BSI FACOM UFU

Programação Orientada a Objetos 2

Lista de Exercícios 1

Propriedades e Princípios de Projeto

Padrões de Projeto: Singleton, Fábrica e Adaptador

Profa. Fabíola S. F. Pereira

Propriedades e Princípios de Projeto

QUESTÃO 1 Descreva três benefícios da propriedade de projeto chamada ocultamento de informação (information hiding)?

- Desenvolvimento em paralelo. Suponha que um sistema X foi implementado por meio de classes C1, C2, …, Cn. Quando essas classes ocultam suas principais informações, fica mais fácil implementá-las em paralelo, por desenvolvedores diferentes. Consequentemente, teremos uma redução no tempo total de implementação do sistema.

- Flexibilidade a mudanças. Por exemplo, suponha que descobrimos que a classe Ci é responsável pelos problemas de desempenho do sistema. Quando detalhes de implementação de Ci são ocultados do resto do sistema, fica mais fácil trocar sua implementação por uma classe Ci', que use estruturas de dados e algoritmos mais eficientes. Essa troca também é mais segura, pois como as classes são independentes, diminui-se o risco de a mudança introduzir bugs em outras classes.

- Facilidade de entendimento. Por exemplo, um novo desenvolvedor contratado pela empresa pode ser alocado para trabalhar em algumas classes apenas. Portanto, ele não precisará entender toda a complexidade do sistema, mas apenas a implementação das classes pelas quais ficou responsável.

QUESTÃO 2 Suponha que um programador adote a seguinte estratégia: ao implementar qualquer nova funcionalidade ou corrigir um bug que implique na modificação de duas classes A e B localizadas em arquivos diferentes, ele conclui a tarefa movendo as classes para o mesmo arquivo. Explicando melhor: após terminar a tarefa de programação que ficou sob sua responsabilidade, ele escolhe uma das classes, digamos a classe B, e a move para o mesmo arquivo da classe A. Agindo dessa maneira, ele estará melhorando qual propriedade de projeto? Por outro lado, qual propriedade de projeto estará sendo afetada de modo negativo? Justifique.
Slides aula de revisão.

QUESTÃO 3 É possível que uma classe A esteja acoplada a uma classe B sem ter uma referência para B em seu código? Se sim, esse acoplamento será aceitável ou será um acoplamento ruim?

Sim, basta criar um acoplamento de dados, que é um acoplamento ruim. Por exemplo, uma classe B grava um valor em um arquivo, que a classe A depois vai ler.

QUESTÃO 4 Suponha um programa em que todo o código está implementado no método main. Ele tem um problema de coesão ou acoplamento? Justifique.

Coesão, certamente é um método enorme, que faz "tudo" no sistema. Logo, ele não pode ser coeso.

QUESTÃO 5 Qual princípio de projeto é violado pelo seguinte código?

```
void onclick() {
  num1 = textfield1.value();
  c1 = BD.getConta(num1)
  num2 = textfield2.value();
  c2 = BD.getConta(num2)
  valor = textfield3.value();
  beginTransaction();
  try {
    c1.retira(valor);
    c2.deposita(valor);
    commit();
  }
  catch() {
    rollback();
  }
}
```

Princípio da Responsabilidade Única, pois esse método tem duas responsabilidades: ele trata eventos de interface e executa uma operação de negócio (transferência bancária). São responsabilidades diferentes e, portanto, deveriam estar em métodos distintos. Da forma como está implementado, não é possível reusar o código de negócio (transferência bancária) com um outro tipo de interface. Para concluir, podemos dizer que esse método não é coeso.

QUESTÃO 6 Costuma-se afirmar que existem três conceitos chaves em orientação a objetos: encapsulamento, polimorfismo e herança. Suponha que você tenha sido encarregado de projetar uma nova linguagem de programação. Suponha ainda que você poderá escolher apenas dois dos três conceitos que mencionamos. Qual dos conceitos eliminaria então da sua nova linguagem? Justifique sua resposta.
Slides aula de revisão.

QUESTÃO 7 Qual princípio de projeto é violado pelo seguinte código? Como você poderia alterar o código do método para atender a esse princípio?

```
void sendMail(ContaBancaria conta, String msg) {
  Cliente cliente = conta.getCliente();
  String mail = cliente.getMailAddress();
```

```
        "Envia mail"
      }
```

Viola o Princípio de Demeter, que recomenda que a implementação de um método deve invocar apenas os seguintes outros métodos: de sua própria classe, de objetos passados como parâmetros, de objetos criados pelo próprio método e de atributos da classe do método.

QUESTÃO 8 Qual princípio de projeto é violado pelo seguinte código? Como você poderia alterar o código do método para atender a esse princípio?

```
void imprimeDataContratacao(Funcionario func) {
  Date data = func.getDataContratacao();
  String msg = data.format();
  System.out.println(msg);
}
```

Idem exercício anterior: viola Demeter

QUESTÃO 9 Para cada assertiva a seguir, assinale verdadeiro ou falso. **Quando falso, justifique.**

*neste gabarito, faltam as justificativas!

a. (V) Desenvolvimento em paralelo, isto é, a possibilidade de desenvolver ao mesmo tempo diversas classes de um sistema, é uma das vantagens de ocultamento de informação.

b. (F) Getters e setters são essenciais para implementação de classes com ocultamento de informação.

c. (F) Coesão e separação de interesses são conceitos antagônicos.

d. (V) Acoplamento estrutural ocorre quando uma classe A referencia em sua implementação uma classe B. Por exemplo, a classe A declara um atributo ou parâmetro do tipo B ou cria um objeto do tipo B.

e. (V) Classes grandes tendem a ter problemas de coesão (baixa) e acoplamento (alto).

f. (F) Todos os atributos de uma determinada classe X são privados. Logo, com certeza, X oferece Ocultamento de Informação.

QUESTÃO 10 Para cada assertiva a seguir, assinale verdadeiro ou falso. **Quando falso, justifique.**
*neste gabarito, faltam as justificativas!

a. (V) Segregação de Variáveis e Responsabilidade Única são princípios úteis para obter classes coesas.

b. (V) O Princípio de Substituição de Liskov aplica-se somente a linguagens com herança.

c. (F) Um nome mais intuitivo para Inversão de Dependências é Prefira Composição a Herança, como Java e C++.

d. (V) Herança de classes envolve reúso de código, pois subclasses não precisam reimplementar os métodos herdados da classe pai.

e. (F) Composição é um mecanismo de reúso caixa-branca, pois a classe cliente tem acesso a diversos detalhes de implementação da classe servidora.

f. (V) Uma chamada obj.getX.getY().getZ().metodo() representa uma violação do Princípio de Demeter.

QUESTÃO 11 Para cada assertiva a seguir, assinale verdadeiro ou falso. **Quando falso, justifique.**

*neste gabarito, faltam as justificativas!

a. (F) De acordo com o Princípio Aberto/Fechado, uma classe deve ser aberta para modificações, mas fechada para extensões.

b. (V) Existe uma relação de composição entre duas classes A e B quando a classe A possui um atributo do tipo B.

c. (F) O Princípio de Liskov recomenda que não devemos usar objetos apenas como intermediários para chegar até um objeto de interesse.

d. (V) Coesão vale para qualquer unidade de código, seja ela um método, uma classe ou um pacote.

e. (V) Em um módulo, uma interface simples é mais importante do que uma implementação simples.

f. (V) Suponha uma classe de domínio X com grupos de métodos M1, M2 e M3, os quais são usados, respectivamente, pelos sistemas contábil, financeiro e de compras de uma empresa. Esse fato é um indicativo de que X viola o Princípio da Responsabilidade Única.

g. (V) 'Não fale com estranhos' é uma frase que resume o Princípio de Demeter.

h. (F) Se uma classe tem mais de um método, ela viola o Princípio da Responsabilidade Única.

Padrões de Projeto

QUESTÃO 12 Para cada assertiva a seguir, assinale verdadeiro ou falso. **Quando falso, justifique.**

*neste gabarito, faltam as justificativas!

a. (F) Singleton tem por objetivo garantir que uma classe tenha ao menos uma instância e fornecer um ponto global de acesso para ela.
b. (V) Singleton é um padrão que garante que uma classe possui, no máximo, uma instância e oferece um ponto único de acesso a ela.
c. (V) Fábrica Abstrata é um padrão que oferece uma interface ou classe abstrata para criação de uma família de objetos relacionados.
d. (F) Wrapper é um padrão que facilita a construção de objetos complexos com vários atributos, sendo alguns deles opcionais.
e. (V) Um adaptador oferece uma interface diferente para o objeto adaptado. Em contraste, um proxy possui a mesma interface que o seu objeto base.
f. (F) Como são padrões distintos, uma Fábrica não pode ser implementada como um Singleton.

QUESTÃO 13 Dê o nome dos seguintes padrões de projeto:

a. Garante que uma classe possui, no máximo, uma instância e oferece um ponto único de acesso a ela.
   Singleton
b. Oferece uma interface ou classe abstrata para criação de uma família de objetos relacionados
   Fábrica
c. Oferece um método para centralizar a criação de um tipo de objeto.
   Fábrica
d. Converte a interface de uma classe para outra interface esperada pelos clientes.
   Wrapper

QUESTÃO 14 Sobre o padrão Singleton: (a) Na implementação desse padrão, como se evitam que sejam criadas instâncias do singleton fora da sua classe? Em outras palavras, suponha que X seja um Singleton; como se proíbe que fora da classe X sejam realizadas operações do tipo new X()? (b) Por que Singleton é um dos padrões de projeto mais criticados?

(a) Criar na classe singleton uma construtora privada e vazia. (b) Porque ele pode ser usado para criar estruturas de dados globais.