Padrões de Projeto CESGRANRIO

Prof. Lúcio Camilo
Email: luciocamilo@gmail.com
http://www.itnerante.com.br/profile/LucioCamilo

http://www.provasdeti.com.br/

Lúcio Camilo

- Resumo – CV
- Analista de Sistemas do Banco do Brasil – Cedido à PREVI
- Especialidade: Administrador de Servidor de Aplicações
- Pós Graduado em Engenharia de Software
- Autor do Livro “Android para Desenvolvedores”, Editora Brasport
- MBA Gerenciamento de Projetos
- Certificações Profissionais:
  - SCJP, OCWD, OCJA Part I
  - RHSA, Big IP Essentials e Advanced

Contatos:

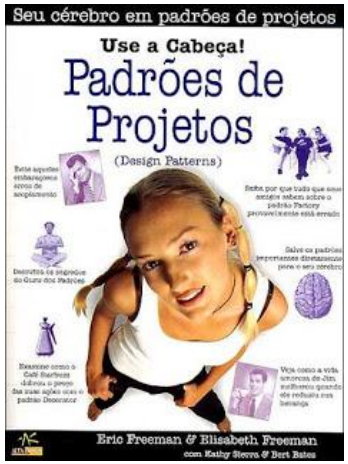
- luciocamilo@gmail.com
- www.itnerante.com.br/profile/luciocamilo

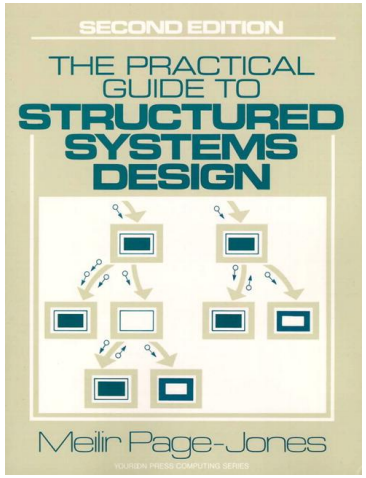
Conteúdo do Curso:

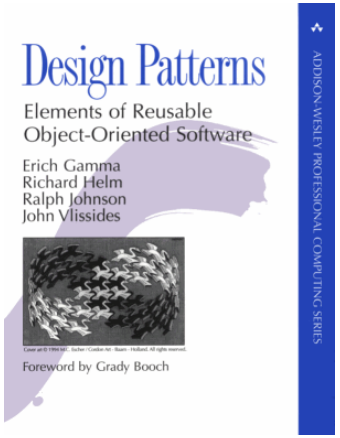
- Conceitos de OO;
- Padrões de Projeto GOF;
- Coesão e Acoplamento;
- Refatoração;
- Padrões Arquiteturais Empresariais;

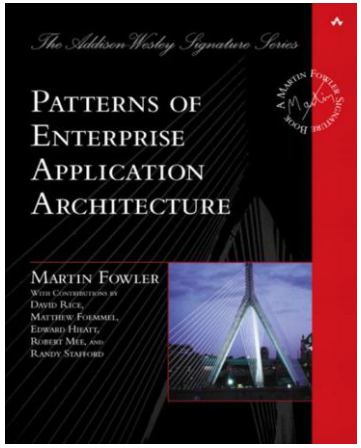
http://www.provasdeti.com.br/

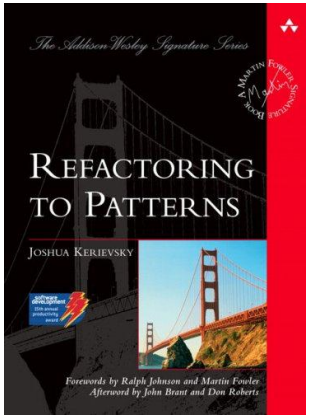
Bibliografia

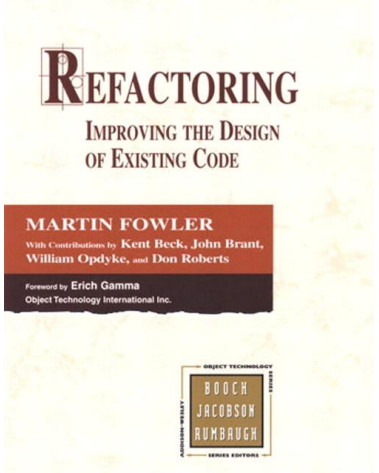












http://martinfowler.com/eaaCatalog/

http://sourcemaking.com/refactoring

http://www.refactoring.com/catalog/

Resumo GoF (1/3)

Adapter	Converter a interface de uma classe em outra interface esperada pelos clientes.
Façade	Oferecer uma interface única de nível mais elevado para um conjunto de interfaces de um subsistema.
Composite	Permitir o tratamento de objetos individuais e composições hierárquicas desses objetos de maneira uniforme.
Bridge	Desacoplar uma abstração de sua implementação, de tal forma que os dois possam variar independentemente
Singleton	Garantir que uma classe só tenha uma única instância, e prover um ponto de acesso global a ela.
Observer	Definir dependência um-para-muitos entre objetos para que, quando um objeto mudar de estado, os seus dependentes sejam notificados e atualizados.
Mediator	Definir um objeto que encapsula a forma como um conjunto de objetos interage
Proxy:	Prover um substituto ou ponto através do qual um objeto possa controlar o acesso a outro.
Chain of Responsibility	Compor objetos em cascata para, através dela, delegar uma requisição até que um objeto a sirva.

Resumo GoF(2/3)

Flyweight	Usar compartilhamento para suportar eficientemente grandes quantidades de objetos complexos.
Builder	Separar a construção de objeto complexo da representação para criar representações diferentes com mesmo processo.
FactoryMethod	Definir uma interface para criar um objeto mas deixar que subclasses decidam que classe instanciar.
Abstract Factory	Prover interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas
Prototype	Especificar tipos a criar usando uma instância como protótipo e criar novos objetos ao copiar este protótipo.
Memento	Externalizaro estado interno de um objeto para que o objeto possa ter esse estado restaurado posteriormente.
TemplateMethod	Definir o esqueleto de um algoritmo dentro de uma operação, deixando alguns passos a serem preenchidos pelas subclasses.
State	Permitir a um objeto alterar o seu comportamento quando o seu estado interno mudar.
Strategy	Definir uma família de algoritmos, encapsular cada um, e fazê-los intercambiáveis.

Resumo GoF(3/3)

Command	Encapsular uma requisição como objeto, para parametrizarclientes com diferentes requisições, filas e dar suporte a ações reversíveis.
Interpreter	Dada uma linguagem, definir uma representação para sua gramática junto com um interpretador.
Decorator	Anexar responsabilidades adicionais a um objeto dinamicamente.
Iterator	Prover acesso sequencial a elementos de um objeto agregado, sem expor sua representação interna.
Visitor	Representar uma operação a ser realizada sobre os elementos de uma estrutura de objetos.

Conceitos de OO

CESGRANRIO

FUNDAÇÃO
CESGRANRIO

01 – CESGRANRIO – EPE - 2007

Numa linguagem de programação orientada a objetos é importante restringir a visibilidade de alguns atributos para garantir o conceito de:

a) classe.

b) encapsulamento.

c) herança.

d) instanciação.

e) polimorfismo.

01 – CESGRANRIO – EPE - 2007

Numa linguagem de programação orientada a objetos é importante restringir a visibilidade de alguns atributos para garantir o conceito de:

a) classe.

b) encapsulamento.

c) herança.

d) instanciação.

e) polimorfismo.

02 – CESGRANRIO – EPE - 2007

Que característica **NÃO** é fundamental em uma linguagem de programação orientada a objeto?

a) Criação de classes.

b) Encapsulamento.

c) Herança múltipla.

d) Herança simples.

e) Instanciação de objetos.

02 – CESGRANRIO – EPE - 2007

Que característica **NÃO** é fundamental em uma linguagem de programação orientada a objeto?

a) Criação de classes.

b) Encapsulamento.

c) Herança múltipla.

d) Herança simples.

e) Instanciação de objetos.

03 – CESGRANRIO – ELETROBRÁS - 2010

Em determinada reunião de arquitetura de um sistema de cadastro de clientes de uma empresa, um analista aponta que, no âmbito de Orientação a Objetos, a herança múltipla

a) é caracterizada pelo comportamento polimórfico de objetos.

b) é caracterizada pela separação de aspectos internos e externos de um objeto.

c) ocorre quando uma classe deriva, por herança, mais de uma subclasse.

d) ocorre quando uma classe herda métodos ou atributos de mais de uma superclasse.

e) ocorre quando uma classe herda mais de um método de uma superclasse.

03 – CESGRANRIO – ELETROBRÁS - 2010

Em determinada reunião de arquitetura de um sistema de cadastro de clientes de uma empresa, um analista aponta que, no âmbito de Orientação a Objetos, a herança múltipla

a) é caracterizada pelo comportamento polimórfico de objetos.

b) é caracterizada pela separação de aspectos internos e externos de um objeto.

c) ocorre quando uma classe deriva, por herança, mais de uma subclasse.

d) ocorre quando uma classe herda métodos ou atributos de mais de uma superclasse.

e) ocorre quando uma classe herda mais de um método de uma superclasse.

04 – CESGRANRIO – DNPM - 2006

Em programação orientada a objetos, é correto afirmar que herança múltipla:

a) é a instância de uma classe abstrata.

b) define no máximo uma classe pai.

c) permite que uma classe herde atributos e métodos de duas ou mais classes.

d) ocorre quando uma classe é a instância de vários objetos.

e) significa o mesmo que polimorfismo.

04 – CESGRANRIO – DNPM - 2006

Em programação orientada a objetos, é correto afirmar que herança múltipla:

a) é a instância de uma classe abstrata.

b) define no máximo uma classe pai.

c) permite que uma classe herde atributos e métodos de duas ou mais classes.

d) ocorre quando uma classe é a instância de vários objetos.

e) significa o mesmo que polimorfismo.

05 – CESGRANRIO – EPE - 2006

Relacione o conceito de Orientação a Objetos à sua respectiva descrição.

Conceito

I - Agregação
II - Coesão
III - Encapsulamento
IV - Herança

Descrição

(P) É uma medida que indica até que ponto as partes de um objeto admitem a finalidade única do objeto.
(Q) Tipo de associação que indica que um dos objetos participantes precisa desempenhar o papel de controlador do outro objeto.
(R) Relacionamento entre classes que permite a uma classe especializada (subclasse) ter acesso às características de uma classe generalizada (superclasse).

A relação correta é:

a) I - P , II - Q , III - R

b) I - Q , II - R , III - P

c) I - Q , II - P , IV - R

d) II- R , III- P , IV- Q

e) II- R , III- Q , IV- P

05 – CESGRANRIO – EPE - 2006

Relacione o conceito de Orientação a Objetos à sua respectiva descrição.

Conceito

I - Agregação
II - Coesão
III - Encapsulamento
IV - Herança

Descrição

(P) É uma medida que indica até que ponto as partes de um objeto admitem a finalidade única do objeto.
(Q) Tipo de associação que indica que um dos objetos participantes precisa desempenhar o papel de controlador do outro objeto.
(R) Relacionamento entre classes que permite a uma classe especializada (subclasse) ter acesso às características de uma classe generalizada (superclasse).

A relação correta é:

a) I - P , II - Q , III - R

b) I - Q , II - R , III - P

c) I - Q , II - P , IV - R

d) II- R , III- P , IV- Q

e) II- R , III- Q , IV- P

06 – CESGRANRIO – CEF - 2008

Na POO (Programação Orientada a Objetos), qual o princípio em que duas ou mais classes, derivadas de uma mesma superclasse, podem invocar métodos que têm a mesma identificação (assinatura), mas comportamentos distintos, especializados para cada classe derivada?

a) Herança múltipla

b) Encapsulamento

c) Polimorfismo

d) Abstração

e) Reuso

06 – CESGRANRIO – CEF - 2008

Na POO (Programação Orientada a Objetos), qual o princípio em que duas ou mais classes, derivadas de uma mesma superclasse, podem invocar métodos que têm a mesma identificação (assinatura), mas comportamentos distintos, especializados para cada classe derivada?

a) Herança múltipla

b) Encapsulamento


c) Polimorfismo

d) Abstração

e) Reuso

07 – CESGRANRIO – FUNASA - 2009

Em programação orientada a objetos, os conceitos de especialização e generalização estão diretamente relacionados ao uso de :

a) Herança

b) Encapsulamento

c) Agregação

d) Composição

e) Associações Ternárias

07 – CESGRANRIO – FUNASA - 2009

Em programação orientada a objetos, os conceitos de especialização e generalização estão diretamente relacionados ao uso de :

a) Herança

b) Encapsulamento

c) Agregação

d) Composição

e) Associações Ternárias

08 – CESGRANRIO – EPE - 2012

Na programação orientada a objeto, na linguagem C# em particular, a capacidade de construir vários métodos com um mesmo nome, porém com parâmetros diferentes na mesma classe, é chamada de:

a) Polimorfismo universal

b) Polimorfismo paramétrico

c) Polimorfismo de subtipo

d) Sobrecarga de operadores

e) Sobrecarga de métodos

08 – CESGRANRIO – EPE - 2012

Na programação orientada a objeto, na linguagem C# em particular, a capacidade de construir vários métodos com um mesmo nome, porém com parâmetros diferentes na mesma classe, é chamada de:

a) Polimorfismo universal

b) Polimorfismo paramétrico

c) Polimorfismo de subtipo

d) Sobrecarga de operadores

e) Sobrecarga de métodos

09 – CESGRANRIO – FUNASA - 2009

Projetos orientados a objetos em geral têm como uma de suas características a componentização, cuja principal justificativa é alcançar um alto índice de:

a) reuso

b) complexidade

c) simplicidade

d) qualidade

e) polimorfismo

09 – CESGRANRIO – FUNASA - 2009

Projetos orientados a objetos em geral têm como uma de suas características a componentização, cuja principal justificativa é alcançar um alto índice de:

a) reuso

b) complexidade

c) simplicidade

d) qualidade

e) polimorfismo

10 – CESGRANRIO – PETROBRÁS - 2011

Em um projeto de software orientado a objetos, surgiu a necessidade de modelar um certo comportamento alternativo com base no tipo específico de uma determinada entidade. Procedimento similar foi desenvolvido no passado, usando lógica condicional através dos comandos se - então - senão em uma linguagem de programação estruturada.

Qual recurso o programador deverá utilizar para solucionar a questão nesse novo projeto?

a) Agregação

b) Classes Abstratas

c) Encapsulamento

d) Polimorfismo

e) Composição

10 – CESGRANRIO – PETROBRÁS - 2011

Em um projeto de software orientado a objetos, surgiu a necessidade de modelar um certo comportamento alternativo com base no tipo específico de uma determinada entidade. Procedimento similar foi desenvolvido no passado, usando lógica condicional através dos comandos se - então - senão em uma linguagem de programação estruturada.

Qual recurso o programador deverá utilizar para solucionar a questão nesse novo projeto?

a) Agregação

b) Classes Abstratas

c) Encapsulamento

d) Polimorfismo

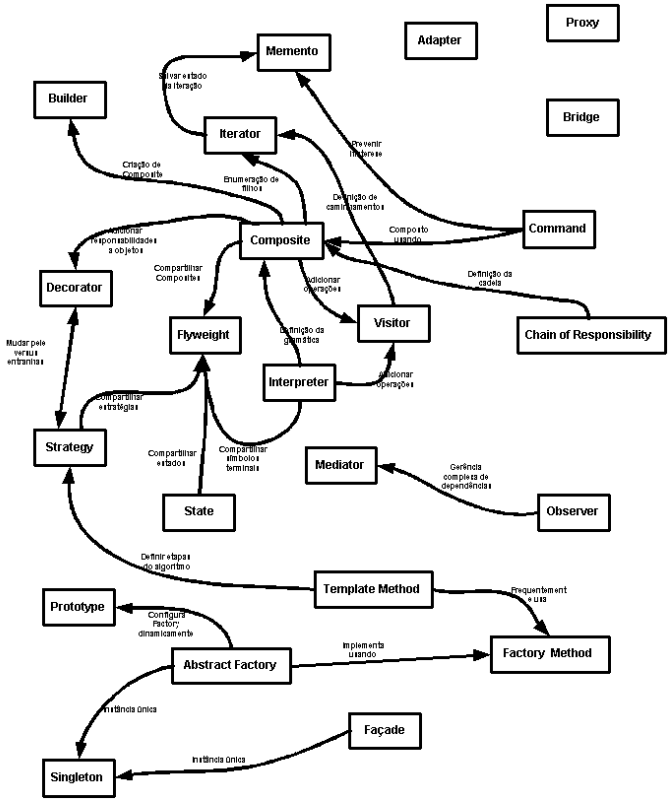
e) Composição

Gabarito – Conceitos OO

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

Padrões de Projeto

CESGRANRIO



FUNDAÇÃO CESGRANRIO

01 – CESGRANRIO – DECEA - 2009

A equipe de desenvolvimento de sistemas de uma empresa utiliza padrões de projetos (design patterns) em seus projetos orientados a objetos. Nesse contexto, **NÃO** é uma característica o:

a) uso de soluções específicas e distintas para projetos similares.

b) identificação de problemas comuns de projeto de software.

c) utilização de soluções testadas e bem documentadas.

d) utilização eficiente de herança, polimorfismo e composição.

e) facilidade na conversão de um modelo de análise em um modelo de implementação.

Introdução

- "Cada padrão descreve um problema que ocorre repetidas vezes em nosso ambiente, e então descreve o núcleo da solução para aquele problema, de tal maneira que pode-se usar essa solução milhões de vezes sem nunca fazê-la da mesma forma duas vezes” Christopher Alexander

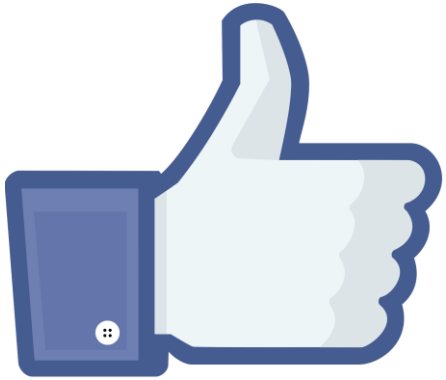
- Objetivo: Solucionar problemas recorrentes de desenvolvimento de software

GOF (Gang of Four)

- 4 Autores se basearam em Christopher Alexander para criar 23 padrões de projeto de Software;

- “Descrição de uma solução para resolver um problema genérico de projeto em um contexto específico [...] Um padrão de projeto dá nome, abstrai e identifica os aspectos-chave de uma estrutura de projeto comum para torná-la reutilizável” Erich Gamma

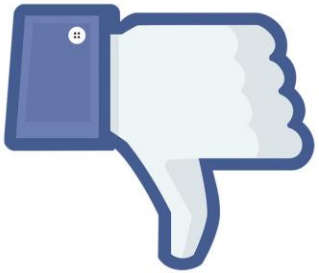
Vantagens



- Padronização do desenvolvimento;
- Utilização de um vocabulário comum;
- Reutilização de soluções;
- Melhores práticas;
- Utilizados para descrever o QUE e POR QUE

Padrões de Projeto NÃO SÃO:

- Soluções prontas para serem aplicadas em qualquer contexto;
- Indicativo de qualidade do código;
- Bibliotecas que irão funcionar de forma específica;
- Códigos prontos para serem reutilizados sem adaptação



01 – CESGRANRIO – DECEA - 2009

A equipe de desenvolvimento de sistemas de uma empresa utiliza padrões de projetos (design patterns) em seus projetos orientados a objetos. Nesse contexto, **NÃO** é uma característica o:

a) uso de soluções específicas e distintas para projetos similares.

b) identificação de problemas comuns de projeto de software.

c) utilização de soluções testadas e bem documentadas.

d) utilização eficiente de herança, polimorfismo e composição.

e) facilidade na conversão de um modelo de análise em um modelo de implementação.

01 – CESGRANRIO – DECEA - 2009

A equipe de desenvolvimento de sistemas de uma empresa utiliza padrões de projetos (design patterns) em seus projetos orientados a objetos. Nesse contexto, **NÃO** é uma característica o:

a) uso de soluções específicas e distintas para projetos similares.

b) identificação de problemas comuns de projeto de software.

c) utilização de soluções testadas e bem documentadas.

d) utilização eficiente de herança, polimorfismo e composição.

e) facilidade na conversão de um modelo de análise em um modelo de implementação.

02 – CESGRANRIO – PETROBRAS - 2010

Em um sistema de software para controlar pedidos para entrega em domicílio, deve haver uma funcionalidade que permita que o atendente solicite a repetição de um pedido anteriormente feito por um cliente. O gerente do restaurante informou que essa funcionalidade aumentaria a agilidade no atendimento aos clientes, visto que muitos deles tendem a fazer pedidos similares aos que já fizeram anteriormente. Ao usar essa funcionalidade, o atendente do restaurante seleciona um pedido cuja composição corresponde a produtos normalmente requisitados pelos clientes e solicita ao sistema a construção de um novo pedido igual ao selecionado. Esse novo pedido pode, então, ser alterado pelo atendente se o cliente solicitar a adição de novos produtos do cardápio, por exemplo. Portanto, a parte principal dessa funcionalidade corresponde a criar uma cópia de um pedido a partir de pedido preexistente. Na implementação dessa funcionalidade, seu desenvolvedor deve utilizar qual padrão de projeto do catálogo GoF (Gang of Four), dentre os listados abaixo?

a) Builder.

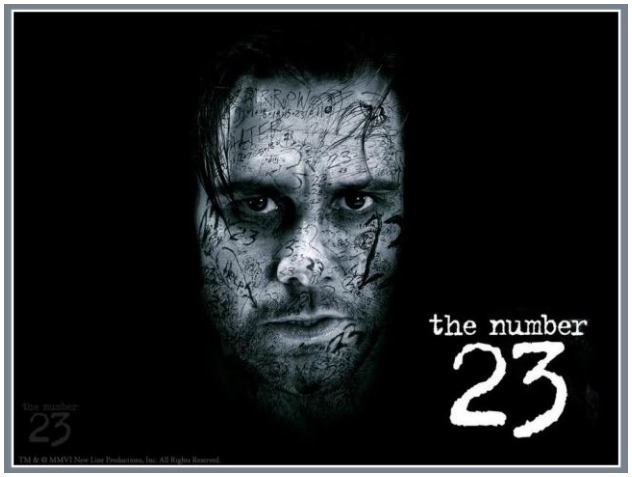
b) Factory Method.

c) Command.

d) Abstract Factory.

e) Prototype.

Padrões GOF



- 23 Padrões

- Escopo:
  - Classe (4)
  - Objeto (20)

- Finalidade:
  - De criação (5)
  - Estruturais (7)
  - Comportamentais (11)

Classificação dos 23 padrões segundo GoF

		Propósito		
		1. Criação	2. Estrutura	3. Comportamento
Escopo	Classe	Factory Method	Class Adapter	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Object Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Mnemônicos - Criacionais

- Uma fábrica abstrata constrói um protótipo único!

- Fábrica = Factory Method
- Abstrata = Abstract Factory
- Constrói = Builder
- Protótipo = Prototype
- Único = Singleton

Fonte: Prof. Rogério Araújo (http://rogerioaraujo.wordpress.com/)

Mnemônicos - Estruturais

- A ponte adaptada é composta de decorações na fachada para o peso mosca se "aproxymar"!

- Ponte = Bridge
- Adaptada = Adapter
- Composta = Composite
- Decorações = Decorator
- Fachada = Façade
- Peso Mosca = Flyweight
- Aproxymar = Proxy

Fonte: Prof. Rogério Araújo (http://rogerioaraujo.wordpress.com/)

02 – CESGRANRIO – PETROBRAS - 2010

Em um sistema de software para controlar pedidos para entrega em domicílio, deve haver uma funcionalidade que permita que o atendente solicite a repetição de um pedido anteriormente feito por um cliente. O gerente do restaurante informou que essa funcionalidade aumentaria a agilidade no atendimento aos clientes, visto que muitos deles tendem a fazer pedidos similares aos que já fizeram anteriormente. Ao usar essa funcionalidade, o atendente do restaurante seleciona um pedido cuja composição corresponde a produtos normalmente requisitados pelos clientes e solicita ao sistema a construção de um novo pedido igual ao selecionado. Esse novo pedido pode, então, ser alterado pelo atendente se o cliente solicitar a adição de novos produtos do cardápio, por exemplo. Portanto, a parte principal dessa funcionalidade corresponde a criar uma cópia de um pedido a partir de pedido preexistente. Na implementação dessa funcionalidade, seu desenvolvedor deve utilizar qual padrão de projeto do catálogo GoF (Gang of Four), dentre os listados abaixo?

a) Builder.

b) Factory Method.

c) Command.

d) Abstract Factory.

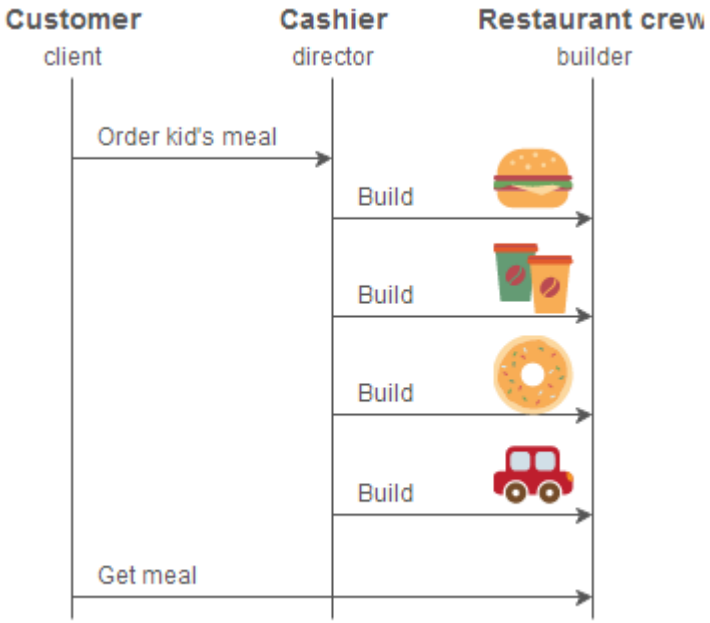
e) Prototype.

Builder *(Criacional)*

“Separar a construção de um objeto complexo de sua representação para que o mesmo processo de construção possa criar representações diferentes”

Builder *(Criacional)*

- Refeições consistem de um item principal, um item para acompanhar, uma bebida e um brinquedo. Pode haver variação do conteúdo, mas o processo de criação é o mesmo

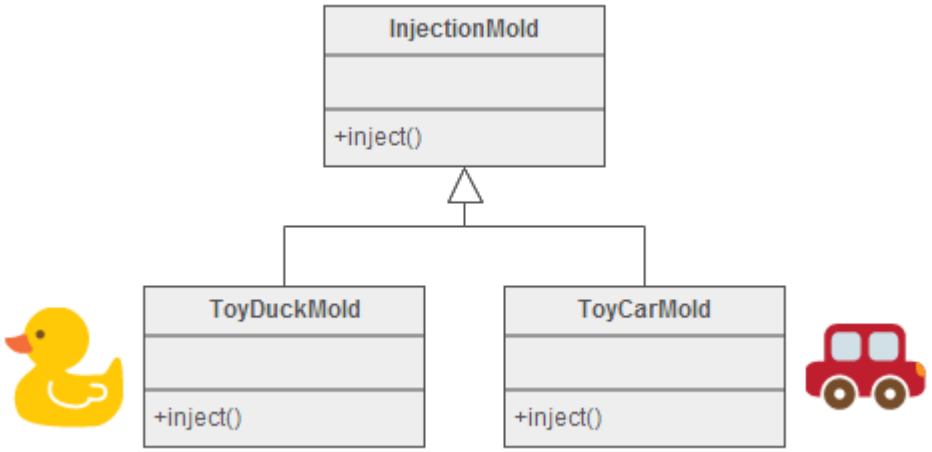


Factory Method *(Criacional)*

Definir uma interface para criar um objeto mas deixar que subclasses decidam que classe instanciar.

Factory Method *(Criacional)*

- Fabricantes de brinquedos plásticos injetam o plástico no molde de acordo com a forma desejada. A classe (car, duck, figure) vai ser determinada pelo molde a ser injetado.

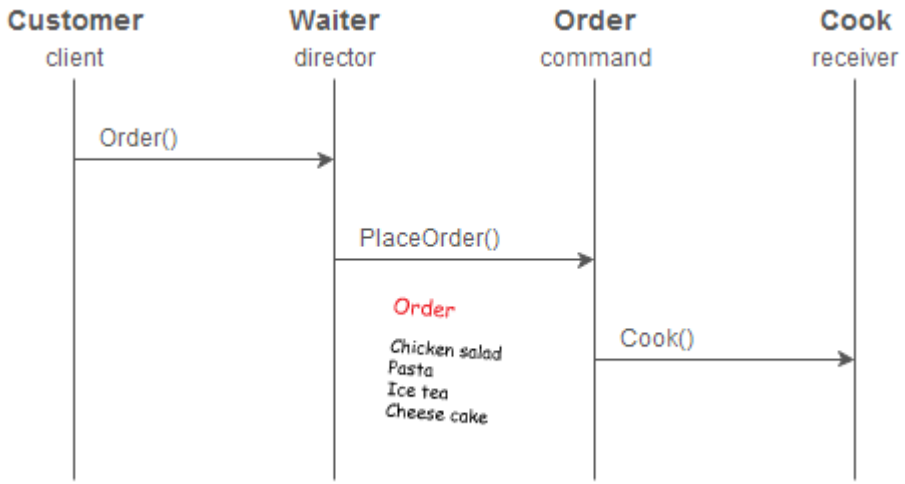


Command *(Comportamental)*

“Encapsular uma requisição como um objeto, permitindo que clientes parametrizem diferentes requisições, filas ou requisições de log, e suportar operações reversíveis.”

Command *(Comportamental)*

- O garçom tem uma ordem e encapsula . A ordem é colocada em uma fila para uma cozinheira. Note que as ordem utilizadas pelos clientes não depende do menu, e pode suportar comandos para vários itens.

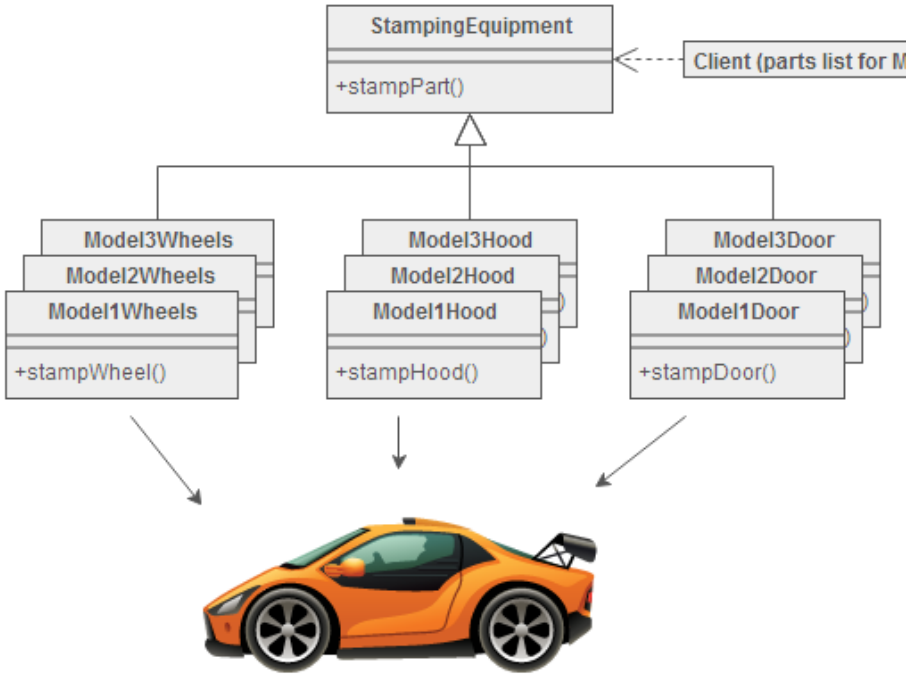


Abstract Factory *(Criacional)*

“Prover uma interface para criar famílias de objetos relacionados ou dependentes sem especificar suas classes concretas”

Abstract Factory *(Criacional)*

- Ao invés de ter um objeto específico, tenho grupos de objetos.

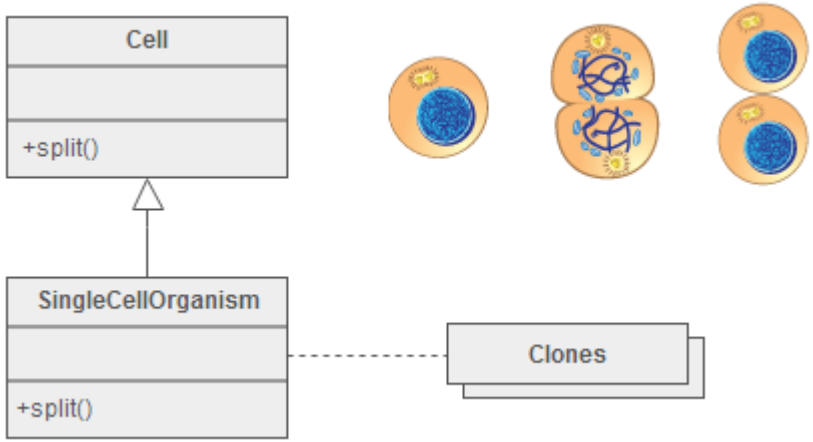


Prototype *(Criacional)*

“Especificar os tipos de objetos a serem criados usando uma instância como protótipo e criar novos objetos ao copiar este protótipo”

Prototype *(Criacional)*

- Uma classe para divisão celular, resulta em duas células idênticas



02 – CESGRANRIO – PETROBRAS - 2010

Em um sistema de software para controlar pedidos para entrega em domicílio, deve haver uma funcionalidade que permita que o atendente solicite a repetição de um pedido anteriormente feito por um cliente. O gerente do restaurante informou que essa funcionalidade aumentaria a agilidade no atendimento aos clientes, visto que muitos deles tendem a fazer pedidos similares aos que já fizeram anteriormente. Ao usar essa funcionalidade, o atendente do restaurante seleciona um pedido cuja composição corresponde a produtos normalmente requisitados pelos clientes e solicita ao sistema a construção de um novo pedido igual ao selecionado. Esse novo pedido pode, então, ser alterado pelo atendente se o cliente solicitar a adição de novos produtos do cardápio, por exemplo. Portanto, a parte principal dessa funcionalidade corresponde a criar uma cópia de um pedido a partir de pedido preexistente. Na implementação dessa funcionalidade, seu desenvolvedor deve utilizar qual padrão de projeto do catálogo GoF (Gang of Four), dentre os listados abaixo?

a) Builder.

b) Factory Method.

c) Command.

d) Abstract Factory.

e) Prototype.

02 – CESGRANRIO – PETROBRAS - 2010

Em um sistema de software para controlar pedidos para entrega em domicílio, deve haver uma funcionalidade que permita que o atendente solicite a repetição de um pedido anteriormente feito por um cliente. O gerente do restaurante informou que essa funcionalidade aumentaria a agilidade no atendimento aos clientes, visto que muitos deles tendem a fazer pedidos similares aos que já fizeram anteriormente. Ao usar essa funcionalidade, o atendente do restaurante seleciona um pedido cuja composição corresponde a produtos normalmente requisitados pelos clientes e solicita ao sistema a construção de um novo pedido igual ao selecionado. Esse novo pedido pode, então, ser alterado pelo atendente se o cliente solicitar a adição de novos produtos do cardápio, por exemplo. Portanto, a parte principal dessa funcionalidade corresponde a criar uma cópia de um pedido a partir de pedido preexistente. Na implementação dessa funcionalidade, seu desenvolvedor deve utilizar qual padrão de projeto do catálogo GoF (Gang of Four), dentre os listados abaixo?

a) Builder.

b) Factory Method.

c) Command.

d) Abstract Factory.

e) Prototype.

03 – CESGRANRIO – CHESF - 2012

Durante o desenvolvimento de um sistema para suporte a uma rede social, um desenvolvedor decidiu criar a facilidade de uma pessoa ter uma lista de amigos para poder enviar e-mails, postagens e/ou fotos. Essa lista pode conter um número indefinido de amigos ou de outras listas de amigos, criando uma estrutura recursiva.

O padrão de projeto que descreve essa estrutura é

a) Abstract factory

b) Chain of responsibility

c) Composite

d) Iterator

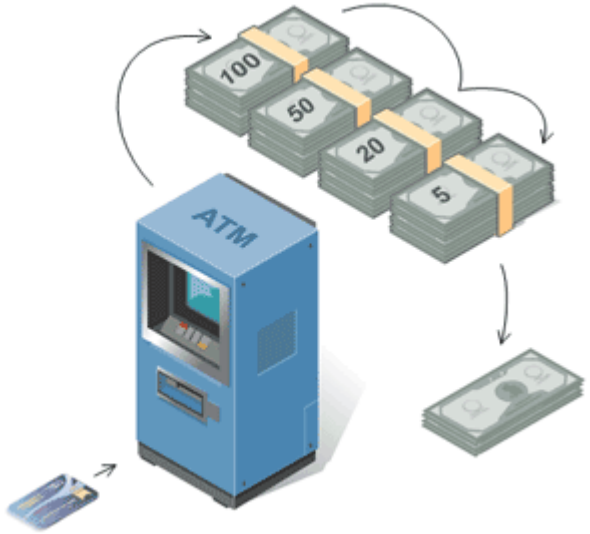
e) Module

Chain of Responsibility *(Comportamental)*

“Evita acoplar o remetente de uma requisição ao seu destinatário ao dar a mais de um objeto a chance de servir a requisição. Compõe os objetos em cascata e passa a requisição pela corrente até que um objeto sirva..”

Chain of Responsability *(Comportamental)*

- Máquinas separadoras de moedas usan o Chain of Responsability, ao invés de ter uma entrada para cada tipo, possui uma única e conforme vai sendo descartada, outro receptor tenta receber.

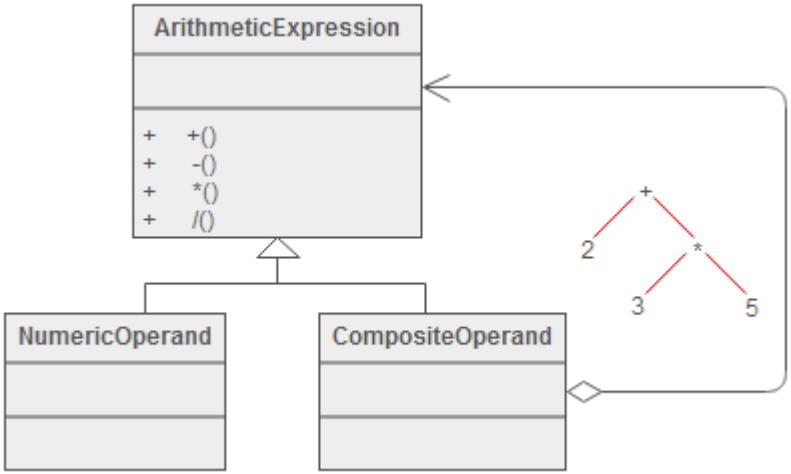


Composite *(Estrutural)*

“Compor objetos em estruturas de árvore para representar hierarquias todo-parte. Composite permite que clientes tratem objetos individuais e composições de objetos de maneira uniforme”

Composite *(Estrutural)*

- Expressão aritmética consiste de um operando, um operador (+ = + /) e outro operando. O operando pode ser um número ou outra expressão aritmética.

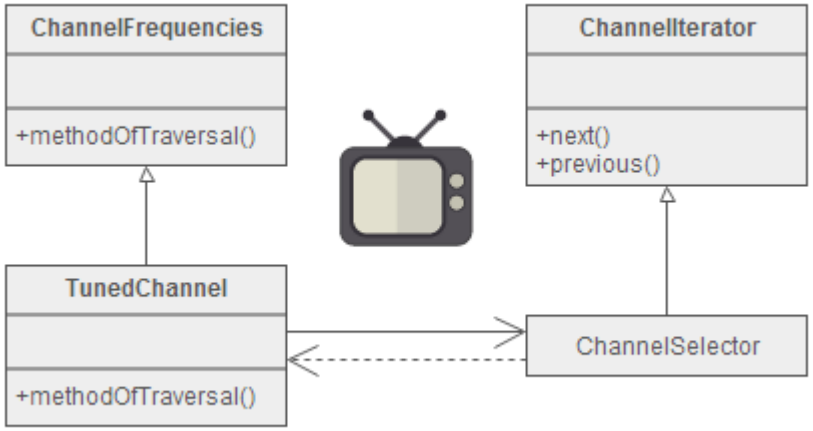


Iterator *(Comportamental)*

“Prover uma maneira de acessar os elementos de um objeto agregado sequencialmente sem expor sua representação interna.”

Iterator *(Comportamental)*

- Nas primeiras televisões um disco era usado para trocar o canal e era necessário rodar até encontrar a sintonia. Ao navegar através dos canais, o número não é importante, mas a programação sim.



03 – CESGRANRIO – CHESF - 2012

Durante o desenvolvimento de um sistema para suporte a uma rede social, um desenvolvedor decidiu criar a facilidade de uma pessoa ter uma lista de amigos para poder enviar e-mails, postagens e/ou fotos. Essa lista pode conter um número indefinido de amigos ou de outras listas de amigos, criando uma estrutura recursiva.

O padrão de projeto que descreve essa estrutura é

a) Abstract factory

b) Chain of responsibility

c) Composite

d) Iterator

e) Module

03 – CESGRANRIO – CHESF - 2012

Durante o desenvolvimento de um sistema para suporte a uma rede social, um desenvolvedor decidiu criar a facilidade de uma pessoa ter uma lista de amigos para poder enviar e-mails, postagens e/ou fotos. Essa lista pode conter um número indefinido de amigos ou de outras listas de amigos, criando uma estrutura recursiva.

O padrão de projeto que descreve essa estrutura é

a) Abstract factory

b) Chain of responsibility

c) Composite

d) Iterator

e) Module

Composite trata todos os elementos como sendo do mesmo tipo, como sendo componentes, na forma de uma árvore, e quando necessário utilizará recursividade.

04 – CESGRANRIO – FINEP- 2011

Sabendo-se que os padrões de projeto podem ser classificados considerando-se o critério de escopo, qual, dentre os padrões de projeto abaixo, possui escopo de classe?

a) Adapter

b) Decorator

c) Mediator

d) Proxy

e) Observer

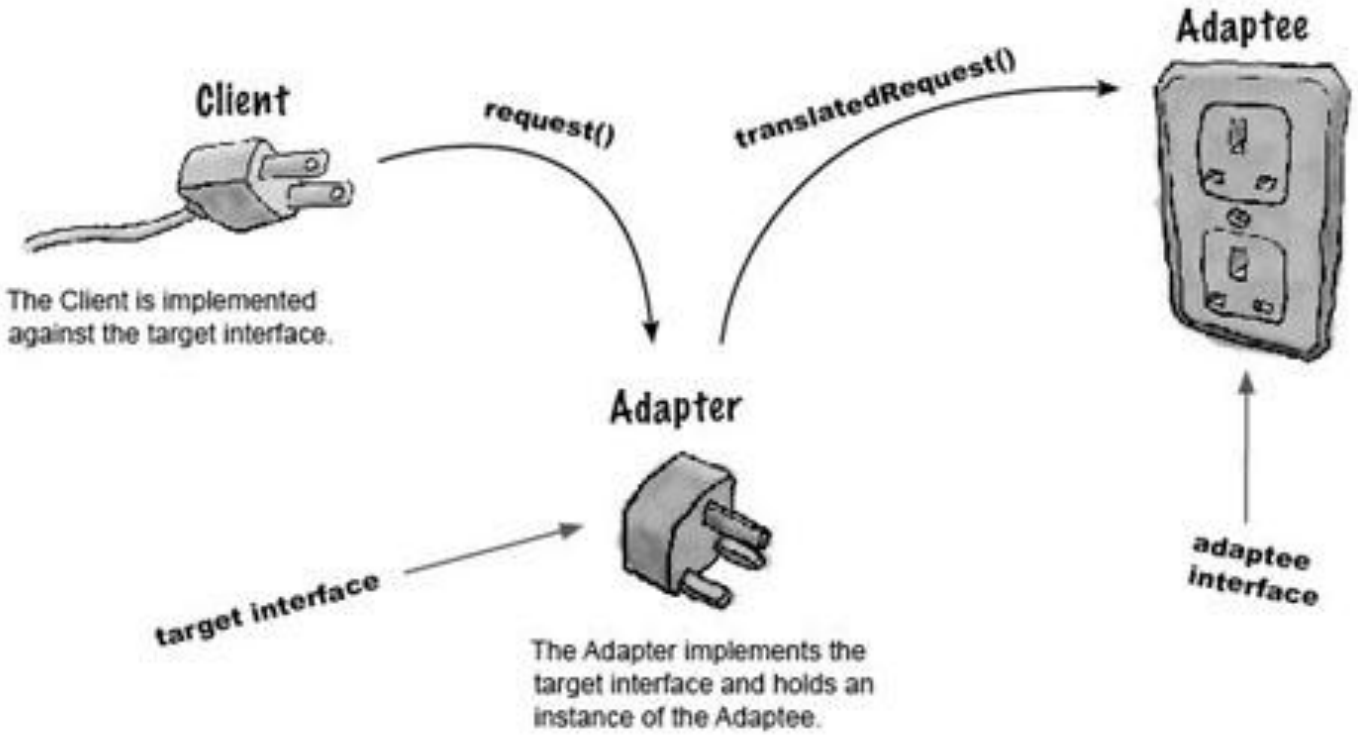
Classificação dos 23 padrões segundo GoF

		Propósito		
		1. Criação	2. Estrutura	3. Comportamento
Escopo	Classe	Factory Method	Class Adapter	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Object Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Adapter *(Estrutural)*

“Converter a interface de uma classe em outra interface esperada pelos clientes. Permite a comunicação entre classes que não poderiam trabalhar juntas devido à incompatibilidade de suas interfaces”

Adapter *(Estrutural)*

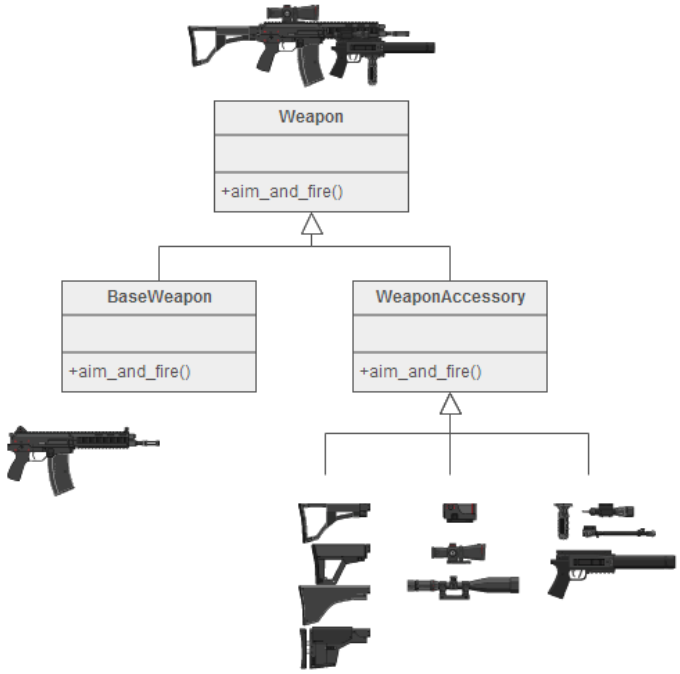


Decorator *(Estrutural)*

“Anexar responsabilidades adicionais a um objeto dinamicamente, Decorators oferecem uma alternativa flexível ao uso de herança para estender uma funcionalidade”

Decorator *(Estrutural)*

- Uma arma de guerra possui vários componentes que irão variar dependendo do momento.

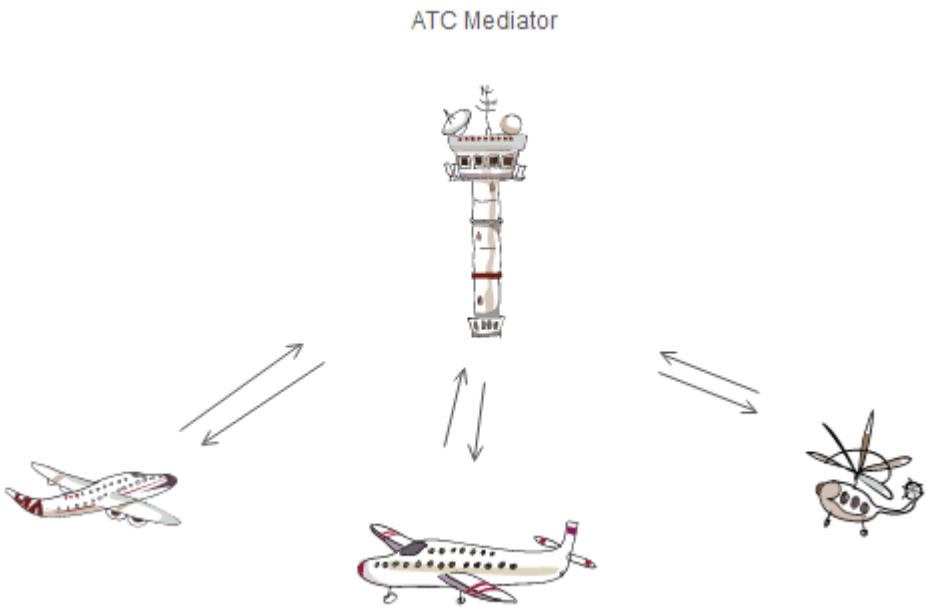


Mediator *(Comportamental)*

“Definir um ojbeto que encapsula como um conjunto de objetos interagem. Mediator promove acoplamento fraco ao manter objetos que não se referem um ao outro explicitamente, permitindo varias sua interação”

Mediator *(Comportamental)*

- Torre de controle de um aeroporto permite que os aviões se comuniquem com a torre ao invés de se comunicarem uns com os outros.

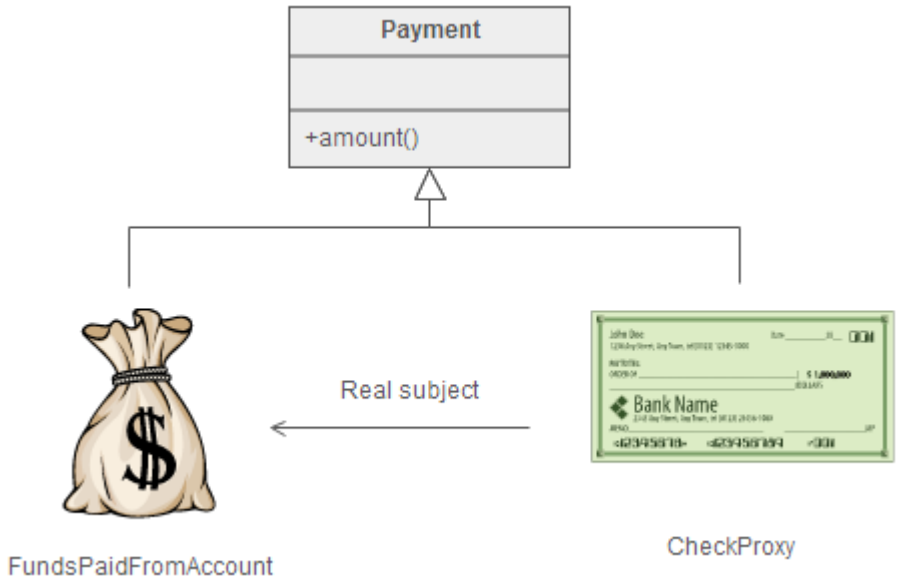


Proxy *(Estrutural)*

“Prover um substituto ou ponto através do qual um objeto possa controlar o acesso a outro”

Proxy *(Estrutural)*

- Um cheque é um exemplo de proxy, pois pode ser utilizado no lugar do dinheiro.

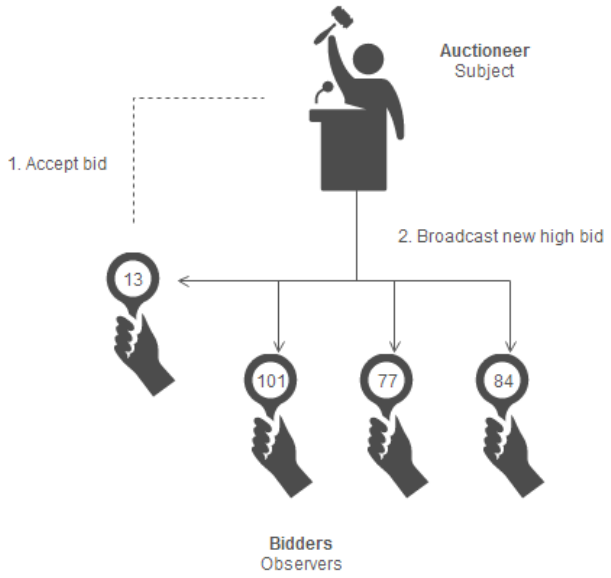


Observer *(Comportamental)*

“Definir uma dependência um para muitos entre objetos para que quando um objeto mudar de estado, todos os seus dependentes sejam notificados e atualizados.”

Observer *(Comportamental)*

- Cada licitante possui uma placa numerada que é utilizada para uma oferta. O leiloeiro observa quando uma placa é aumentada. A mudança do preço da oferta é transmitida para todos os licitantes.



04 – CESGRANRIO – FINEP- 2011

Sabendo-se que os padrões de projeto podem ser classificados considerando-se o critério de escopo, qual, dentre os padrões de projeto abaixo, possui escopo de classe?

a) Adapter

b) Decorator

c) Mediator

d) Proxy

e) Observer

04 – CESGRANRIO – FINEP- 2011

Sabendo-se que os padrões de projeto podem ser classificados considerando-se o critério de escopo, qual, dentre os padrões de projeto abaixo, possui escopo de classe?

a) Adapter
b) Decorator
c) Mediator
d) Proxy
e) Observer

O escopo de Classes possui tais padrões: **Factory Method**(Criação), **Adapter**(Estrutural), **Interpreter** e **Template Method** (Comportamentais).

05 – CESGRANRIO – TRANSPETRO - 2011

Relacione os padrões de projeto às suas indicações de uso.

I - Ponte
II - Observador
III - Decorador
P - Mostra como adicionar responsabilidades aos objetos dinamicamente.
Q - Separa a abstração de um objeto de sua implementação.
R - Define e mantém dependência entre objetos.
S - Define um objeto que encapsula como um conjunto de objetos que interagem.
As associações corretas são:

a) I - P , II - Q , III - R

b) I - Q , II - P , III - S

c) I - Q , II - R , III - P

d) I - R , II - P , III - S

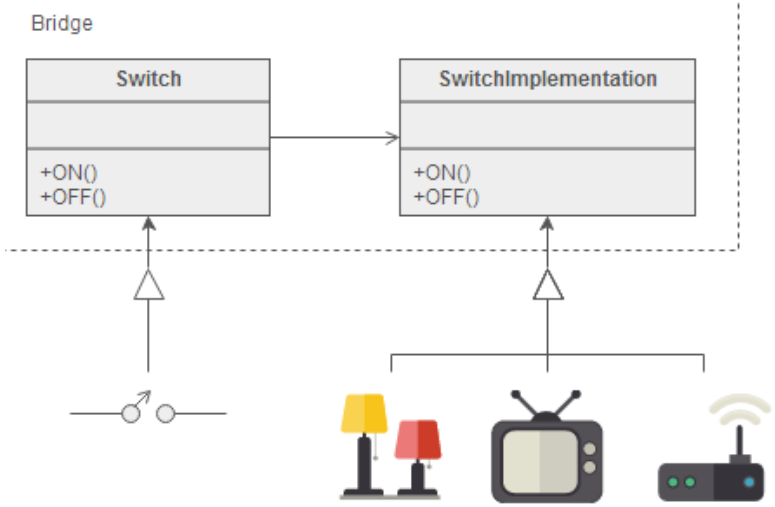
e) I - S , II - R , III - Q

Bridge

“Desacoplar uma abstração de sua implementação para que os dois possam variar independentemente”

Bridge

- Um interruptor de casa controlando luzes, ventiladores de teto, etc., é um exemplo de Bridge. O objetivo da chave é transformar um dispositivo ligado ou desligado. O botão de liga desliga pode ser implementado de várias formas diferentes.



05 – CESGRANRIO – TRANSPETRO - 2011

Relacione os padrões de projeto às suas indicações de uso.

I - Ponte Implementar um design que permita total desacoplamento entre interface e implementação.
II - Observador Desacopla um objeto do conhecimento de que outros objetos dependem dele.
III - Decorador Adiciona responsabilidades a um objeto dinamicamente.
P - Mostra como adicionar responsabilidades aos objetos dinamicamente.
Q - Separa a abstração de um objeto de sua implementação.
R - Define e mantém dependência entre objetos.
S - Define um objeto que encapsula como um conjunto de objetos que interagem. Mediator
As associações corretas são:

a) I - P , II - Q , III - R
b) I - Q , II - P , III - S
c) I - Q , II - R , III - P
d) I - R , II - P , III - S
e) I - S , II - R , III - Q

06 – CESGRANRIO – PETROBRÁS- 2010

Em um sistema de software para controlar pedidos para entrega em domicílio, deve haver uma funcionalidade que permita que o atendente solicite a repetição de um pedido anteriormente feito por um cliente. O gerente do restaurante informou que essa funcionalidade aumentaria a agilidade no atendimento aos clientes, visto que muitos deles tendem a fazer pedidos similares aos que já fizeram anteriormente. Ao usar essa funcionalidade, o atendente do restaurante seleciona um pedido cuja composição corresponde a produtos normalmente requisitados pelos clientes e solicita ao sistema a construção de um novo pedido igual ao selecionado. Esse novo pedido pode, então, ser alterado pelo atendente se o cliente solicitar a adição de novos produtos do cardápio, por exemplo. Portanto, a parte principal dessa funcionalidade corresponde a criar uma cópia de um pedido a partir de pedido preexistente. Na implementação dessa funcionalidade, seu desenvolvedor deve utilizar qual padrão de projeto do catálogo GoF (Gang of Four), dentre os listados abaixo?

a) Builder.

b) Factory Method.

c) Command.

d) Abstract Factory.

e) Prototype.

06 – CESGRANRIO – PETROBRÁS- 2010

Em um sistema de software para controlar pedidos para entrega em domicílio, deve haver uma funcionalidade que permita que o atendente solicite a repetição de um pedido anteriormente feito por um cliente. O gerente do restaurante informou que essa funcionalidade aumentaria a agilidade no atendimento aos clientes, visto que muitos deles tendem a fazer pedidos similares aos que já fizeram anteriormente. Ao usar essa funcionalidade, o atendente do restaurante seleciona um pedido cuja composição corresponde a produtos normalmente requisitados pelos clientes e solicita ao sistema a construção de um novo pedido igual ao selecionado. Esse novo pedido pode, então, ser alterado pelo atendente se o cliente solicitar a adição de novos produtos do cardápio, por exemplo. Portanto, a parte principal dessa funcionalidade corresponde a criar uma cópia de um pedido a partir de pedido preexistente. Na implementação dessa funcionalidade, seu desenvolvedor deve utilizar qual padrão de projeto do catálogo GoF (Gang of Four), dentre os listados abaixo?

a) Builder.

b) Factory Method.

"Especificar os tipos de objetos a serem criados usando uma instância como protótipo e criar novos objetos ao copiar este protótipo."

c) Command.

d) Abstract Factory.

e) Prototype.

07 – CESGRANRIO – PETROBRÁS- 2010

Um dos participantes da equipe de desenvolvimento de um framework deve implementar uma operação em uma das classes desse framework. Seja X o nome dessa classe. Essa operação implementa um algoritmo em particular. Entretanto, há passos desse algoritmo que devem ser implementados pelos usuários do framework através da definição de uma subclasse de X. Sendo assim, qual o padrão de projeto do catálogo GoF (Gang of Four) a ser usado pelo desenvolvedor do framework na implementação da referida operação, dentre os listados a seguir?

a) Singleton.

b) Decorator.

c) Interpreter.

d) Template Method.

e) Observer.

Singleton

“Garantir que uma classe só tenha uma única instância, e prover um ponto de acesso global a ela”

Singleton

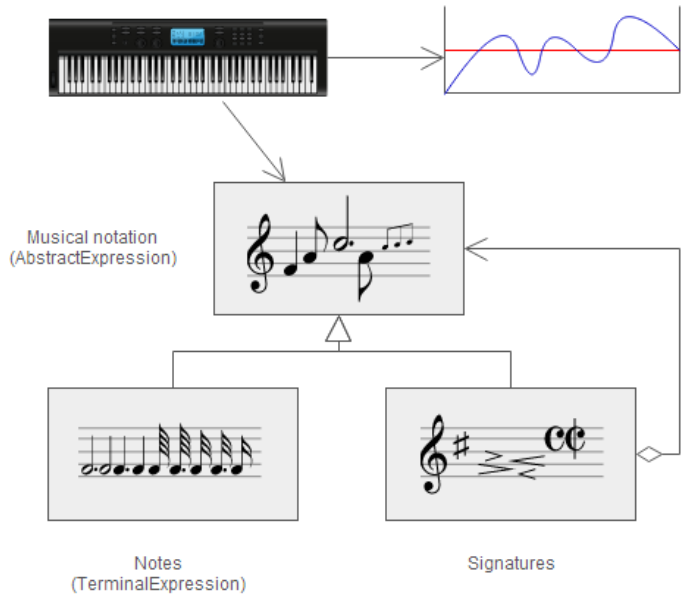
- Escritório do Presidente dos EUA é um Singleton. Da forma que a constituição foi feita, só pode ter um presidente ativo por vez, independente da identidade da pessoa. O “Presidente” é um ponto global que identifica a pessoa no escritório.

Interpreter

“Dada uma linguagem, definir uma representação para sua gramática junto com um interpretador que usa a representação para interpretar sentenças na linguagem.”

Interpreter

- Altura de um som e sua duração pode ser representada em notação musical. Esta notação fornece a linguagem da música. Músicos conseguem reproduzir o tom e a duração de cada nota.

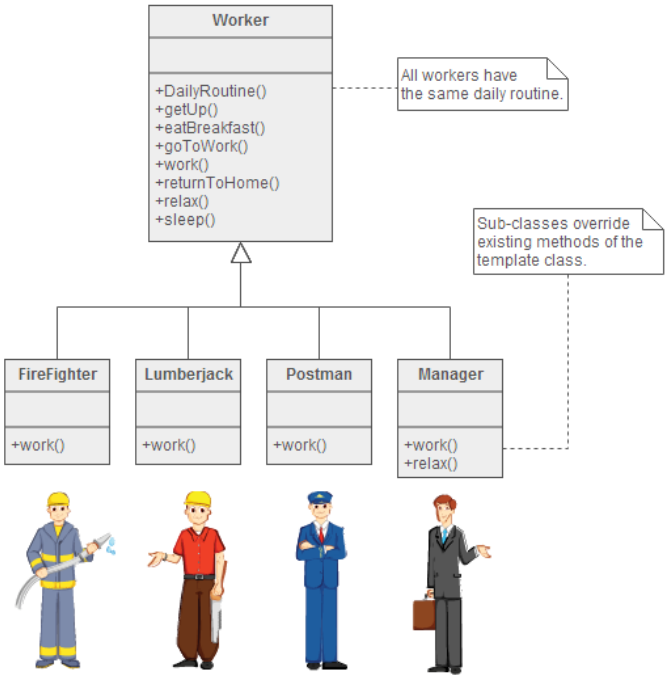


Template Method

“Definir o esqueleto de um algoritmo dentro de uma operação, deixando alguns passos a serem preenchidos pelas subclasses. Permite que suas subclasses redefinam certos passos de um algoritmo sem mudar sua estrutura.”

Template Method

- Um trabalhador terá sempre comportamentos padrões, sendo que apenas algumas variações irão ocorrer dependendo do tipo de trabalhador.



07 – CESGRANRIO – PETROBRÁS- 2010

Um dos participantes da equipe de desenvolvimento de um framework deve implementar uma operação em uma das classes desse framework. Seja X o nome dessa classe. Essa operação implementa um algoritmo em particular. Entretanto, há passos desse algoritmo que devem ser implementados pelos usuários do framework através da definição de uma subclasse de X. Sendo assim, qual o padrão de projeto do catálogo GoF (Gang of Four) a ser usado pelo desenvolvedor do framework na implementação da referida operação, dentre os listados a seguir?

a) Singleton.

b) Decorator.

c) Interpreter.

d) Template Method.

e) Observer.

07 – CESGRANRIO – PETROBRÁS- 2010

Um dos participantes da equipe de desenvolvimento de um framework deve implementar uma operação em uma das classes desse framework. Seja X o nome dessa classe. Essa operação implementa um algoritmo em particular. Entretanto, há passos desse algoritmo que devem ser implementados pelos usuários do framework através da definição de uma subclasse de X. Sendo assim, qual o padrão de projeto do catálogo GoF (Gang of Four) a ser usado pelo desenvolvedor do framework na implementação da referida operação, dentre os listados a seguir?

a) Singleton.

b) Decorator.

c) Interpreter.


d) Template Method.

e) Observer.

"Definir o esqueleto de um algoritmo dentro de uma operação, deixando alguns passos a serem preenchidos pelas subclasses. Template Method permite que suas subclasses redefinam certos passos de um algoritmo sem mudar sua estrutura."

08 – CESGRANRIO – TJ/RO - 2008

Muitos frameworks utilizam um padrão de projeto (design pattern) que se baseia no princípio de projeto "Não nos chame, nós iremos chamar você" (O Princípio de Hollywood: "Don't call us, we'll call you."). Uma classe da aplicação herda de uma classe do framework que possui métodos abstratos, os quais são chamados em outros métodos concretos. A classe da aplicação sobrepõe os métodos abstratos da classe do framework com métodos concretos, permitindo que os métodos concretos da classe do framework sejam chamados. O padrão de projeto (design pattern) em questão chama-se

a) Strategy

b) Singleton

c) Decorator

d) Abstract Factory

e) Template Method

08 – CESGRANRIO – TJ/RO - 2008

Muitos frameworks utilizam um padrão de projeto (design pattern) que se baseia no princípio de projeto "Não nos chame, nós iremos chamar você" (O Princípio de Hollywood: "Don't call us, we'll call you."). Uma classe da aplicação herda de uma classe do framework que possui métodos abstratos, os quais são chamados em outros métodos concretos. A classe da aplicação sobrepõe os métodos abstratos da classe do framework com métodos concretos, permitindo que os métodos concretos da classe do framework sejam chamados. O padrão de projeto (design pattern) em questão chama-se

a) Strategy

b) Singleton

c) Decorator

d) Abstract Factory

e) Template Method

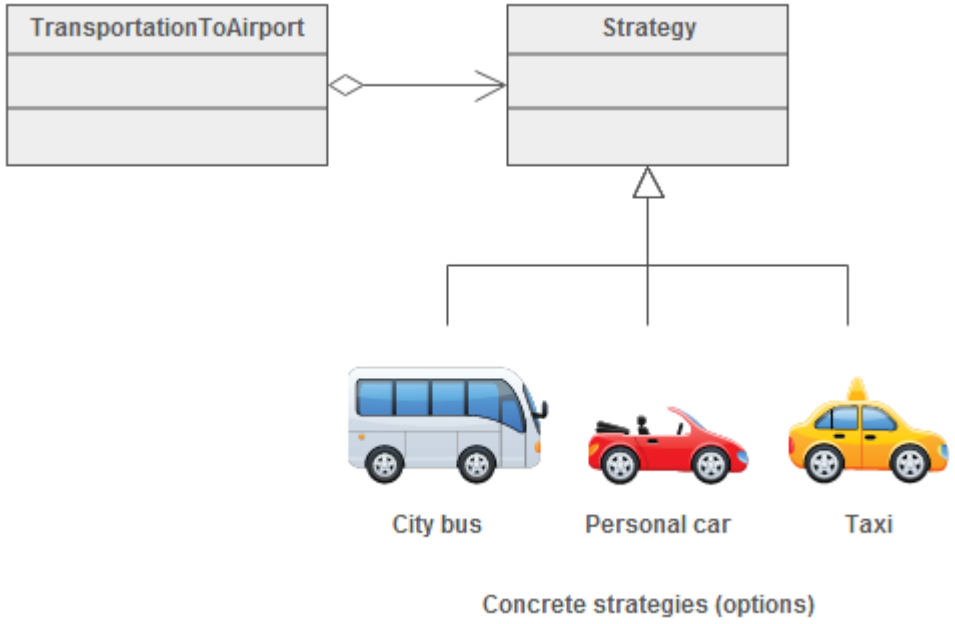
Template Method auxilia na definição de algorítmos com partes dos mesmos definidos por métodos abstratos. As subclasses devem se responsabilizar pelas partes abstratas desse algorítmo, que poderam ser implementadas de muitas formas.

Strategy

“Definir uma família de algoritmos, encapsular cada um, e fazê-los intercambiáveis. Strategy permite que algoritmos mudem independentemente entre clientes que os utilizam.”

Strategy

- Modos de transporte para o aeroporto é um exemplo de Strategy. Existem várias opções, em alguns até helicópteros. O viajante escolhe a melhor estratégia.



09 – CESGRANRIO – BNDES - 2009

Ao consultar informações a respeito dos padrões de projeto Adapter e Bridge, um Analista de Sistemas identificou uma afirmativa **INCORRETA**. Assinale-a.

a) Ambos promovem a flexibilidade ao fornecer um nível de endereçamento indireto para outro objeto.

b) Ambos são padrões estruturais que possuem alguns atributos em comum.

c) O foco do Adapter é a solução de incompatibilidades entre duas interfaces existentes.

d) O Adapter é inferior ao Bridge porque não evita a replicação de código.

e) O Bridge estabelece uma ponte entre uma abstração e suas possíveis implementações.

09 – CESGRANRIO – BNDES - 2009

Ao consultar informações a respeito dos padrões de projeto Adapter e Bridge, um Analista de Sistemas identificou uma afirmativa **INCORRETA**. Assinale-a.

a) Ambos promovem a flexibilidade ao fornecer um nível de endereçamento indireto para outro objeto.

b) Ambos são padrões estruturais que possuem alguns atributos em comum.

c) O foco do Adapter é a solução de incompatibilidades entre duas interfaces existentes.

d) O Adapter é inferior ao Bridge porque não evita a replicação de código. Não existe hierarquia de padrões

e) O Bridge estabelece uma ponte entre uma abstração e suas possíveis implementações.

10 – CESGRANRIO – BNDES - 2009

O presidente de uma empresa determinou que fosse disponibilizado um sistema de vendas na Internet. No entanto, o software de controle de estoque que deve ser acessado pela aplicação de vendas é muito antigo e provê uma API (*Application Programming Interface*) de uso muito complicado. Para que os desenvolvedores possam acessar uma interface mais simples, o arquiteto do sistema pode determinar o uso do padrão de projeto

a) Prototype.

b) Decorator.

c) Observer.

d) Façade.

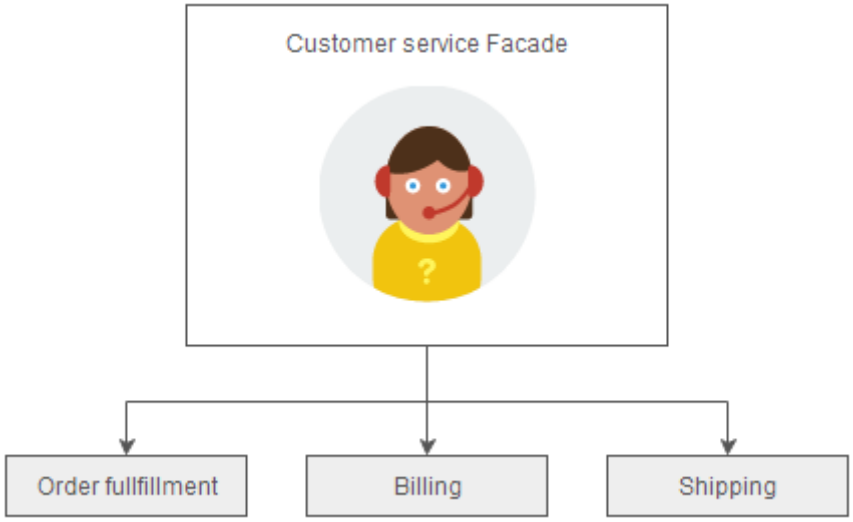
e) Flyweight.

Façade *(Estrutural)*

“Oferecer uma interface única para um conjunto de interfaces de um subsistema. Façade define uma interface de nível mais elevado que torna o subsistema mais fácil de usar”

Façade *(Estrutural)*

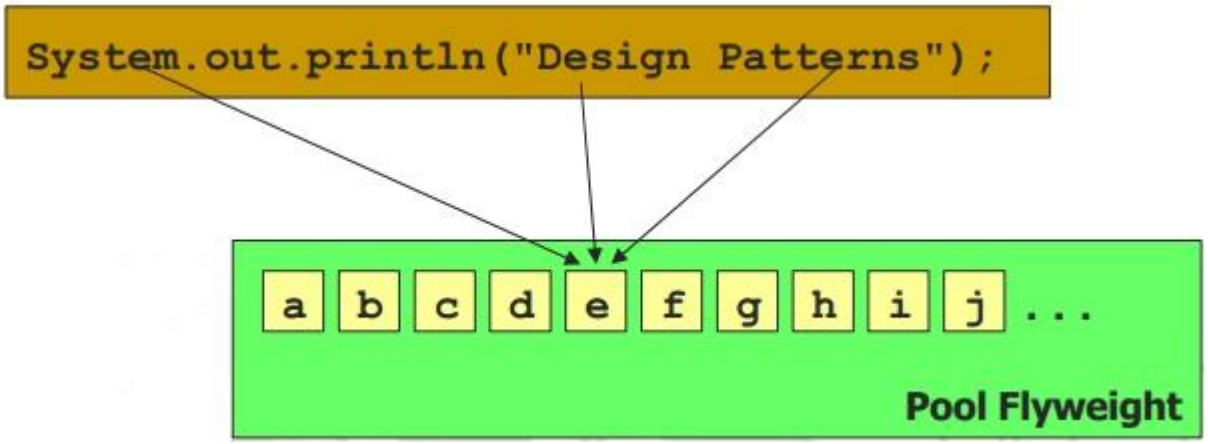
- Serviço de atendimento ao cliente funciona como uma fachada, proporcionando uma interface com os vários departamentos da empresa.

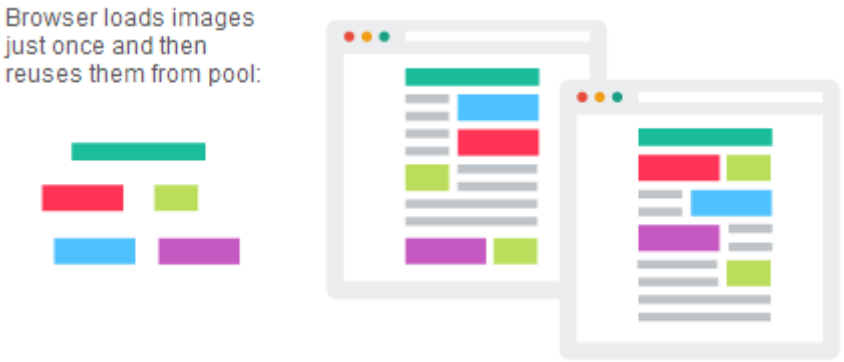


Flyweight *(Estrutural)*

“Usar compartilhamento para suportar grandes quantidades de objetos refinados eficientemente”

Flyweight (Estrutural)





10 – CESGRANRIO – BNDES - 2009

O presidente de uma empresa determinou que fosse disponibilizado um sistema de vendas na Internet. No entanto, o software de controle de estoque que deve ser acessado pela aplicação de vendas é muito antigo e provê uma API (*Application Programming Interface*) de uso muito complicado. Para que os desenvolvedores possam acessar uma interface mais simples, o arquiteto do sistema pode determinar o uso do padrão de projeto

a) Prototype.

b) Decorator.

c) Observer.

d) Façade.

e) Flyweight.

10 – CESGRANRIO – BNDES - 2009

O presidente de uma empresa determinou que fosse disponibilizado um sistema de vendas na Internet. No entanto, o software de controle de estoque que deve ser acessado pela aplicação de vendas é muito antigo e provê uma API (*Application Programming Interface*) de uso muito complicado. Para que os desenvolvedores possam acessar uma interface mais simples, o arquiteto do sistema pode determinar o uso do padrão de projeto

a) Prototype. Permite criar objetos a partir de um modelo

b) Decorator. Alternativa flexível ao uso de subclasses para estender funcionalidade

c) Observer. Dependência um para muitos entre objetos

d) Façade.

e) Flyweight. Manipular vários objetos de granularidade fina

11 – CESGRANRIO – BNDES - 2009

Por motivo de segurança, deseja-se adicionar registro (log) das operações efetuadas no sistema de contabilidade de uma empresa. O arquiteto do sistema decide que deve existir somente uma instância de uma classe de registro (log) e que esta será o ponto de acesso global para os demais componentes do sistema. Que padrão de projeto pode ser utilizado nesse caso?

a) Iterator.

b) Visitor.

c) Bridge.

d) Memento.

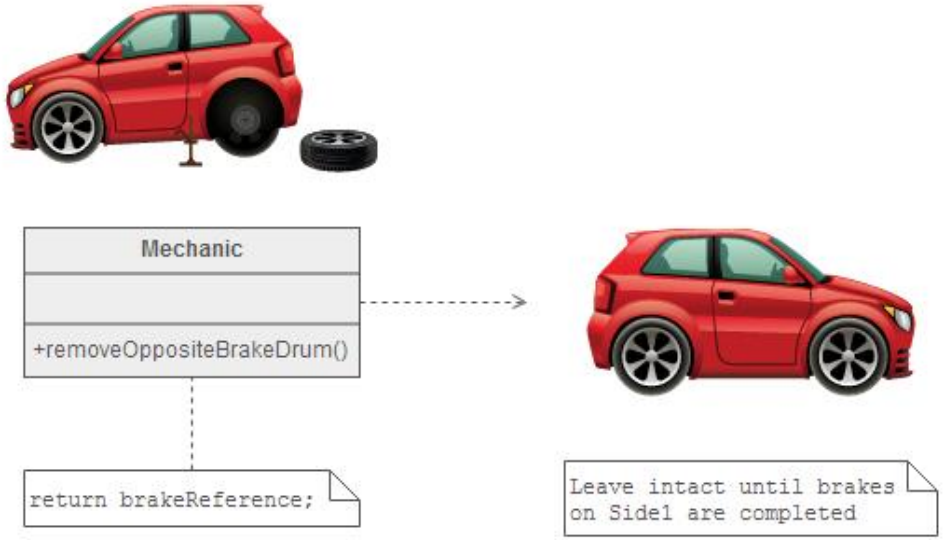
e) Singleton.

Memento *(Comportamental)*

“Sem violar o encapsulamento, capturar e externalizar o estado interno de um objeto para que o objeto possa ter esse estado restaurado posteriormente”

Memento *(Comportamental)*

- Comum entre mecânicos ao reparar o tambor de freio de carros. Somente um lado é desmontado, e o outro serve como uma lembrança da forma que as peças se encaixam.

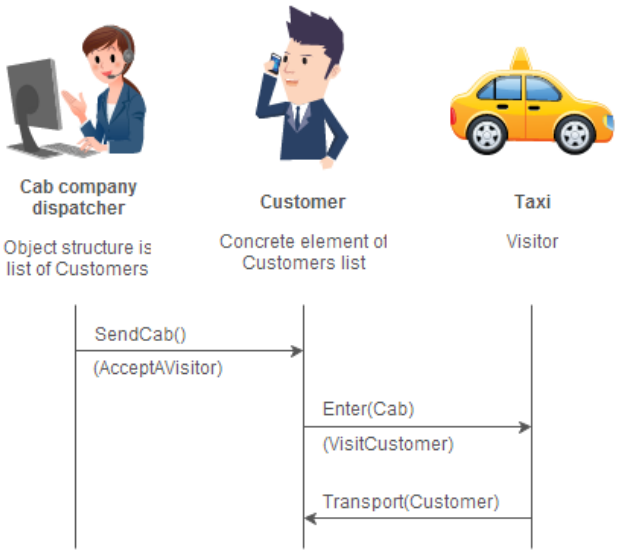


Visitor *(Comportamental)*

“Representar uma operação a ser realizada sobre os elementos de uma estrutura de objetos. Permite definir uma nova operação sem mudar as classes dos elementos nos quais opera.”

Visitor *(Comportamental)*

- Quando uma pessoa pede um taxi à uma empresa, ela se torna parte da lista de clientes. A empresa envia um taxi, ao entrar no taxi (visitor), o cliente não está mais no controle do seu próprio transporte.

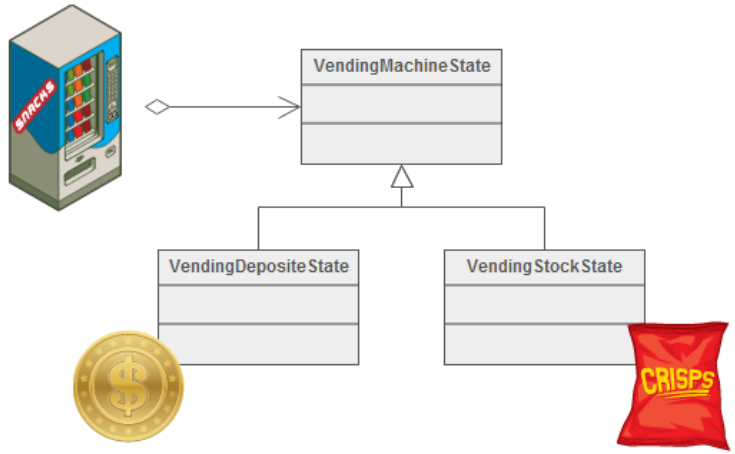


State *(Comportamental)*

“Permitir a um objeto alterar o seu comportamento quando o seu estado interno mudar. O objeto irá aparentar mudar de classe.”

State *(Comportamental)*

- Máquinas de venda tem estados com base no inventário, quantidade de moeda, item selecionado, etc. Quando a moeda é depositada a máquina poderá ter vários estados possíveis.



11 – CESGRANRIO – BNDES - 2009

Por motivo de segurança, deseja-se adicionar registro (log) das operações efetuadas no sistema de contabilidade de uma empresa. O arquiteto do sistema decide que deve existir somente uma instância de uma classe de registro (log) e que esta será o ponto de acesso global para os demais componentes do sistema. Que padrão de projeto pode ser utilizado nesse caso?

a) Iterator.

b) Visitor.

c) Bridge.

d) Memento.

e) Singleton.

11 – CESGRANRIO – BNDES - 2009

Por motivo de segurança, deseja-se adicionar registro (log) das operações efetuadas no sistema de contabilidade de uma empresa. O arquiteto do sistema decide que deve existir somente uma instância de uma classe de registro (log) e que esta será o ponto de acesso global para os demais componentes do sistema. Que padrão de projeto pode ser utilizado nesse caso?

a) Iterator.

b) Visitor.

c) Bridge.

d) Memento.

e) Singleton.

12 – CESGRANRIO – BACEN - 2010

Um arquiteto de software estuda que padrões de projeto são apropriados para o novo sistema de vendas de uma empresa. Ele deve considerar que o padrão

a) Bridge separa a construção de um objeto complexo de sua representação, de modo que o mesmo processo de construção possa criar diferentes representações.

b) Builder garante que uma classe seja instanciada somente uma vez, fornecendo também um ponto de acesso global.

c) Singleton separa uma abstração de sua implementação, de modo que os dois conceitos possam variar de modo independente.

d) Chain of Responsibility evita o acoplamento entre o remetente de uma solicitação e seu destinatário, dando oportunidade para mais de um objeto tratar a solicitação.

e) Template Method utiliza compartilhamento para suportar, eficientemente, grandes quantidades de objetos de granularidade fina.

12 – CESGRANRIO – BACEN - 2010

Um arquiteto de software estuda que padrões de projeto são apropriados para o novo sistema de vendas de uma empresa. Ele deve considerar que o padrão

a) Bridge separa a construção de um objeto complexo de sua representação, de modo que o mesmo processo de construção possa criar diferentes representações. Builder

b) Builder garante que uma classe seja instanciada somente uma vez, fornecendo também um ponto de acesso global. Singleton

c) Singleton separa uma abstração de sua implementação, de modo que os dois conceitos possam variar de modo independente. Bridge

d) Chain of Responsibility evita o acoplamento entre o remetente de uma solicitação e seu destinatário, dando oportunidade para mais de um objeto tratar a solicitação.

e) Template Method utiliza compartilhamento para suportar, eficientemente, grandes quantidades de objetos de granularidade fina. FlyWeight

13 – CESGRANRIO – BNDES- 2008

No âmbito de padrões de arquitetura de aplicações corporativas, que padrão permite que objetos sejam carregados na memória somente quando são efetivamente necessários?

a) *Identity Field*

b) *Association Table Mapping*

c) *Serialized* LOB

d) *Lazy Load*

e) *Embedded Value*

13 – CESGRANRIO – BNDES- 2008

No âmbito de padrões de arquitetura de aplicações corporativas, que padrão permite que objetos sejam carregados na memória somente quando são efetivamente necessários?

a) *Identity Field*

b) *Association Table Mapping*

c) *Serialized* LOB

d) *Lazy Load*

e) *Embedded Value*

14 – CESGRANRIO – BNDES- 2008

Ao alterar o estado de uma classe, o desenvolvedor deseja que uma ou mais classes da interface gráfica sejam modificadas. Entretanto, o desenvolvedor não acha interessante criar um acoplamento forte entre essas classes. Qual padrão de projeto comportamental é mais adequado para resolver essa situação?

a) Composite

b) Adapter

c) Observer

d) Abstract Factory

e) Decorator

14 – CESGRANRIO – BNDES- 2008

Ao alterar o estado de uma classe, o desenvolvedor deseja que uma ou mais classes da interface gráfica sejam modificadas. Entretanto, o desenvolvedor não acha interessante criar um acoplamento forte entre essas classes. Qual padrão de projeto comportamental é mais adequado para resolver essa situação?

a) Composite estrutural

b) Adapter estrutural

➡ c) Observer

d) Abstract Factory criacional

e) Decorator estrutural

15 – CESGRANRIO – EPE - 2006

Considere os padrões que podem ser utilizados em um projeto de software e relacione os padrões com suas respectivas intenções primárias.

Padrão

I - Bridge

II - Builder

III - Factory Method

Intenção primária

(P) Padrão estrutural cuja intenção é converter a interface de uma classe em outra interface, esperada pelos clientes, permitindo que classes com interfaces incompatíveis trabalhem em conjunto, o que, de outra forma, seria impossível.

(Q) Padrão estrutural cuja intenção é desacoplar uma abstração da sua implementação, de modo que as duas possam variar independentemente.

(R) Padrão de criação cuja intenção é separar a construção de um objeto complexo da sua representação de modo que o mesmo processo de construção possa criar diferentes representações.

(S) Padrão de criação cuja intenção é definir uma interface para criar um objeto, mas deixando as subclasses decidirem que classe instanciar. Ele permite adiar a instanciação para subclasses.

A relação correta é:

a) I - P , II - Q , III - R

b) I - P , II - Q , III - S

c) I - Q , II - R , III - S

d) I - Q , II - R , III - P

e) I - Q , II - S , III - P

15 – CESGRANRIO – EPE - 2006

Considere os padrões que podem ser utilizados em um projeto de software e relacione os padrões com suas respectivas intenções primárias.

Padrão

I - Bridge
II - Builder
III - Factory Method

Intenção primária

(P) Padrão estrutural cuja intenção é converter a interface de uma classe em outra interface, esperada pelos clientes, permitindo que classes com interfaces incompatíveis trabalhem em conjunto, o que, de outra forma, seria impossível. Adapter

(Q) Padrão estrutural cuja intenção é desacoplar uma abstração da sua implementação, de modo que as duas possam variar independentemente.

(R) Padrão de criação cuja intenção é separar a construção de um objeto complexo da sua representação de modo que o mesmo processo de construção possa criar diferentes representações.

(S) Padrão de criação cuja intenção é definir uma interface para criar um objeto, mas deixando as subclasses decidirem que classe instanciar. Ele permite adiar a instanciação para subclasses.

A relação correta é:

a) I - P , II - Q , III - R

b) I - P , II - Q , III - S

c) I - Q , II - R , III - S

d) I - Q , II - R , III - P

e) I - Q , II - S , III - P

16 – CESGRANRIO – PETROBRÁS - 2006

Christopher Alexander afirma: "cada padrão descreve um problema no nosso ambiente e o cerne da sua solução, de tal forma que você possa usar essa solução mais de um milhão de vezes, sem nunca faze-lo da mesma maneira". Muito embora Alexander estivesse falando a cerca de padrões em construções e cidades, o que ele diz é verdadeiro em relação aos padrões de projeto orientados a objeto. Neste caso, as soluções são expressas em termos de objetos e interfaces em vez de paredes e portas, mas no cerne de ambos os tipos de padrões está a solução para um problema num determinado contexto. Quanto à indicação para o uso dos padrões de projeto é **FALSO** afirmar que o padrão:

a) Abstract Factory é indicado quando: um sistema deve ser independente de como seus produtos são criados, compostos ou representados; um sistema deve ser configurado como um produto de uma família de múltiplos produtos; uma família de objetos-produto for projetada para ser usada em conjunto, e você necessita garantir esta restrição; você quer fornecer uma biblioteca de classes de produtos e quer revelar somente suas interfaces, não suas implementações.

b) Builder é indicado quando: uma classe não pode antecipar a classe de objetos que deve criar; uma classe quer que suas subclasses especifiquem os objetos que criam; classes delegam responsabilidade para uma dentre várias subclasses auxiliares, e você quer localizar o conhecimento de qual subclasse auxiliar que é a delegada.

c) Mediator é indicado quando: um conjunto de objetos se comunica de maneiras bem definidas, porém complexas; a reutilização de um objeto é difícil porque ele referencia e se comunica com muitos outros objetos; um comportamento que está distribuído entre várias classes deveria ser customizável, ou adaptável, sem excessiva especialização em subclasses.

d) Memento é indicado quanto: um instantâneo de estado de um objeto deve ser salvo de maneira que possa ser restaurado para esse estado mais tarde; uma interface direta para obtenção do estado exporia detalhes de implementação e romperia o encapsulamento do objeto.

e) Composite é indicado quando: quiser representar hierarquias partes-todo de objetos; quiser que os clientes sejam capazes de ignorar a diferença entre composições de objetos e objetos individuais, neste caso, os clientes tratarão todos os objetos na estrutura composta de maneira uniforme

16 – CESGRANRIO – PETROBRÁS - 2006

Christopher Alexander afirma: "cada padrão descreve um problema no nosso ambiente e o cerne da sua solução, de tal forma que você possa usar essa solução mais de um milhão de vezes, sem nunca faze-lo da mesma maneira". Muito embora Alexander estivesse falando a cerca de padrões em construções e cidades, o que ele diz é verdadeiro em relação aos padrões de projeto orientados a objeto. Neste caso, as soluções são expressas em termos de objetos e interfaces em vez de paredes e portas, mas no cerne de ambos os tipos de padrões está a solução para um problema num determinado contexto. Quanto à indicação para o uso dos padrões de projeto é **FALSO** afirmar que o padrão:

a) Abstract Factory é indicado quando: um sistema deve ser independente de como seus produtos são criados, compostos ou representados; um sistema deve ser configurado como um produto de uma família de múltiplos produtos; uma família de objetos-produto for projetada para ser usada em conjunto, e você necessita garantir esta restrição; você quer fornecer uma biblioteca de classes de produtos e quer revelar somente suas interfaces, não suas implementações. Factory Method

b) Builder é indicado quando: uma classe não pode antecipar a classe de objetos que deve criar; uma classe quer que suas subclasses especifiquem os objetos que criam; classes delegam responsabilidade para uma dentre várias subclasses auxiliares, e você quer localizar o conhecimento de qual subclasse auxiliar que é a delegada.

c) Mediator é indicado quando: um conjunto de objetos se comunica de maneiras bem definidas, porém complexas; a reutilização de um objeto é difícil porque ele referencia e se comunica com muitos outros objetos; um comportamento que está distribuído entre várias classes deveria ser customizável, ou adaptável, sem excessiva especialização em subclasses.

d) Memento é indicado quanto: um instantâneo de estado de um objeto deve ser salvo de maneira que possa ser restaurado para esse estado mais tarde; uma interface direta para obtenção do estado exporia detalhes de implementação e romperia o encapsulamento do objeto.

e) Composite é indicado quando: quiser representar hierarquias partes-todo de objetos; quiser que os clientes sejam capazes de ignorar a diferença entre composições de objetos e objetos individuais, neste caso, os clientes tratarão todos os objetos na estrutura composta de maneira uniforme

17 – CESGRANRIO – PETROBRAS - 2013

Deseja-se que uma aplicação possa manipular diferentes tipos de bancos de dados de modo transparente às classes que necessitam de serviços de acesso a dados. Inicialmente será necessário fornecer suporte a bancos de dados XML e SQL, entretanto novos tipos poderão ser futuramente adicionados.

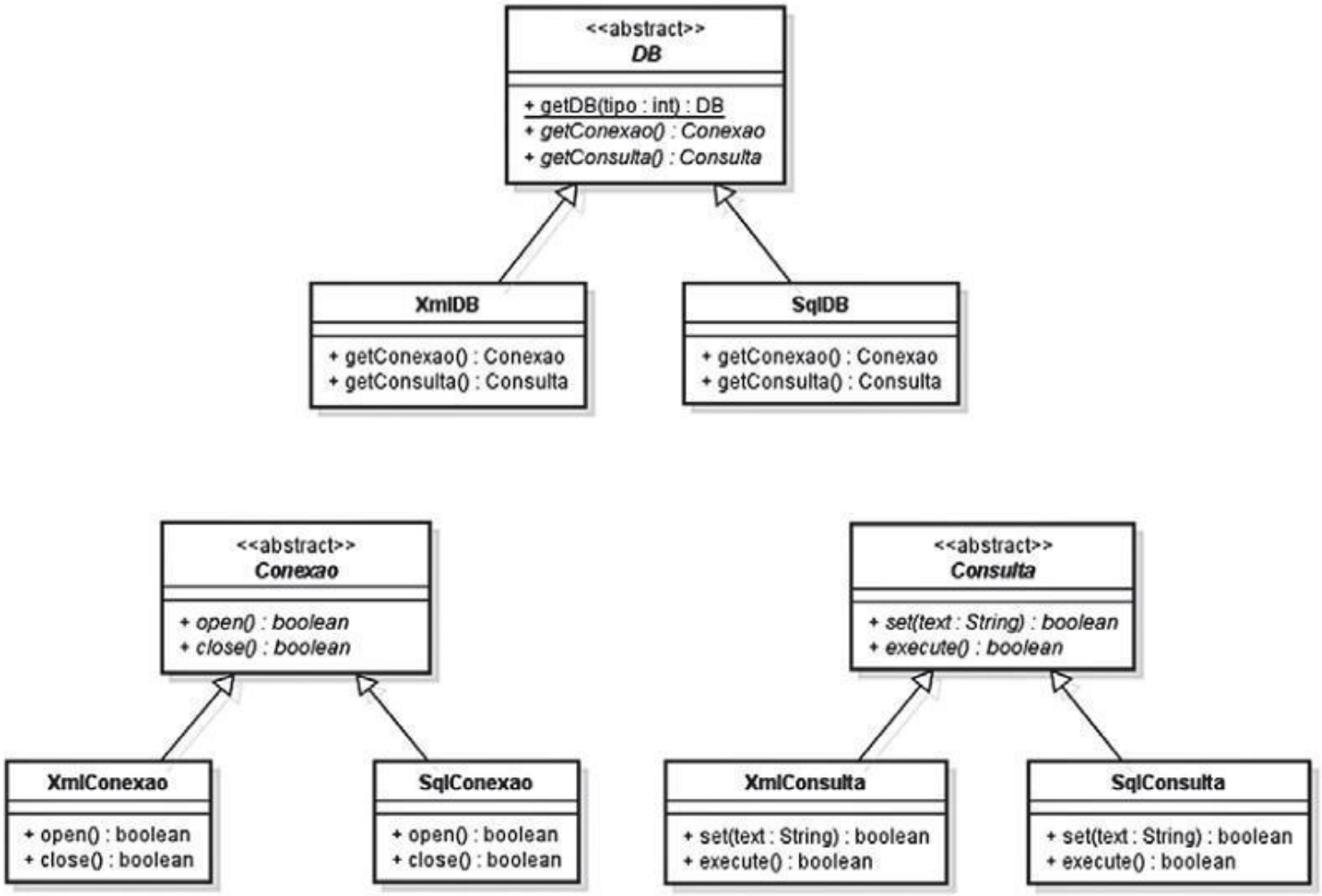
A solução proposta é a seguinte:

1. Uma classe abstrata (DB) será responsável por instanciar um objeto correspondente ao tipo de banco de dados desejado. Isso será feito através do método estático getDB(), que irá retornar um objeto de uma de suas subclasses concretas, de acordo com o parâmetro (tipo) passado.

2. O objeto criado no passo anterior irá instanciar conexões e consultas correspondentes ao tipo de DB criado; isto é, caso um DB XML tenha sido instanciado, apenas consultas XML e conexões XML serão instanciadas; caso um DB SQL tenha sido instanciado, apenas consultas SQL e conexões SQL serão instanciadas.

O diagrama de classe a seguir ilustra a estrutura descrita acima.

17 – CESGRANRIO – PETROBRAS - 2013



17 – CESGRANRIO – PETROBRAS - 2013

Qual padrão de projeto foi utilizado na solução proposta?

a) Adapter

b) Builder

c) Composite

d) Abstract Factory

e) Chain of Responsibility

17 – CESGRANRIO – PETROBRAS - 2013

Qual padrão de projeto foi utilizado na solução proposta?

a) Adapter

b) Builder

c) Composite

d) Abstract Factory

e) Chain of Responsibility

18 – CESGRANRIO – TRANSPETRO - 2011

Em uma aplicação para gerenciamento de currículos profissionais, deve haver uma funcionalidade para pesquisa (busca) por informações previamente registradas. Essa funcionalidade deve ser apresentada aos usuários como um formulário da interface gráfica da aplicação cuja composição deve ser diferente em cada uma das seguintes situações. Na primeira, o formulário deve apresentar ao usuário campos que permitam realizar buscas por empresas que oferecem vagas de emprego. Na segunda situação, o formulário deve apresentar campos para permitir aos usuários a realização de buscas por currículos de candidatos a vagas de emprego. O engenheiro de software, encarregado da implementação dessa funcionalidade, decidiu usar um padrão de projeto do catálogo GoF (Gang of Four). Esse padrão foi escolhido porque permite construir produtos com diferentes representações de tal forma que o mesmo processo de construção possa ser usado para criar diferentes produtos. No caso da funcionalidade descrita acima, os diferentes produtos a serem criados são as duas variantes do formulário de busca.

Dentre os listados abaixo, qual foi o padrão escolhido pelo engenheiro de software?

a) Abstract Factory

b) Bridge

c) Builder

d) Mediator

e) Prototype

18 – CESGRANRIO – TRANSPETRO - 2011

Em uma aplicação para gerenciamento de currículos profissionais, deve haver uma funcionalidade para pesquisa (busca) por informações previamente registradas. Essa funcionalidade deve ser apresentada aos usuários como um formulário da interface gráfica da aplicação cuja composição deve ser diferente em cada uma das seguintes situações. Na primeira, o formulário deve apresentar ao usuário campos que permitam realizar buscas por empresas que oferecem vagas de emprego. Na segunda situação, o formulário deve apresentar campos para permitir aos usuários a realização de buscas por currículos de candidatos a vagas de emprego. O engenheiro de software, encarregado da implementação dessa funcionalidade, decidiu usar um padrão de projeto do catálogo GoF (Gang of Four). Esse padrão foi escolhido porque permite construir produtos com diferentes representações de tal forma que o mesmo processo de construção possa ser usado para criar diferentes produtos. No caso da funcionalidade descrita acima, os diferentes produtos a serem criados são as duas variantes do formulário de busca.

Dentre os listados abaixo, qual foi o padrão escolhido pelo engenheiro de software?

a) Abstract Factory
b) Bridge
c) Builder
d) Mediator
e) Prototype

19 – CESGRANRIO – BNDES - 2013

O padrão de projeto Proxy é uma solução adequada para alguns problemas de design, **EXCETO**:

a) controlar o acesso a um objeto que necessite de permissão para tal.

b) fornecer um representante local para um objeto que se encontra em outro espaço de endereçamento.

c) fornecer uma interface mais adequada do que a oferecida pelo objeto que ele representa.

d) criar um substituto para um objeto cuja instanciação seja custosa ou demorada.

e) carregar um objeto persistente em memória quando ele for referenciado pela primeira vez.

19 – CESGRANRIO – BNDES - 2013

O padrão de projeto Proxy é uma solução adequada para alguns problemas de design, **EXCETO**:

a) controlar o acesso a um objeto que necessite de permissão para tal.

b) fornecer um representante local para um objeto que se encontra em outro espaço de endereçamento.


c) fornecer uma interface mais adequada do que a oferecida pelo objeto que ele representa.

d) criar um substituto para um objeto cuja instanciação seja custosa ou demorada.

e) carregar um objeto persistente em memória quando ele for referenciado pela primeira vez.

20 – CESGRANRIO – PETROBRÁS - 2011

O Padrão de Projeto Decorador é formado por uma hierarquia de classes cuja classe mais genérica representa um componente ou um componente abstrato. Para o padrão ser útil, essa classe deve ser diretamente especializada em, pelo menos, outras duas classes que representam um(a)

a) componente cliente e um componente adaptador

b) componente cliente e um decorador

c) componente concreto e um decorador

d) componente concreto e um componente adaptador

e) instância única e um decorador

20 – CESGRANRIO – PETROBRÁS - 2011

O Padrão de Projeto Decorador é formado por uma hierarquia de classes cuja classe mais genérica representa um componente ou um componente abstrato. Para o padrão ser útil, essa classe deve ser diretamente especializada em, pelo menos, outras duas classes que representam um(a)

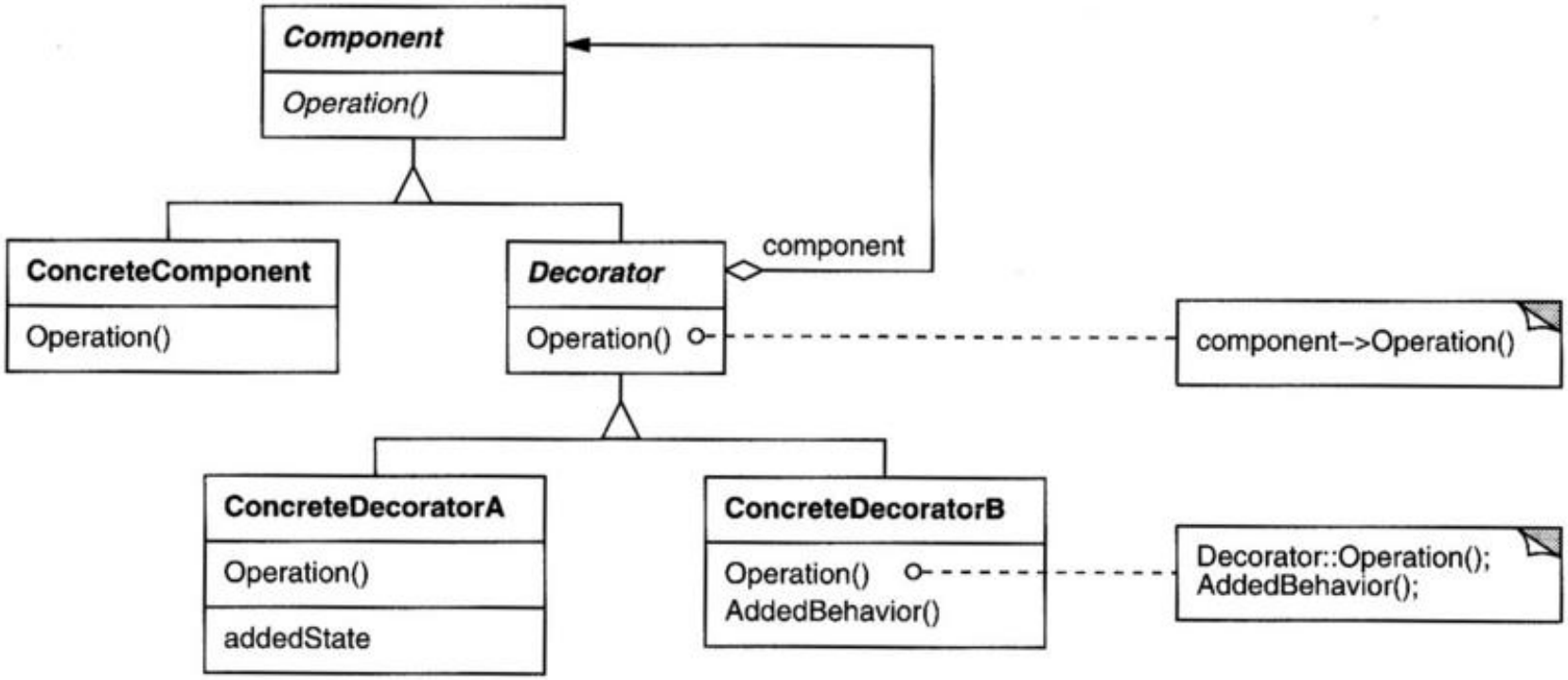
a) componente cliente e um componente adaptador

b) componente cliente e um decorador

c) componente concreto e um decorador

d) componente concreto e um componente adaptador

e) instância única e um decorador


Component
Operation()
ConcreteComponent
Operation()
Decorator
Operation()
component
component->Operation()
ConcreteDecoratorA
Operation()
addedState
ConcreteDecoratorB
Operation()
AddedBehavior()
Decorator::Operation();
AddedBehavior();

Gabarito – Padrões GoF

1- A	8- E	15-C
2- E	9- D	16-B
3- C	10- D	17- D
4- A	11- E	18- C
5- C	12- D	19- C
6- E	13- D	20- C
7- D	14- C	

Coesão e Acoplamento

CESGRANRIO

FUNDAÇÃO CESGRANRIO

21 – CESGRANRIO – BNDES - 2011

O acoplamento de conteúdo acontece quando, entre dois módulos, um referencia o interior do outro. Esses dois módulos podem estar acoplados por mais de uma forma.

Desse modo, o acoplamento desses módulos será definido pela(o)

a) coesão existente entre esses módulos

b) portabilidade das atividades de um módulo para outro

c) troca de mensagens de suas funcionalidades

d) pior tipo de acoplamento por eles apresentado

e) tamanho das instruções comuns ao acoplamento

Coesão e Acoplamento

- Príncipios de Engenharia de Software muito utilizados;

- Acoplamento:
  - Significa o quanto uma classe depende da outra para funcionar;
  - A e B estão acoplados quando B precisa alterar seu comportamento devido uma mudança de A;
- Coesão:
  - Está ligado ao princípio da responsabilidade única, aonde uma classe deve fazer exatamente aquilo que propõe, sem assumir responsabilidades que não são suas;

Coesão (pior para melhor)

Pior

Melhor

- Coincidental – Elementos não estão relacionados
- Lógica – Elementos possuem atividades semelhantes
- Temporal – Operações relacionadas apenas no tempo
- Procedural - Funções pouco relacionadas
- Comunicacional - Funções com mesma entrada/saída
- Sequencial – Saída de uma função é entrada para outra
- Funcional - Responsabilidade bem definida

- COincidentemente, LOrraine é do TiPo que CaSA Fumando

Acoplamento (pior para melhor)

Pior

Melhor

- Conteúdo – um referencia o conteúdo do outro
- Comum – referenciam a mesma área de dados
- Controle – um controla a lógica do outro
- Imagem – referenciam a mesma estrutura de dados
- Dados – comunicação por parâmetros

- Conteúdo comum controla a imagem de dados

21 – CESGRANRIO – BNDES - 2011

O acoplamento de conteúdo acontece quando, entre dois módulos, um referencia o interior do outro. Esses dois módulos podem estar acoplados por mais de uma forma.

Desse modo, o acoplamento desses módulos será definido pela(o)

a) coesão existente entre esses módulos

b) portabilidade das atividades de um módulo para outro

c) troca de mensagens de suas funcionalidades

d) pior tipo de acoplamento por eles apresentado

e) tamanho das instruções comuns ao acoplamento

21 – CESGRANRIO – BNDES - 2011

O acoplamento de conteúdo acontece quando, entre dois módulos, um referencia o interior do outro. Esses dois módulos podem estar acoplados por mais de uma forma.

Desse modo, o acoplamento desses módulos será definido pela(o)

a) coesão existente entre esses módulos

b) portabilidade das atividades de um módulo para outro

c) troca de mensagens de suas funcionalidades

d) pior tipo de acoplamento por eles apresentado

e) tamanho das instruções comuns ao acoplamento

22 – CESGRANRIO – BNDES - 2011

Pode-se avaliar a modularidade de um sistema de software pelo critério de coesão de seus módulos.

O módulo que contém elementos que contribuem para a execução de uma e somente uma atividade relacionada ao problema constitui um módulo com coesão

a) comunicacional
b) funcional
c) procedural
d) sequencial
e) temporal

22 – CESGRANRIO – BNDES - 2011

Pode-se avaliar a modularidade de um sistema de software pelo critério de coesão de seus módulos.

O módulo que contém elementos que contribuem para a execução de uma e somente uma atividade relacionada ao problema constitui um módulo com coesão

a) comunicacional

b) funcional

c) procedural

d) sequencial

e) temporal

23 – CESGRANRIO – DCEA - 2006

Em relação a projeto estruturado de sistemas, os níveis de coesão, ordenados do melhor para o pior, são:

a) comunicacional, funcional, seqüencial, procedural, lógica, temporal, coincidental.

b) seqüencial, procedural, coincidental, lógica, funcional, temporal, comunicacional.

c) lógica, seqüencial, funcional, comunicacional, temporal, coincidental, procedural.

d) coincidental, lógica, comunicacional, seqüencial, procedural, temporal, funcional.

e) funcional, seqüencial, comunicacional, procedural, temporal, lógica, coincidental.

23 – CESGRANRIO – DCEA - 2006

Em relação a projeto estruturado de sistemas, os níveis de coesão, ordenados do melhor para o pior, são:

a) comunicacional, funcional, seqüencial, procedural, lógica, temporal, coincidental.

b) seqüencial, procedural, coincidental, lógica, funcional, temporal, comunicacional.

c) lógica, seqüencial, funcional, comunicacional, temporal, coincidental, procedural.

d) coincidental, lógica, comunicacional, seqüencial, procedural, temporal, funcional.

e) funcional, seqüencial, comunicacional, procedural, temporal, lógica, coincidental.

24 – CESGRANRIO – BNDES - 2011

Determinado grupo de pesquisa de uma universidade, no processo de criação de uma linguagem de programação, estabelece que erros de tipo sempre devem ser detectados.

Essa é uma característica conhecida como

a) acoplamento fraco
b) acoplamento forte
c) tipificação fraca
d) tipificação forte
e) tolerância a falhas

24 – CESGRANRIO – BNDES - 2011

Determinado grupo de pesquisa de uma universidade, no processo de criação de uma linguagem de programação, estabelece que erros de tipo sempre devem ser detectados.

Essa é uma característica conhecida como

a) acoplamento fraco
b) acoplamento forte
c) tipificação fraca
d) tipificação forte
e) tolerância a falhas

25 – CESGRANRIO – PETROBRÁS - 2010

Tanto no paradigma orientado a objetos quanto no paradigma estruturado, existem diversas técnicas úteis para averiguar se um sistema de software foi bem projetado. No primeiro, essas medidas são aplicáveis a classes, e no segundo, são aplicáveis a módulos. Quais, dentre os termos apresentados a seguir, são medidas de qualidade de projeto aplicáveis em ambos os paradigmas?

a) Fan-in, fan-out e herança.

b) Encapsulamento, herança e coesão.

c) Coesão, acoplamento e polimorfismo.

d) Fan-in, fan-out e acoplamento.

e) Coesão, acoplamento e polimorfismo.

25 – CESGRANRIO – PETROBRÁS - 2010

Tanto no paradigma orientado a objetos quanto no paradigma estruturado, existem diversas técnicas úteis para averiguar se um sistema de software foi bem projetado. No primeiro, essas medidas são aplicáveis a classes, e no segundo, são aplicáveis a módulos. Quais, dentre os termos apresentados a seguir, são medidas de qualidade de projeto aplicáveis em ambos os paradigmas?

a) Fan-in, fan-out e herança.

b) Encapsulamento, herança e coesão.

c) Coesão, acoplamento e polimorfismo.

d) Fan-in, fan-out e acoplamento.

e) Coesão, acoplamento e polimorfismo.

Gabarito – Coesão e Acoplamento

21- D
22- B
23- E
24- D
25- D

Refatoração e Padrões Arquiteturais

CESGRANRIO

FUNDAÇÃO CESGRANRIO

01 – CESGRANRIO – BACEN - 2009

Um programador inexperiente gerou, em determinado sistema, código-fonte de baixa qualidade que foi analisado e reprovado auditoria de código.

Considerando-se que existem muitos códigos duplicados nesse sistema, que técnicas de refatoração são indicadas nessa situação?

a) Introduce Parameter Object e Move Method.

b) Duplicate Observed Data e Replace Inheritance with Delegation.

c) Move Method e Move Field.

d) Extract Method e Substitute Algorithm.

e) Remove Double Negative e Introduce Parameter Object.

01 – CESGRANRIO – BACEN - 2009

Um programador inexperiente gerou, em determinado sistema, código-fonte de baixa qualidade que foi analisado e reprovado auditoria de código.

Considerando-se que existem muitos códigos duplicados nesse sistema, que técnicas de refatoração são indicadas nessa situação?

a) Introduce Parameter Object e Move Method.

b) Duplicate Observed Data e Replace Inheritance with Delegation.

c) Move Method e Move Field.

d) Extract Method e Substitute Algorithm.

e) Remove Double Negative e Introduce Parameter Object.

02 – CESGRANRIO – BNDES - 2009

No âmbito da Refatoração de software, analise as afirmativas a seguir.

I - Introduce Parameter Object é uma técnica de refatoração que pode ser aplicada em métodos muito longos.

II - Código duplicado é candidato para aplicação das técnicas Extract Method e Substitute Algorithm.

III - Extract Class é uma técnica de refatoração que pode ser aplicada em classes muito grandes.

Está(ão) correta(s) a(s) afirmativa(s)

a) I, apenas.

b) II, apenas.

c) III, apenas.

d) I e II, apenas.

e) I, II e III.

02 – CESGRANRIO – BNDES - 2009

No âmbito da Refatoração de software, analise as afirmativas a seguir.

I - Introduce Parameter Object é uma técnica de refatoração que pode ser aplicada em métodos muito longos.

II - Código duplicado é candidato para aplicação das técnicas Extract Method e Substitute Algorithm.

III - Extract Class é uma técnica de refatoração que pode ser aplicada em classes muito grandes.

Está(ão) correta(s) a(s) afirmativa(s)

a) I, apenas.

b) II, apenas.

c) III, apenas.

d) I e II, apenas.

e) I, II e III.

03 – CESGRANRIO – BNDES - 2009

O principal sistema de informação de uma empresa, desenvolvido internamente, contém regras de negócio no código da interface de usuário. Qual a técnica de refatoração que o arquiteto de software pode indicar para melhorar consideravelmente o sistema nesse caso?

a) Tease Apart Inheritance

b) Separate Domain from Presentation

c) Extract Hierarchy

d) Convert Procedural Design to Objects

e) Apply Table Data Gateway Pattern

03 – CESGRANRIO – BNDES - 2009

O principal sistema de informação de uma empresa, desenvolvido internamente, contém regras de negócio no código da interface de usuário. Qual a técnica de refatoração que o arquiteto de software pode indicar para melhorar consideravelmente o sistema nesse caso?

a) Tease Apart Inheritance

b) Separate Domain from Presentation

c) Extract Hierarchy

d) Convert Procedural Design to Objects

e) Apply Table Data Gateway Pattern

Domain Logic Patterns

Transaction Script	Organiza toda a lógica como um procedimento único
Domain Model	Cria objetos interconectados onde cada objeto representa um significado individual do modelo
Table Module	Organiza a lógica de domínio com uma classe por tabela
Service Layer	Define uma fronteira para a aplicação e os serviços oferecidos

Data Source Architectural Patterns

Table Data Gateway	Concentra todas as sqls de uma determinada tabela
Row Data Gateway	Disponibiliza objetos que funcionam exatamente como um registro da estrutura de dados acessada
Active Record	Coloca a lógica de negócio em um objeto de domínio
Data Mapper	Camada de software que separa objetos da memória em objetos da base de dados

Object Relational Structural Patterns

Identity Field	Armazenar o valor da chave primária em um campo do objeto
Foreign Key Mapping	Mapeia um objeto que referencia a chave estrangeira na base de dados
Association Table Mapping	Mapeia campos multivalorados
Dependent Mapping	Mapeia objetos dependentes
Embedded Value	Mapeia o valor de um objeto em registros do proprietario do objeto
Serialized LOB	Forma para persistir objetos grandes
Single Table Inheritance	Mapeia todos os campos das classes de uma estrutura de herança
Class Table Inheritance	Utiliza um objeto por tabela em uma estrutura de herança
Concrete Table Inheritance	Mapear objetos oriundos de herança em bancos relacionais
Inheritance Mappers	Mapear herança oo em memória para bancos relacionais minimizando código

Object Relational Metadata Mapping PAtterns

Metadata Mapping	Permite definir mapeamentos em forma de tabela simples que possa ser processada por código genérico
Query Object	Uma estrutura de objetos que podem formar-se em uma consulta SQL
Repository	Faz a fronteira entre camadas de domínio e mapeamento de dados, agindo como uma coleção de objetos em memória

Web Presentation Patterns

Model View Controller	Possui um forte papel na influencia da maioria das estruturas de interface do usuário e no design da interface.
Page Controller	Um objeto que corresponde ao controlador de uma página
Font Controller	Consolida toda a manipulação de pedidos através de um único objeto manipulador.
Template View	Possibilita que a parte estática da página funciona como um modelo para uma resposta específica
Transform View	Dados do modelo funcionam como entrada e o html como saída
Two-Step View	Lida com o problema de separar páginas diferentes em uma divisão em duas partes
Application Controller	Coloca todo a lógica de fluxo em um controlador da aplicação

Distribution Patterns

Remote Facade	Traduz métodos de granulação grossa para objetos de granulação fina
Data Transfer Object	Um objeto que contém todos os dados para a chamada de uma conexão

Offline Concurrency Patterns

Optimistic Offline Lock	Assume que a chance de um conflito de sessão é baixa, permitindo que vários usuários trabalhem com os mesmos dados ao mesmo tempo
Pessimistic Offline Lock	Previne conflitos evitando que seja necessário adquirir lock para acessar uma informação
Coarse Grained Lock	Lock único que cobre vários objetos
Implicit Lock	Trata o lock de forma implícita

Session State Patterns

Client Session State	Armazena o estado da sessão no cliente
Server Session State	Mantém o estado da sessão em um servidor
Database Session State	Armazena dados da sessão no banco de dados

Base Patterns

Gateway	Encapsula acesso para um sistema ou recurso externo
Mapper	Estabelece uma comunicação entre dois objetos independentes
Layer Supertype	Um tipo que age como um supertipo para todos os tipos em sua camada
Separated Interface	Define uma interface em um pacote separado a partir de sua implementação
Registry	Um objeto conhecido que todos os outros podem utilizar para encontrar objetos e serviços comuns
Value Object	Um pequeno objeto como intervalo de datas
Money	Representa valor monetário
Special Case	Subclasse que prove comportamento especial para casos específicos
Plugin	Liga classes durante a configuração ao invés da compilação
Service Stub	Remove dependência de serviços durante testes (WSDL)
Record Set	Representação em memória de dados tabulares

04 – CESGRANRIO – BNDES - 2009

Sobre o padrão arquitetural Gateway de linha de dados (Row Data Gateway), é **INCORRETO afirmar que**

a) é restrito a linguagens de programação que suportam herança múltipla.

b) mantém os valores de suas propriedades ao longo do seu ciclo de vida.

c) contém uma instância por registro da tabela de banco de dados.

d) interage bem com o padrão Roteiro de Transação (Transaction Script).

e) pode fazer conversões de tipos de dados de inteiro para string.

04 – CESGRANRIO – BNDES - 2009

Sobre o padrão arquitetural Gateway de linha de dados (Row Data Gateway), é **INCORRETO afirmar que**

a) é restrito a linguagens de programação que suportam herança múltipla.

b) mantém os valores de suas propriedades ao longo do seu ciclo de vida.

c) contém uma instância por registro da tabela de banco de dados.

d) interage bem com o padrão Roteiro de Transação (Transaction Script).

e) pode fazer conversões de tipos de dados de inteiro para string.

05 – CESGRANRIO – BNDES - 2009

No âmbito de padrões de arquitetura de aplicações corporativas, assinale a opção que NÃO apresenta uma característica do padrão “Table Module”

a) É ideal para trabalhar com record sets.

b) Constitui uma instância para todos os registros da tabela.

c) Constitui ótimo suporte ao polimorfismo.

d) Organiza a lógica de domínio com uma classe para cada tabela que contém todos os procedimentos com a lógica.

e) Os objetos que residem na camada de negócios são praticamente mapeados diretamente de tabelas de banco de dados.

05 – CESGRANRIO – BNDES - 2009

No âmbito de padrões de arquitetura de aplicações corporativas, assinale a opção que NÃO apresenta uma característica do padrão "Table Module"

a) É ideal para trabalhar com record sets.

b) Constitui uma instância para todos os registros da tabela.

c) Constitui ótimo suporte ao polimorfismo.

d) Organiza a lógica de domínio com uma classe para cada tabela que contém todos os procedimentos com a lógica.

e) Os objetos que residem na camada de negócios são praticamente mapeados diretamente de tabelas de banco de dados.

Gabarito – Padrões Arquiteturais

1- E
2- C
3- C
4- A
5- C