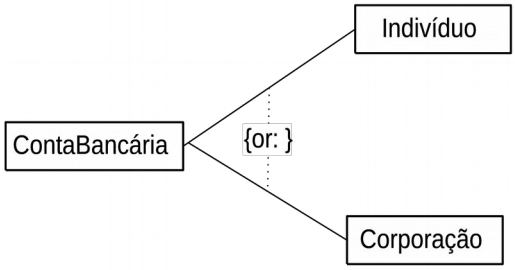
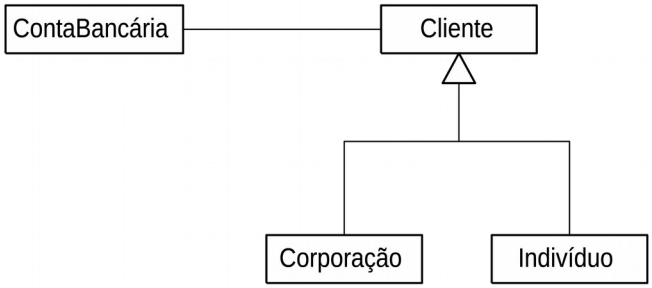
1) Considere os diagramas de classes de análise fornecidos nos itens (a) e (b) abaixo, ambos de acordo com a notação da UML. Esses diagramas desejam representar o fato de que uma conta bancária pode estar associada a uma pessoa, que pode ser ou uma pessoa física (representada pela classe Indivíduo), ou uma pessoa jurídica (representada pala classe Corporação). Uma dessas duas soluções é melhor que a outra? Se sim, qual delas e em que sentido? Justifique sua resposta considerando alguns dos padrões GRASP.

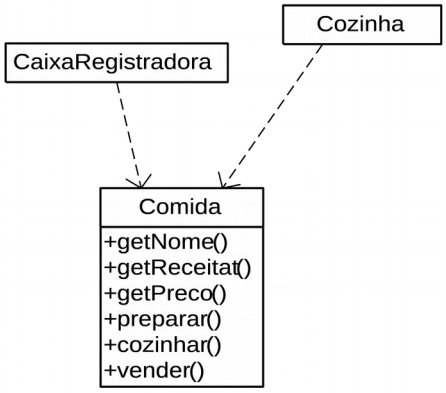


(a)

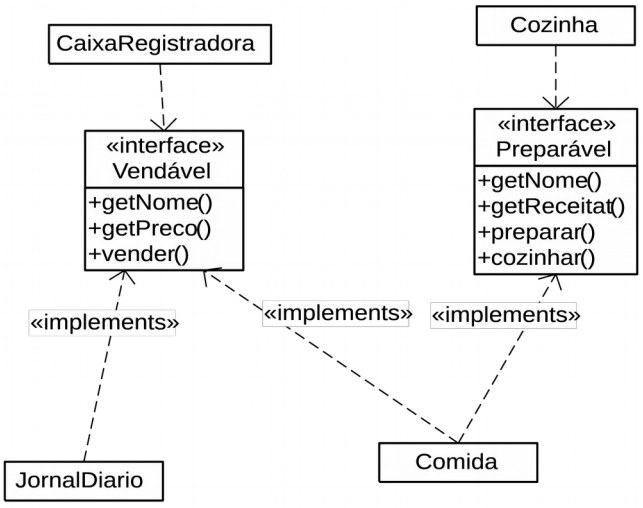


(b)

2) Considere uma aplicação para um bar-café. Nessa aplicação, considere a existência de uma classe que representa um comestível qualquer vendido pelo bar-café: Comida. Considere ainda duas outras classes nessa aplicação, Cozinha e CaixaRegistradora. A classe cozinha manipula objeto da classe Comida para montar pratos. Já a classe CaixaRegistradora manipula objeto comida para registrar a venda dos mesmos e cobrar por eles. Portanto, essas duas classes dependem dos serviços fornecidos pela classe Comida. Em um primeiro modelo dessa aplicação, o modelador fez com que as classes Cozinha e CaixaRegistradora dependessem diretamente da classe Comida, conforme a Figura 2a. No entanto, conforme o desenvolvimento foi se evoluindo, o modelador identificou um novo requisito na aplicação: agora era preciso registrar a venda de coisas não comestíveis. Por exemplo: o café-bar passou a vender jornais diários. Para atender ao novo requisito, o modelador criou duas novas interfaces, Vendável e Comestível, conforme está na Figura 2b. Discuta detalhadamente a decisão de projeto do modelador. Você achou a decisão adequada? Que princípios de projeto levaram o modelador a tomar tal decisão?



(2a)



(2b)

3) Os principais ambientes de desenvolvimento integrado (integrated development environment, IDE) modernos têm a capacidade de fornecer ao desenvolvedor o recurso da compilação incremental. Nesse recurso, à medida que o programador codifica sua aplicação, a IDE se encarrega de apontar erros de sintaxe nas linhas de código. Isso permite que esse programador corrija possíveis erros de sintaxe gradativamente, sem precisar identificá-los e corrigi-los apenas após a compilação explícita do código fonte. Para prover esse recurso, a IDE deve tratar pequenos ou grandes trechos de um programa como unidades compiláveis. Dessa forma, uma linha de instrução, uma classe inteira (com diversos métodos) e uma aplicação (com diversas classes) devem ser consideradas indistintamente para fins de compilação pela IDE. O NetBeans (Sun), o Eclipse (Open Source) e o Visual Studio (Microsoft) são exemplos de IDE's que dão fornecem esse recurso. Se você fosse o implementador desse recurso em uma dessas IDE, que padrão de projeto GoF poderia ser utilizado para facilitar sua tarefa? Explique sua resposta, fornecendo um diagrama de classes que resume a estrutura de classes de sua solução.

4) Você está desenvolvendo uma aplicação para uma empresa que vende componentes de computador. Atualmente, você está construindo uma hierarquia de classes para representar os diferentes tipos de componentes (você nomeou essas classes como Processador, DiscoRígido e CDROM). Essa hierarquia contém também uma superclasse abstrata denominada Componente, da qual são derivadas as demais classes. Por outro lado, um amigo seu já desenvolveu um sistema semelhante e lhe passou classes para representar discos rígidos e CPUs (as classes dele são denominadas HardDisk e CPU respectivamente). De que maneira você pode construir sua hierarquia de classes aproveitando (reutilizando) as classes fornecidas pelo seu amigo, sem precisar modificá-las? Que padrão de software poderia ser empregado para auxiliar na definição da estrutura de classes para representar a situação discutida acima? Utilizando esse padrão, forneça um fragmento de diagrama de classes que represente a situação acima.

5) Imagine uma classe que representa imagens, Imagem. Imagine um componente que apresenta uma imagem. Então esse componente recebe uma instância de uma classe que é subclasse de Imagem. Mas a classe Imagem é uma classe abstrata e de acordo com o tipo de imagem que o usuário selecionar para visualizar, a subclasse ImagemJPEG ou ImagemGIF será instanciada. Sendo assim, o objeto Imagem será instanciado a partir da classe necessária para tratar o tipo de imagem que for selecionada pelo usuário. Que padrão de projeto pode ser utilizado nessa situação? Esboce a solução com o uso desse padrão.

6) (TRT-15ª Região/2015) Os padrões de projeto tornam mais fácil reutilizar projetos e arquiteturas bem sucedidas. Atualmente existem diversos padrões de projetos conforme abaixo:

I. Fornece uma interface para a criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.

II. Converte a interface de uma classe em outra interface esperada pelos clientes permitindo que certas classes trabalhem em conjunto, pois de outra forma, seria impossível por causa de suas interfaces incompatíveis.

III. Fornece uma maneira de acessar sequencialmente os elementos de uma agregação de objetos sem expor sua representação subjacente.

Os padrões de projeto apresentados em I, II e III são, respectivamente:

a) Façade, Builder e Mediator.

b) Abstract Factory, Adapter e Iterator.

c) Façade, Adapter e Interpreter.

d) Singleton, Builder e Mediator.

e) Abstract Factory, Prototype e Iterator.

7) (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.

8) (Petrobras/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.

9) (INMETRO/2015) Em padrões de projeto, delegação é uma maneira de tornar a composição tão poderosa para fins de reutilização quanto à herança, sendo que dois objetos são envolvidos no tratamento de uma solicitação. É uma boa escolha de projeto somente quando ela simplifica mais o que complica. Ao definir quais padrões deverão ser utilizados no projeto, considerando que diversos padrões de projeto usam delegação, mas três padrões dependem dela. Assinale-os.

a) State, Strategy e Visitor.

b) Adapter, Bridge e Composite.

c) Builder, Prototype e Singleton.

d) Façade, Command e Decorator.

e) Factory Method, Interpreter e Template Method.

10) (INMETRO/2015) De acordo com o padrão orientado a objeto, é necessário determinar um padrão de projeto a ser utilizado em certa situação. O padrão escolhido foi o Iterator. Cada padrão tem uma intenção para o qual foi desenvolvido e/ou criado.

Assinale, a seguir, a intenção do

a) Garantir que uma classe tenha somente uma instância e fornecer um ponto global de acesso para ela.

b) Fornecer um objeto representado, ou um marcador de outro objeto, para controlar o acesso ao mesmo.

c) Permitir que um objeto altere seu comportamento quando seu estado interno muda. O objeto parecerá ter mudado de classe.

d) Especificar os tipos de objetos a serem criados usando uma instância prototípica e criar novos objetos copiando esse protótipo.

e) Fornecer uma maneira de acessar, sequencialmente, os elementos de uma agregação de objetos sem expor a sua representação subjacente.

11) (INMETRO/2015) Um projeto de software orientado a objetos não é algo muito fácil. Mas, projetar software reutilizável, orientado a objetos, é ainda mais complicado. Muitas ações devem ser realizadas como: identificar objetos, separá-los em classes, definir interfaces, entre outros. Normalmente, o projeto deve ser específico para aquele problema que se quer resolver, mas também genérico o suficiente para atender problemas e requisitos futuros. Os padrões de projeto tornam mais fácil a reutilização de projetos e arquiteturas bem sucedidas. Também ajudam a escolher alternativas de projeto que tornam um sistema reutilizável e a evitar alternativas que comprometam a reutilização. Os padrões de projeto podem ser classificados em: de criação, estruturais e comportamentais. Assinale, a seguir, um padrão de projeto da classe estrutural.

a) State.

b) Adapter.

c) Mediator.

d) Memento.

e) Chain of Responsability.

12) (MDA/2014) Em relação aos padrões de projeto, são exemplos de padrão de criação, padrão estrutural e padrão comportamental, respectivamente:

a) Abstract factory, bridge e observer.

b) Mediator, composite e facade.

c) Decorator, singleton ememento.

d) Prototype, adapter e composite.

e) Visitor, decorator e builter.

13) (CNMP/2015) Um Analista de Desenvolvimento de Sistemas do CNMP deve indicar o padrão de projeto mais adequado para ser aplicado na seguinte situação:

Uma aplicação que existe simultaneamente em um dispositivo móvel e no ambiente corporativo, necessita de um processo de sincronização entre as informações processadas no dispositivo móvel e na base corporativa. Ambas as aplicações devem se comunicar com um objeto que deve ser único para processar este sincronismo, a fim de evitar a possibilidade de criar dados na base.

O padrão de projeto corretamente indicado pelo Analista deve ser:

a) Factory Method, um padrão de criação, que busca definir o fluxo de um algoritmo em uma operação, postergando (deferring) alguns passos para subclasses, sem mudar a estrutura do mesmo.

b) Prototype, um padrão estrutural, que busca fornecer uma interface para criação de famílias de objetos relacionados ou dependentes sem especificar suas classes concretas.

c) Singleton, um padrão de criação, que busca garantir que um objeto terá apenas uma única instância, ou seja, uma classe irá gerar apenas um objeto e que este estará disponível de forma única para todo o escopo de uma aplicação.

d) Command, um padrão comportamental, que busca definir o fluxo de um algoritmo em uma operação, postergando (deferring) alguns passos para subclasses, sem mudar a estrutura do mesmo.

e) Façade, um padrão estrutural, que busca garantir que um objeto terá apenas uma única instância, ou seja, uma classe irá gerar apenas um objeto e que este estará disponível de forma única para todo o escopo de uma aplicação.

14) (SERPRO/2014) Os padrões GoF refletem situações muito recorrentes em projetos orientados a objetos. Esses padrões são classificados em três famílias: padrões de criação, padrões estruturais e padrões comportamentais. Considere os objetivos principais de alguns desses padrões, tais como:

I. produzir objetos utilizando uma estrutura de árvore para representar hierarquias de todo-parte, de forma a permitir que objetos do tipo todo ou do tipo parte sejam tratados da mesma maneira.

II. atribuir responsabilidades adicionais a um objeto de forma dinâmica, para atender a algumas situações em que seja desejado que um objeto tenha mais responsabilidades que os demais da sua classe.

III. prover uma interface única para um conjunto de interfaces de um subsistema, facilitando o seu uso, para atender a situações em que um conjunto de classes deve se comportar como um componente.

Os padrões cujos objetivos foram descritos em I, II e III são, respectivamente:

a) Abstract Factory, Prototype e Singleton, da família de padrões de criação.

b) Composite, Decorator e Facade, da família de padrões estruturais.

c) Template Method, State e Mediator, da família de padrões comportamentais.

d) Composite, Decorator e Facade, da família de padrões de criação.

e) Abstract Factory, Prototype e Singleton, da família de padrões estruturais.