

Universidade Federal de Uberlândia
Sistemas de Informação – Campus Monte Carmelo
Prof. Eduardo Cunha Campos

Lista de Exercícios para a prova

1. Considerando os padrões de projetos (*design patterns*) da engenharia de *software* que utilizam as melhores práticas em orientação a objetos para atingir os resultados desejados, é correto afirmar que o padrão:

- a) *decorator* é utilizado para prover uma maneira de acessar os elementos de um objeto agregado sequencialmente sem expor sua representação interna.
- b) *bridge* é utilizado para desacoplar uma abstração de sua implementação para que os dois possam variar independentemente.
- c) *composite* é utilizado para oferecer uma interface única para um conjunto de interfaces de um subsistema, definindo uma interface de nível mais elevado que torna o subsistema mais fácil de usar.
- d) *memento* permite definir uma nova operação, sem mudar as classes dos elementos nos quais opera.
- e) *façade* é utilizado para compor objetos em estruturas de árvore, para representar hierarquias.

2. Qual é a finalidade de um sistema de controle de versão (SCV) durante o ciclo de desenvolvimento de *software*? Descreva 3 características fundamentais que todo SCV deve possuir.

3. Qual é a diferença entre um SCV centralizado e um SCV distribuído? Quando usar um ou outro no desenvolvimento de *software*? Cite exemplos utilizados no mundo real para essas 2 categorias de SCVs.

4. Em relação aos métodos e classes declarados como “final”, é correto afirmar que:

- a) um método *final* em uma superclasse pode ser sobrescrito em uma subclasse.
- b) os métodos que são declarados *static* são implicitamente final.
- c) uma classe que é declarada final pode ser uma superclasse.

- d) os métodos declarados *private* não são implicitamente final.
e) nenhum método em uma classe final é implicitamente final.

5. Em relação aos problemas, para os quais são definidas soluções, usando padrões de projeto (*design patterns*), correlacione as colunas a seguir:

Padrão	Problema/Solução
I. Adaptador (GoF adapter)	() <u>Problema</u> : Permitir apenas uma instância de uma classe. <u>Solução</u> : Definir um método estático que retorne o objeto.
II. Objeto unitário (GoF singleton)	() <u>Problema</u> : Usar uma interface comum e unificada para um conjunto não uniforme de implementações ou interfaces dentro de um subsistema. <u>Solução</u> : Definir um único ponto de contato através de um objeto com uma única interface unificada, responsável por colaborar com os componentes do subsistema.
III. Composto (GoF composite)	() <u>Problema</u> : Como tratar um grupo de objetos (polimorficamente), da mesma forma que um objeto atômico? <u>Solução</u> : Definir classes para os grupos e para os objetos atômicos para que eles implementem a mesma interface.
IV. Fachada (GoF facade)	() <u>Problema</u> : Como resolver o problema de interfaces incompatíveis? <u>Solução</u> : Converter a interface original de um componente em outra interface usando um objeto intermediário.

Está **CORRETA** a seguinte sequência de respostas, de cima para baixo:

- a) II, IV, III, I.
- b) IV, II, I, III.
- c) I, II, III, IV.
- d) II, IV, I, III.
- e) II, I, IV, III.

6. Para facilitar a manutenção da aplicação, há um *design pattern* que tem como objetivo principal centralizar o acesso aos dados em uma única camada. Esse *design pattern* é o

- a) DTO.
- b) Business Object.
- c) DAO.
- d) Application Service.
- e) MVC.

7. Em relação a padrões de projeto de *software*, assinale a afirmativa **incorreta**.

- a) *Builder* é um padrão utilizado quando se deseja separar a construção de um objeto complexo de sua representação de modo que o mesmo processo de construção possa criar diferentes representações.
- b) *Factory Method* é um padrão utilizado quando se deseja definir uma interface para criar um objeto e deixar as subclasses decidirem que classe instanciar.
- c) *Adapter* é um padrão utilizado quando se deseja converter a interface de uma classe em outra interface, esperada pelos clientes.
- d) *Singleton* é um padrão utilizado quando se deseja compor objetos em estrutura de árvore para representarem hierarquias partes-todo.
- e) *Proxy* é um padrão também conhecido como surrogate utilizado quando se deseja fornecer um substituto ou marcador da localização de outro objeto para controlar o acesso ao mesmo.

8. De acordo com o livro: “Padrões de Projeto: soluções reutilizáveis de software orientado a objetos”, os padrões “GoF” são divididos em 24 tipos. Em função dessa grande quantidade de padrões, foi necessário classificá-los de acordo com as suas finalidades. São 3 categorias: padrões de criação, padrões estruturais e padrões comportamentais. Descreva um exemplo de padrão de projeto para cada uma dessas categorias.

9. O que são Idiomas de Programação? Cite e explique este conceito através de exemplos de escrita de código.

10. Na linguagem Java, são modificadores para controle de acesso às variáveis e aos métodos de uma classe:

- a) *public* e *private*.
- b) *private* e *static*.
- c) *static* e *public*.
- d) *final* e *static*.
- e) *protected* e *static*.