma das quais compreende diversos parágrafos e figuras. Um parágrafo compreende diversas sentenças, cada uma das quais compreende diversas palavras. (Você pode ignorar a pontuação e não precisa ir mais adiante em busca da estrutura de uma figura.)
- 4) Monte um diagrama de classes para um sistema usado para simplificar o horário e pontuação de competições atléticas julgadas, como ginástica artística, salto aquático, etc. Existem vários eventos e competidores. Cada competidor entra em vários eventos, e cada evento possui muitos competidores. Cada evento possui juízes, que avaliam subjetivamente o desempenho dos competidores nesse evento. Um juiz da nota a cada competidor do evento. Em algumas situações um juiz pode dar notas para mais de um evento.

- 5) Prepare um diagrama de classes para um editor de documentos gráficos que admita agrupamento. Suponha que um documento consista de várias páginas. Cada página contém objetos de desenho, incluindo texto, objetos geométricos e grupos. Um grupo é simplesmente um objeto de desenho, podendo incluir outros grupos. Um grupo precisa conter pelo menos dois objetos de desenho. Um objeto de desenho pode ser um membro direto de, no máximo, um grupo. Objetos geométricos incluem círculos, elipses, retângulos, linhas e quadrados.
- 6) Prepare um modelo de classes para descrever um grafo não direcionado.
- 7) Uma instituição pode emitir muitas contas de cartão de crédito, cada uma identificada por um número de conta. Cada conta possui um limite máximo de crédito, um saldo atual e um endereço de correspondência. A conta atende a um ou mais clientes, que residem no endereço de correspondência. A instituição emite periodicamente um extrato para cada conta. O extrato lista uma data de vencimento, encargos financeiros e pagamento mínimo. O extrato relaciona as diversas transações que ocorreram durante o intervalo de cobrança: saques em dinheiro, cobrança de juros, compras, taxas e ajustes de conta. O nome do comerciante é impresso para cada compra. Escreva um diagrama de classes que permite responder as seguintes perguntas:
- Que transações ocorreram para uma conta de cartão de crédito em um intervalo de tempo?
 - Que volume de transações foi realizado por uma instituição no ano passado?
 - Quais clientes fizeram compras com um comerciante no ano passado usando qualquer tipo de cartão de crédito?
 - Quantas contas de cartão de crédito um cliente tem atualmente?
 - Qual é o crédito total máximo de um cliente, considerando-se todas as suas contas?
- 8) Um usuário tem acesso a múltiplos arquivos, cada um acessível por um tipo de permissão de acesso. Descreva em UML o relacionamento entre as classes correspondentes.
- 9) Uma pessoa pode ter várias assinaturas de revista. Uma revista possui vários assinantes. Para cada assinatura é importante acompanhar a data e o valor de cada pagamento, além da data atual de expiração. Projete o diagrama de classes que descreva a assinatura de revistas.

- 2) Implemente uma lista tal como implementada na linguagem Lisp com as operações `cons`, `car` e `cdr` como uma extensão de uma `Collection<E>` em Java. (ver seção vii do manual de lisp em

<http://www.softwarepreservation.org/projects/LISP/book/LISP%201.5%20Programmers%20Manual.pdf>

- 3) Implemente uma árvore AVL como uma extensão de uma `Collection<E>`
- 4) Defina uma interface genérica para uma estrutura de grafo como uma `Collection<E>`.
- 5) Considere um conjunto documentos na forma de arquivos textos. Cada documento deve disponibilizar um modo de medir sua similaridade quando confrontado a outro de mesma natureza. Além disso, um documento é caracterizado por um perfil que serve para distinguir documentos diferentes. Existem inúmeras formas de se definir perfis. Uma possibilidade é usar a frequência de elementos no documento. Um exemplo de perfil é construído com base no conceito de frequência de *k-grams* onde um *k-gram* é uma substring de tamanho *k* na *string* que compõe o documento (entenda que o documento pode ser visto como uma única *string*). Para construir um histograma de *k-grams* é necessário associar um número a uma substring via *hashing*. Perfis podem ser comparados através de uma métrica que meça a distância entre os vetores unitários associados ao histograma definido sobre a frequência dos *k-grams*. Pode-se utilizar a métrica euclidiana ou um simples produto escalar. Escreva um programa que receba um conjunto de documentos e determine o grau de similaridade entre cada um deles.

Uma vez feita sua implementação reflita sobre os seguintes problemas:

Que mudanças podem ser necessárias caso:

- a) for preciso trabalhar com outros Documentos;
- b) for permitido usar diferentes formas de descrever os documentos (usar diferentes perfis);
- c) for preciso usar métricas diferentes para comparar as similaridades entre os perfis;
- d) for permitida diferentes visualizações dos resultados na interface gráfica;
- e) as configurações puderem ser feitas dinamicamente em tempo de execução.

Polimorfismo

- 1) Diferencie tipo e objetos de dado.
- 2) Defina um sistema de tipos.
- 3) Diferencia uma linguagem fortemente tipada de uma linguagem fracamente tipada.
- 4) Dê exemplos de linguagens fracamente tipadas e justifique sua afirmação.
- 5) Defina o conceito de compatibilidade de tipos. Quais as duas formas mais comuns de compatibilidade e como funcionam?

- 6) Defina os diferentes tipos de polimorfismo.

Princípios de Projeto O.O.

- 1) Explique os conceitos de fan-in, fan-out, coesão e acoplamento.
- 2) Explique os diferentes tipos de congneridade e cite exemplos.
- 3) Analise as funções amigáveis de C++ do ponto de vista de congneridade.
- 4) Considere as duas linhas de código abaixo. Cite os possíveis tipos de congneridade que podem ocorrer exemplificando:

```
int i;  
i = 7;
```

- 5) Considere um jogo do tipo *Text Adventure* onde você pode navegar por um mapa usando comandos norte, sul, leste e oeste. Suponha que no programa as direções associadas aos comandos forma armazenadas na forma de atributos de uma classe com os seguintes valores

norte = 0; leste = 1; sul = 2; oeste = 3;

Que tipo de congeneridade existe nesta situação.

- 6) Uma biblioteca de classes que você comprou de um distribuidor de classes gerais provavelmente conterá somente classes de base. Por que você acha que isso ocorre?
- 7) Defina os diferentes tipos de domínios em um sistema.
- 8) Porque classes não devem pertencer a mais de um domínio?
- 9) Determine a dimensionalidade do espaço de estados de um retângulo supondo que o mesmo seja capaz de girar, transladar e sofre escalas na altura e largura (Difícil).
- 10) Defina o conceito de invariante de classe. Cite um exemplo.