Questões

# Questão 1

Qual é a principal vantagem do padrão Chain of Responsibility?

* Reduz dependências diretas entre objetos
* Permite que um pedido seja processado por múltiplos handlers(x)
* Simplifica a criação de objetos complexos
* Facilita a implementação de coleções iteráveis

# Questão 2

Qual padrão de projeto é utilizado para encapsular um pedido como um objeto, permitindo que seja parametrizado com diferentes requisições?

* Chain of Responsibility
* Command(x)
* Iterator
* Mediator

# Questão 3

O que o padrão Iterator permite fazer?

* Permite a passagem de um pedido por uma corrente de handlers
* Reduz dependências caóticas entre objetos
* Percorrer elementos de uma coleção sem expor sua estrutura(x)
* Salvar e restaurar o estado anterior de um objeto

# Questão 4

Qual padrão de projeto restringe comunicações diretas entre objetos e os força a colaborar apenas através de um objeto mediador?

* Observer
* State
* Mediator(x)
* Visitor

# Questão 5

Qual padrão de projeto permite que você salve e restaure o estado anterior de um objeto sem revelar os detalhes de sua implementação?

* Memento(x)
* Command
* Iterator
* State

# Questão 6

No padrão Observer, o que é o papel do 'observado'?

* Enviar pedidos para handlers
* Alterar seu comportamento conforme seu estado interno
* Notificar múltiplos objetos sobre quaisquer eventos que aconteçam com ele(x)
* Medir a comunicação entre diferentes objetos

# Questão 7

Qual padrão de projeto permite que um objeto altere seu comportamento quando seu estado interno muda?

* Strategy
* Template Method
* State(x)
* Visitor

# Questão 8

Qual é a principal funcionalidade do padrão Strategy?

* Permitir que um objeto altere seu comportamento conforme seu estado interno
* Definir uma família de algoritmos, colocando-os em classes separadas(x)
* Estabelecer conexões unidirecionais entre remetentes e destinatários
* Reduzir as dependências caóticas entre objetos

# Questão 9

O que o padrão Template Method define?

* Uma interface para criar uma família de algoritmos
* O esqueleto de um algoritmo na superclasse, permitindo que subclasses sobrescrevam etapas específicas(x)
* Um objeto independente que contém toda a informação sobre um pedido
* Um mecanismo de assinatura para notificar múltiplos objetos sobre eventos

# Questão 10

Qual padrão de projeto permite separar algoritmos dos objetos nos quais eles operam?

* Chain of Responsibility
* Command
* Template Method
* Visitor(x)

# Questão 11

Explique o funcionamento do padrão Chain of Responsibility e forneça um exemplo de aplicação prática.

Resposta:

Com esse padrão comportamental é possivel passer a responsabildiades para outros objetos(handler), e cada handler tem a possibilidades de encerrar uma determinada solicitação ou passer adiante.

Um exemplo pratico seria, quando a sua tia pede para você levar o lixo para fora, com essa ‘solicitacao’ passada a você, você pede para a sua vó levar o lixo para fora.

# Questão 12

Descreva o padrão Command e explique como ele pode ser utilizado para implementar um sistema de desfazer/refazer (undo/redo).

Com ese padrão comportamental é possivel transformar uma solicitação em um objeto independente que armazena toda a informação daquela solicitação.

Para implementar essas funcionalidades será necessário armazenar em pilha os historicos das operações, com essas informações armazenadas é possivel realizer o desfazer/refazer

# Questão 13

Com o padrão Iterator melhora a flexibilidade no acesso a coleções? Forneça um exemplo.

Com esse padrão comportamental é possivel interar sobre os objetos de forma paralela, já que cada objeto tem seu propio estado.

Um exemplo seria a renderizacao dos forms em html com um iterator

# Questão 14

Explique o papel do Mediator em sistemas complexos e como ele pode ser utilizado para reduzir o acoplamento entre componentes.

Com esse padrão comportamental ele é utilizado como um ‘centralizador’ de chamadas dos componentes, possibilitando que o Sistema tenha somente 1 controle, e não varios.

# Questão 15

Qual é a principal vantagem do padrão Memento em termos de encapsulamento e proteção do estado interno de um objeto?

Com esse padrão comportamental é possivel criar novas classes sem ter que alterar o codigo da ‘publicadora’

# Questão 16

Descreva um cenário em que o padrão Observer seria a melhor escolha para implementar a funcionalidade desejada.

Um Sistema de publicação de noticias seria um otimo cenario para esse padrão comportamental

# Questão 17

Como o padrão State facilita a implementação de máquinas de estado? Forneça um exemplo de sua aplicação.

Um exemplo para essa aplicação seria um script para escolher um tipo de refrigerante emu ma maquina de refrigerante. Dá para separar em estados nesse script, se a maquina está com Moedas, ou não, sem tem refrigerante ou não, se vendeu um regri ou não. Sendo possivel extrair esses estados de dentro das classes facilitando a implementação

# Questão 18

Explique como o padrão Strategy pode ser utilizado para escolher algoritmos dinamicamente em tempo de execução.

O padrão Strategy permite que você defina uma família de algoritmos, coloque cada um deles em uma classe separada e torne esses algoritmos intercambiáveis.

# Questão 19

Discuta a diferença entre os padrões Strategy e Template Method, destacando suas aplicações práticas.

Ambos resolvem praticamente o mesmo problema, são apenas maneiras diferentes de atingir o mesmo objetivo. Template method atinge o objetivo usando herança, enquanto Strategy podemos dizer que é por delegação/composição.

# Questão 20

Como o padrão Visitor permite adicionar operações a objetos sem modificar suas classes? Dê um exemplo.

Com esse padrão comportamental é possivel separar os objetos dos algoritmos sem modificar suas classes.

Um exemplo seria um script de utilizar/acessar um tipo de documento dependendo do seu ‘visitor’

# Questão 21

Compare os padrões Chain of Responsibility e Command em termos de estrutura e propósito. Quando seria mais apropriado usar um em vez do outro?

**Chain of Responsibility:** Use quando a decisão sobre quem deve processar uma solicitação pode ser feita dinamicamente e de forma encadeada entre vários objetos.

**Command:** Use quando deseja encapsular uma solicitação como um objeto, com suporte a operações como desfazer ou reverter, ou quando deseja parametrizar objetos com operações para execução posterior.