PUC Minas

Quizz 4

INFORMAÇÕES DOCENTE						
CURSO: ENGENHARIA DE SOFTWARE	**DISCIPLINA:** PROJETO DE SOFTWARE	**TURNO**	MANHÃ	TARDE	NOITE	**PERÍODO/SALA:** 4º
			x		x	
PROFESSOR (A): João Paulo Carneiro Aramuni						

Padrões GRASP – General Responsability Assignment Software Patterns

Padrão GRASP Especialista

Questão 1)

```
class Pedido {
   private List<Item> itens;

   public double calcularTotal() {
      double total = 0;
      for (Item item : itens) {
         total += item.getPreco() * item.getQuantidade();
      }
      return total;
   }
}
```

Por que o método `calcularTotal` foi implementado na classe `Pedido`?
A) Porque a classe `Pedido` é especialista na soma de itens.
B) Para seguir o padrão GRASP Criador.
C) Para manter alta coesão no sistema.
D) Porque a classe `Item` não pode acessar o preço diretamente.
E) Para reduzir o acoplamento entre classes.

Questão 2)

```
class Funcionario {
   private double salarioBase;
   private double bonus;

   public double calcularSalarioTotal() {
      return salarioBase + bonus;
   }
}
```

Por que o cálculo do salário total foi atribuído à classe `Funcionario`?
A) Para seguir o padrão GRASP Acoplamento Fraco.
B) Porque a classe `Funcionario` é especialista em seus próprios dados.
C) Para encapsular a lógica em uma classe independente.
D) Para melhorar a reutilização do código.
E) Para simplificar a lógica do sistema.

Padrão GRASP Criador

Questão 3)

```
class Pedido {
   private List<Item> itens;

   public void adicionarItem(String nome, double preco) {
      Item item = new Item(nome, preco);
      itens.add(item);
   }
}
```

Por que a classe `Pedido` cria instâncias de `Item`?
A) Para aumentar a coesão da classe.
B) Para seguir o padrão GRASP Criador.
C) Porque a classe `Item` depende do `Pedido`.
D) Para encapsular a lógica de criação em uma única classe.
E) Para simplificar a lógica de negócio.

Questão 4)

```
class Carrinho {
   private List<Produto> produtos;

   public Produto criarProduto(String nome, double preco) {
      return new Produto(nome, preco);
   }
}
```

Qual é o motivo para a classe `Carrinho` ser responsável pela criação de `Produto`?
A) Porque `Carrinho` usa diretamente os objetos da classe `Produto`.
B) Para reduzir o acoplamento entre as classes.
C) Para implementar o padrão GRASP Controlador.

D) Para evitar que `Produto` precise conhecer os detalhes do carrinho.
E) Porque `Carrinho` segue o padrão GRASP Criador.

Padrão GRASP Coesão Alta

Questão 5)

```
class Relatorio {
   private List<Dado> dados;

   public void gerarRelatorio() {
      for (Dado dado : dados) {
         System.out.println(dado.getInfo());
      }
   }
}
```

Por que o método `gerarRelatorio` está na classe `Relatorio`?
A) Para manter alta coesão.
B) Para evitar acoplamento fraco.
C) Para permitir que outras classes gerem relatórios.
D) Para centralizar a lógica em um único local.
E) Para facilitar a reutilização de código.

Questão 6)

```
class Calculadora {
   public double somar(double a, double b) {
      return a + b;
   }
}
```

Por que o método `somar` está na classe `Calculadora`?
A) Porque a classe é especialista em cálculos.
B) Para reduzir o acoplamento entre módulos.
C) Para seguir o padrão GRASP Controlador.
D) Para garantir alta coesão na classe.
E) Para centralizar a lógica em uma única classe.

Padrão GRASP Acoplamento Fraco

Questão 7)

```
class Cliente {
   private Pedido pedido;

   public double obterValorTotal() {
      return pedido.calcularTotal();
   }
}
```

Por que o método `obterValorTotal` em `Cliente` delega o cálculo ao `Pedido`?
A) Para reduzir o acoplamento entre `Cliente` e os itens.
B) Para seguir o padrão GRASP Criador.
C) Para aumentar a coesão da classe `Cliente`.
D) Para encapsular a lógica do cálculo no `Cliente`.
E) Para evitar duplicação de código.

Questão 8)

```
class Pagamento {
   private ServicoPagamento servico;

   public boolean processarPagamento(double valor) {
      return servico.autorizar(valor);
   }
}
```

Por que `Pagamento` não implementa diretamente a lógica de autorização?
A) Para aumentar a coesão da classe `Pagamento`.
B) Para seguir o padrão GRASP Acoplamento Fraco.
C) Para permitir que `ServicoPagamento` seja substituído sem impactar `Pagamento`.
D) Para manter o princípio de responsabilidade única.
E) Para simplificar o teste da classe `Pagamento`.

Padrão GRASP Controlador

Questão 9)

```
class ControladorUsuario {
   private ServicoUsuario servico;

   public boolean autenticar(String login, String senha) {
      return servico.autenticarUsuario(login, senha);
   }
}
```

Por que a responsabilidade de autenticar foi atribuída a `ControladorUsuario`?
A) Para seguir o padrão GRASP Criador.
B) Para centralizar a lógica de autenticação no controlador.
C) Para implementar o padrão GRASP Controlador.
D) Para manter o acoplamento fraco entre camadas.
E) Para encapsular os detalhes de autenticação.

Questão 10)

```
class ControladorVenda {
   private ServicoVenda servico;

   public boolean processarVenda(int idProduto, int quantidade) {
      return servico.registrarVenda(idProduto, quantidade);
   }
}
```

Por que a responsabilidade de processar a venda foi delegada ao `ControladorVenda`?
A) Para evitar lógica de negócios na camada de interface.
B) Para garantir alta coesão no serviço.
C) Para aplicar o padrão GRASP Controlador.
D) Para encapsular os detalhes de registro da venda.
E) Para reduzir a duplicação de código entre as classes.

PUC Minas

Gabarito:

1) A
2) B
3) B
4) E
5) A
6) D
7) A
8) B
9) C
10) C