EXPLICADORES.NET

2022/2023

EAGS SIN / CAP

POO

Prof. Camila Dias

PROGRAMAÇÃO ORIENTADA A OBJETO

Bibliografia e conteúdo

SINTES, Anthony.Aprenda Programação Orientada a Objeto em 21 Dias.
São Paulo:
Makron Books, 2002.

PROGRAMAÇÃO ORIENTADA A OBJETOS

Introdução à programação orientada a objetos.
Encapsulamento.
Método.
Propriedades.
Construtores.
Herança.
Polimorfismo.
Introdução à UML.
Introdução à Análise Orientada a Objetos.
Introdução ao Projeto Orientado a Objetos.
Reutilizando projetos através de padrões de projeto.
Padrões avançados de projeto.
OO e programação de interface com o usuário.
Construindo software confiável através de testes.
Prática da orientação a objetos.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

A interface com o usuário (UI) fornece a interface entre o usuário e seu sistema. Quase todo sistema moderno terá alguma forma de UI, seja gráfica, dirigida pela linha de comando ou mesmo baseada em telefone ou fala. (Alguns sistemas podem combinar todos os quatro tipos!). Em qualquer caso, você precisa tomar um cuidado especial no projeto e implementação de suas interfaces com o usuário. Felizmente, a POO pode trazer as mesmas vantagens para sua UI que apresenta para outros aspectos do sistema.

Hoje você aprenderá:

- Como a POO e a construção da UI se relacionam
- Sobre a importância de uma UI desacoplada
- Quais padrões o ajudam a desacoplar a UI

POO e a interface com o usuário

Fundamentalmente, o processo de projetar e programar interfaces com o usuário não é diferente do processo de projetar e programar qualquer outro aspecto de seu sistema. Talvez você precise aprender algumas novas APIs para que possa construir suas UIs, mas no final, você precisa aplicar na UI os mesmos princípios orientados a objetos que aplicaria nas outras partes do seu sistema.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

A questão merece ênfase: ao projetar e programar suas interfaces com o usuário, você *deve* aplicar em suas UIs os mesmos princípios de OO que aplicaria no restante do seu sistema! Freqüentemente, as interfaces com o usuário são simplesmente reunidas e jogadas no sistema como uma cogitação posterior.

Em vez disso, o código de sua UI precisa ser tão orientado a objetos quanto o código do restante do sistema. A implementação da UI precisa usar encapsulamento, herança e polimorfismo corretamente.

Você também precisa considerar a UI enquanto realiza AOO e POO (Projeto Orientado a Objetos). Sem uma análise e projeto corretos, você pode perder certos requisitos e verificar que escreveu uma UI que não é suficientemente flexível para fornecer o nível desejado de funcionalidade ou uma UI que não pode se adaptar às alterações futuras.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

A importância das UIs desacopladas

Você verá que o mesmo sistema freqüentemente exige diversas interfaces diferentes, muitas vezes não relacionadas. Por exemplo, um sistema de abastecimento pode permitir que as pessoas façam pedidos pela Web, pelo telefone, através de PDA ou de um aplicativo local personalizado. Cada uma dessas interfaces se ligará ao mesmo sistema; entretanto, cada estratégia se ligará ao sistema e apresentará as informações de sua própria maneira.

Você também verá que os requisitos da interface com o usuário podem se tornar um alvo móvel. Os sistemas amadurecem com o tempo, quando novos recursos são adicionados e quando os usuários expõem áreas de debilidade. Em resposta, você precisará atualizar continuamente a interface com o usuário para poder expor cada novo recurso e corrigir todos os defeitos. Essa realidade pede uma interface com o usuário cujo projeto seja flexível e possa aceitar alterações prontamente.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

A melhor maneira de obter flexibilidade é projetando um sistema que seja completamente desacoplado de sua UI. Um projeto desacoplado permite que você adicione qualquer UI no sistema e faça alterações nas UIs existentes, sem ter de fazer alterações correspondentes no sistema em si. Um projeto desacoplado também permite testar os recursos do sistema, mesmo que você não tenha terminado de desenvolver a UI. Além disso, um projeto desacoplado permite que você aponte precisamente erros que são da UI ou que são do sistema.

Felizmente, a POO é a solução perfeita para esses problemas. Isolando as responsabilidades corretamente, você pode diminuir o impacto das alterações em partes não relacionadas do sistema. Isolando funcionalidade, você deve conseguir adicionar qualquer interface em seu sistema, a qualquer momento, sem fazer alterações no sistema subjacente. O segredo é não incorporar o código da UI dentro do próprio sistema. Os dois *devem* ser separados.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

`VisualBankAccount` fornece toda a funcionalidade da classe `BankAccount`, apresentada em lições anteriores. `VisualBankAccount` também sabe como se apresentar, como se vê na Figura 13.1.

FIGURA 13.1
`VisualBankAccount` *dentro de um frame.*

Quando você digitar um valor e clicar no botão Deposit ou Withdraw, a conta bancária extrairá o valor do campo de entrada e chamará seu método `withdrawFunds()` ou `depositFunds()`, com o valor.

Como `VisualBankAccount` é um `Jpanel`,você pode incorporá-lo em qualquer GUI Java. Infelizmente, a UI não está desacoplada da classe de conta bancária. Tal acoplamento forte torna impossível usar a conta bancária em outras formas de interfaces com o usuário ou para fornecer uma UI diferente, sem ter de alterar a própria classe `VisualBankAccount`. Na verdade, você precisará criar uma versão separada da classe para cada tipo de UI dentro da qual queira que ela participe.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

Como desacoplar a UI usando o padrão Model View Controller

O padrão de projeto MVC (Model View Controller) fornece uma estratégia para o projeto de interfaces com o usuário que desacoplam completamente o sistema subjacente da interface com o usuário.

NOTA

MVC é apenas uma estratégia para o projeto de interfaces com o usuário orientadas a objetos. Existem outras estratégias válidas para o projeto de interface com o usuário; entretanto, a MVC é uma estratégia testada que é popular no setor do software. Se você acabar realizando o trabalho da interface com o usuário, especialmente em relação à Web e ao J2EE da Sun, encontrará o MVC.

O Document/View Model, popularizado pelas Microsoft Foundation Classes e o padrão de projeto PAC (*Presentation Abstraction Control*), fornecem alternativas à MVC. Veja o livro *Pattern–Oriented Software Architecture A System of Patterns*, de Frank Buschmann et al, para uma apresentação completa dessas alternativas.

O padrão MVC desacopla a UI do sistema, dividindo o projeto da UI em três partes separadas:

- O modelo, que representa o sistema
- O modo de visualização, que exibe o modelo
- O controlador, que processa as entradas do usuário

Cada parte da tríade MVC tem seu conjunto próprio de responsabilidades exclusivas.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

O modelo

O modelo é responsável por fornecer:

- Acesso à funcionalidade básica do sistema
- Acesso às informações de estado do sistema
- Um sistema de notificação de mudança de estado

O modelo é a camada da tríade MVC que gerencia o comportamento básico e o estado do sistema. O modelo responde às consultas sobre seu estado a partir do modo de visualização e do controlador e aos pedidos de mudança de estado do controlador.

Um sistema pode ter muitos modelos diferentes. Por exemplo, um sistema de banco pode ser constituído de um modelo de conta e um modelo de caixa. Vários modelos pequenos repartem melhor a responsabilidade do que um único modelo grande.

Não deixe o termo *modelo* confundi-lo. Um modelo é apenas um objeto que representa o sistema.

O controlador é a camada da tríade MVC que interpreta a entrada do usuário. Em resposta à entrada do usuário, o controlador pode comandar o modelo ou o modo de visualização para que mude ou execute alguma ação.

O modo de visualização é a camada da tríade MVC que exibe a representação gráfica ou textual do modelo. O modo de visualização recupera todas as informações de estado a respeito do modelo, a partir do modelo.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

Em qualquer caso, o modelo não sabe absolutamente que um modo de visualização ou controlador está fazendo uma chamada de método. O modelo só sabe que um objeto está chamando um de seus métodos. A única conexão que um modelo mantém com a UI é através do sistema de notificação de mudança de estado.

Se um modo de visualização ou controlador estiver interessado na notificação de mudança de estado, ele se registrará no modelo. Quando o modelo mudar de estado, percorrerá sua lista de objetos registrados (freqüentemente chamados de receptores ou observadores) e informará cada objeto da mudança de estado. Para construir esse sistema de notificação, os modelos normalmente empregarão o padrão Observer.

O padrão Observer

O padrão `Observer` fornece um projeto para um mecanismo de publicação/assinatura entre objetos. O padrão `Observer` permite que um objeto (o observador) registre seu interesse em outro objeto (o observável). Quando o observável quiser notificar os seus observadores de uma alteração, ele chamará um método `update()` em cada observador.

A Listagem 13.2 define a interface `Observer`. Todos os observadores que quiserem se registrar com o objeto observável devem implementar a interface `Observer`.

LISTAGEM 13.2 *Observer.java*

```
public interface Observer {
   public void update();
}
```

Um observável fornecerá um método, através do qual os observadores podem registrar e anular o registro de seu interesse em atualizações. A Listagem 13.3 apresenta uma classe que implementa o padrão `Observer`.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

O modo de visualização

O modo de visualização é responsável por:

- Apresentar o modelo para o usuário
- Registrar no modelo notificação de mudança de estado
- Recuperar informações de estado do modelo

O modo de visualização é a camada da tríade MVC que exibe informações para o usuário. O modo de visualização obtém informações de exibição do modelo, usando a interface pública deste, e também se registrará no modelo para que ele possa ser informado da mudança de estado e se atualize de acordo.

Um único modelo pode ter muitos modos de visualização diferentes.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

O controlador

O controlador é responsável por:

- Interceptar os eventos do usuário do modo de visualização
- Interpretar o evento e chamar os métodos corretos do modelo ou modo de visualização
- Registrar-se no modelo para notificação de mudança de estado, se estiver interessado

O controlador atua como a cola entre o modo de visualização e o modelo. O controlador intercepta eventos do modo de visualização e depois os transforma em pedidos do modelo ou do modo de visualização.

Um modo de visualização tem apenas um controlador e um controlado tem apenas um modo de visualização. Alguns modos de visualização permitem que você configure seu controlador diretamente.

Cada modo de visualização tem um controlador e toda a interação com o usuário passa por esse controlador. Se o controlador for dependente das informações de estado, ele também será registrado no modelo para notificação de mudança de estado.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

Problemas com o MVC

Assim como acontece com qualquer projeto, o MVC tem suas deficiências e também seus pontos críticos. Os problemas incluem:

- Uma ênfase nos dados
- Um forte acoplamento entre o modo de visualização/controlador e o modelo
- Uma oportunidade de ineficiência

A gravidade dessas deficiências depende do problema que se está resolvendo e seus requisitos.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

Uma ênfase nos dados

Em uma escala OO de pureza, o padrão MVC não se classifica próximo ao topo, devido a sua ênfase nos dados. Em vez de pedir a um objeto para que faça algo com seus dados, o modo de visualização pede seus dados ao modelo e depois os exibe.

Você pode diminuir a gravidade desse problema, exibindo apenas os dados que retira do modelo. Não realize processamento adicional nos dados. Se você se achar realizando processamento extra nos dados, após recuperá-los ou antes de chamar um método no modelo, são boas as chances de que o modelo deve fazer esse trabalho para você. Existe uma linha tênue entre fazer muito e fazer o que é necessário nos dados. Com o passar do tempo, você aprenderá a diferenciar entre fazer muito com os dados e fazer apenas o que é necessário.

Se você verificar que repete o mesmo código em cada modo de visualização, considere a colocação dessa lógica no modelo.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

Acoplamento forte

O modo de visualização e o controlador são fortemente acoplados à interface pública do modelo. As alterações na interface do modelo exigirão alterações no modo de visualização e no controlador. Quando usa o padrão MVC, você supõe implicitamente que o modelo é estável, e é provável que o modo de visualização mude. Se esse não for o caso, você precisará escolher um projeto diferente ou estar preparado para fazer alterações no modo de visualização e no controlador.

O modo de visualização e o controlador também são intimamente relacionados entre si. Um controlador é quase sempre usado exclusivamente com um modo de visualização específico. Você pode tentar encontrar reutilização através de um projeto cuidadoso; mas mesmo que não encontre, o padrão MVC ainda fornece uma boa divisão de responsabilidades entre os objetos. A OO não é simplesmente um meio de reutilização.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

Ineficiência

Você deve tomar o cuidado de evitar ineficiências ao projetar e implementar uma UI baseada em MVC. As ineficiências podem aparecer no sistema em qualquer parte da tríade MVC.

O modelo deve evitar a propagação de notificações de mudança de estado desnecessárias para seus observadores. Um modelo pode enfileirar notificações de mudança relacionadas para que uma notificação possa significar muitas mudanças de estado. O modelo de evento AWT (Abstract Window Toolkit) da linguagem Java usa essa estratégia para redesenhar a tela. Em vez de redesenhar após cada evento, AWT enfileira os eventos e realiza uma única operação para redesenhar.

Ao projetar o controlador e o modo de visualização, talvez você queira considerar a colocação dos dados na memória cache, caso a recuperação de dados do modelo seja lenta. Após uma notificação de mudança de estado, recupere apenas o estado que mudou. Você pode aumentar o padrão do observador para que o modelo passe um identificador para o método `update()`. O modo de visualização pode usar esse identificador para decidir se precisa ou não se atualizar.

PROGRAMAÇÃO ORIENTADA A OBJETO

OO e programação de interface com o usuário

Resumo

A interface com o usuário é uma parte importante de qualquer sistema. Para alguns, ela pode ser a única parte do sistema com a qual eles interagem; para esses, a UI *é* o sistema. Você sempre deve encarar a análise, o projeto e a implementação da UI exatamente como encara qualquer outra parte do sistema. Uma UI nunca deve ser uma cogitação posterior ou algo colocado no sistema no último momento.

Embora existam muitas estratégias para o projeto da UI, o padrão MVC fornece um projeto que oferece flexibilidade, desacoplando a UI do sistema subjacente. Mas, assim como acontece com qualquer outra decisão de projeto, você ainda precisa ponderar os prós e contras do MVC, antes de decidir utilizá-lo. O MVC não o exime das realidades de seu sistema.

EXPLICADORES.NET

EXERCÍCIOS

PROGRAMAÇÃO ORIENTADA A OBJETOS

2016

83 – Assinale a alternativa que completa correta e respectivamente as lacunas da assertiva a seguir relacionada à programação orientada a objetos. O nível de acesso que você escolhe é muito importante para seu projeto. O acesso _____ garante o acesso a todos os objetos. Para garantir o acesso à instância, você precisa ter acesso _____ ou _____. Lembrando que o acesso _____ garante o acesso apenas para a instância, ou seja, para aquele objeto.

a) público – protegido – privado – privado
b) privado – privado – protegido – protegido
c) protegido – público – privado – privado
d) público – privado – protegido – protegido

PROGRAMAÇÃO ORIENTADA A OBJETOS

2016

83 – Assinale a alternativa que completa correta e respectivamente as lacunas da assertiva a seguir relacionada à programação orientada a objetos. O nível de acesso que você escolhe é muito importante para seu projeto. O acesso _____ garante o acesso a todos os objetos. Para garantir o acesso à instância, você precisa ter acesso _____ ou _____. Lembrando que o acesso _____ garante o acesso apenas para a instância, ou seja, para aquele objeto.

a) público – protegido – privado – privado
b) privado – privado – protegido – protegido
c) protegido – público – privado – privado
d) público – privado – protegido – protegido

2016

85 – Qual Padrão de Projeto apresenta uma solução para o problema de existir mais de uma instância de um objeto em determinado momento?

a) Singleton.
b) Abstract Factory.
c) Adapter.
d) Proxy.

2016

85 – Qual Padrão de Projeto apresenta uma solução para o problema de existir mais de uma instância de um objeto em determinado momento?

a) Singleton.
b) Abstract Factory.
c) Adapter.
d) Proxy.

2016

87 – Segundo Anthony Sintes, quais são as armadilhas que precisam ser evitadas no aprendizado, pela primeira vez, da Orientação a Objetos?

a) Pensar na POO simplesmente como uma linguagem; medo da reutilização; pensar na OO como uma solução para tudo; programação egoísta.

b) Pensar natural; pensar de forma confiante; pensar na reutilização; pensar na manutenção.

c) Pensar natural; pensar na POO simplesmente como uma linguagem; pensar na manutenção; programar de forma egoísta.

d) Pensar natural; medo da reutilização; pensar na OO como uma solução para tudo; programar de forma egoísta.

2016

87 – Segundo Anthony Sintes, quais são as armadilhas que precisam ser evitadas no aprendizado, pela primeira vez, da Orientação a Objetos?

a) Pensar na POO simplesmente como uma linguagem; medo da reutilização; pensar na OO como uma solução para tudo; programação egoísta.

b) Pensar natural; pensar de forma confiante; pensar na reutilização; pensar na manutenção.

c) Pensar natural; pensar na POO simplesmente como uma linguagem; pensar na manutenção; programar de forma egoísta.

d) Pensar natural; medo da reutilização; pensar na OO como uma solução para tudo; programar de forma egoísta.

2016

87 – Segundo Anthony Sintes, quais são as armadilhas que precisam ser evitadas no aprendizado, pela primeira vez, da Orientação a Objetos?

a) Pensar na POO simplesmente como uma linguagem; medo da reutilização; pensar na OO como uma solução para tudo; programação egoísta.

Armadilhas:

1- Nunca pense que ao usar uma linguagem OO, você está programando Orientado a Objeto:

A POO é muito mais do que simplesmente usar uma linguagem orientada a objetos ou conhecer certo conjunto de definições.

2 – Ter medo da reutilização:

Você deve aprender a reutilizar código. Aprender a reutilizar sem culpa frequentemente é uma das lições mais difíceis de aprender, quando você escolhe a POO pela primeira vez.

3 – Pensar na OO como uma solução para tudo:

Existem ocasiões em que você não deve usar orientação a objetos. Você ainda precisa usar bom senso na escolha da ferramenta correta para o trabalho a ser feito. Mais importante, a POO não garante o sucesso de seu projeto. Seu projeto não terá sucesso automaticamente, apenas porque você usa uma linguagem orientação a objetos. O sucesso aparece somente com planejamento, projeto e codificação cuidadosos.

4- Programação egoísta:

Assim como você deve aprender a reutilizar, também deve aprender a compartilhar o código que cria. Compartilhar significa que você encorajará outros desenvolvedores a usarem suas classes. Entretanto, compartilhar também significa que você tomará fácil para outros reutilizarem essas classes

PROGRAMAÇÃO ORIENTADA A OBJETOS

2016

88 – Segundo Anthony Sintes, quais são as três características do encapsulamento eficaz na Orientação a Objetos?

a) Abstração, ocultação da implementação e divisão de responsabilidade.

b) Implementação de classes com atributos públicos, herança simples e abstração.

c) Abstração, a não divisão de responsabilidade e a ocultação da implementação.

d) Abstração, ocultação da implementação dependendo da linguagem e a negação da divisão de responsabilidade.

PROGRAMAÇÃO ORIENTADA A OBJETOS

2016

88 – Segundo Anthony Sintes, quais são as três características do encapsulamento eficaz na Orientação a Objetos?

a) Abstração, ocultação da implementação e divisão de responsabilidade.

b) Implementação de classes com atributos públicos, herança simples e abstração.

c) Abstração, a não divisão de responsabilidade e a ocultação da implementação.

d) Abstração, ocultação da implementação dependendo da linguagem e a negação da divisão de responsabilidade.

PROGRAMAÇÃO ORIENTADA A OBJETOS

2016

89 – Marque (V) para verdadeiro ou (F) para falso. Em seguida, assinale a alternativa com a sequência correta.

( ) Objeto é um elemento da UML que permite a ampliação do vocabulário da própria linguagem UML.
( ) Na UML um relacionamento é uma conexão entre dois ou mais elementos da notação.
( ) A implementação define como algo é feito. Em termos de programação, implementação é o código.
( ) Estereótipo é uma construção de software que encapsula estado e comportamento.

a) F – V – V – V
b) V – V – F – V
c) V – F – V – F
d) F – V – V – F

PROGRAMAÇÃO ORIENTADA A OBJETOS

2016

89 – Marque (V) para verdadeiro ou (F) para falso. Em seguida, assinale a alternativa com a sequência correta.

( ) Objeto é um elemento da UML que permite a ampliação do vocabulário da própria linguagem UML.
( ) Na UML um relacionamento é uma conexão entre dois ou mais elementos da notação.
( ) A implementação define como algo é feito. Em termos de programação, implementação é o código.
( ) Estereótipo é uma construção de software que encapsula estado e comportamento.

a) F – V – V – V
b) V – V – F – V
c) V – F – V – F
d) F – V – V – F

2016

90 – Dentre as opções abaixo, qual representa uma das maneiras de se basear casos de teste?

a) Caixa preta.

b) Livre de erro.

c) Teste unitário reverso.

d) Teste de espera.

2016

90 – Dentre as opções abaixo, qual representa uma das maneiras de se basear casos de teste?

a) Caixa preta.

b) Livre de erro.

c) Teste unitário reverso.

d) Teste de espera.

2016

91 – Quais os três conceitos que são frequentemente referidos como os três pilares da POO?

a) Classe, objeto e herança.

b) Encapsulamento, herança e polimorfismo.

c) Classe, herança e polimorfismo.

d) Classe, encapsulamento e herança.

2016

91 – Quais os três conceitos que são frequentemente referidos como os três pilares da POO?

a) Classe, objeto e herança.

b) Encapsulamento, herança e polimorfismo.

c) Classe, herança e polimorfismo.

d) Classe, encapsulamento e herança.

PROGRAMAÇÃO ORIENTADA A OBJETOS

2017

54 – A Programação Orientada a Objetos (POO) define seis objetivos sobrepostos para desenvolvimento de software. Relacione a coluna da esquerda com a da direita, alinhando os objetivos com as características que explicam como a POO funciona para atender a cada um deles.

1 – Natural
2 – Confiável
3 – Reutilizável
4 – Manutenível
5 – Oportuno

( ) A natureza modular dos objetos permite fazer alterações em uma parte do programa, sem afetar outras partes. Os objetos isolam o conhecimento e a responsabilidade de onde pertencem.
( ) A programação orientada a objetos permite modelar um problema em um nível funcional e não em nível de implementação.
( ) O código orientado a objetos permite corrigir um erro em um lugar. Todos os outros objetos se beneficiarão automaticamente do aprimoramento.
( ) A programação orientada a objetos introduz a herança, para permitir que se estendam objetos existentes e o polimorfismo, para que se possa escrever código genérico.
( ) A divisão de um programa em vários objetos permite que o desenvolvimento de cada parte ocorra em paralelo. Vários desenvolvedores podem trabalhar nas classes independentemente.

a) 2 – 1 – 5 – 3 – 4
b) 2 – 5 – 4 – 3 – 1
c) 1 – 2 – 4 – 5 – 3
d) 2 – 1 – 4 – 3 – 5

PROGRAMAÇÃO ORIENTADA A OBJETOS

2017

54 – A Programação Orientada a Objetos (POO) define seis objetivos sobrepostos para desenvolvimento de software. Relacione a coluna da esquerda com a da direita, alinhando os objetivos com as características que explicam como a POO funciona para atender a cada um deles.

1 – Natural
2 – Confiável
3 – Reutilizável
4 – Manutenível
5 – Oportuno

( ) A natureza modular dos objetos permite fazer alterações em uma parte do programa, sem afetar outras partes. Os objetos isolam o conhecimento e a responsabilidade de onde pertencem.
( ) A programação orientada a objetos permite modelar um problema em um nível funcional e não em nível de implementação.
( ) O código orientado a objetos permite corrigir um erro em um lugar. Todos os outros objetos se beneficiarão automaticamente do aprimoramento.
( ) A programação orientada a objetos introduz a herança, para permitir que se estendam objetos existentes e o polimorfismo, para que se possa escrever código genérico.
( ) A divisão de um programa em vários objetos permite que o desenvolvimento de cada parte ocorra em paralelo. Vários desenvolvedores podem trabalhar nas classes independentemente.

a) 2 – 1 – 5 – 3 – 4
b) 2 – 5 – 4 – 3 – 1
c) 1 – 2 – 4 – 5 – 3
d) 2 – 1 – 4 – 3 – 5

PROGRAMAÇÃO ORIENTADA A OBJETOS

2017

86 – Correlacione os conceitos abaixo utilizados pela Programação Orientada a Objetos, relacionando a coluna da esquerda com a da direita; em seguida, assinale a alternativa que contém a sequência correta.

1 – Construtor
2 – Classe
3 – Comportamento
4 – Domínio
5 – Objeto

( ) É uma ação executada por um objeto quando passada uma mensagem ou em resposta a uma mudança de estado.
( ) Define os atributos e comportamentos comuns compartilhados por um tipo de objeto.
( ) É um método usado para inicializar objetos durante sua instanciação.
( ) É uma construção de software que encapsula estado e comportamento.
( ) Trata-se do espaço onde um problema reside

a) 3 – 2 – 1 – 4 – 5
b) 3 – 2 – 5 – 1 – 4
c) 2 – 3 – 4 – 1 – 5
d) 3 – 2 – 1 – 5 – 4

PROGRAMAÇÃO ORIENTADA A OBJETOS

2017

86 – Correlacione os conceitos abaixo utilizados pela Programação Orientada a Objetos, relacionando a coluna da esquerda com a da direita; em seguida, assinale a alternativa que contém a sequência correta.

1 – Construtor
2 – Classe
3 – Comportamento
4 – Domínio
5 – Objeto

( ) É uma ação executada por um objeto quando passada uma mensagem ou em resposta a uma mudança de estado.
( ) Define os atributos e comportamentos comuns compartilhados por um tipo de objeto.
( ) É um método usado para inicializar objetos durante sua instanciação.
( ) É uma construção de software que encapsula estado e comportamento.
( ) Trata-se do espaço onde um problema reside

a) 3 – 2 – 1 – 4 – 5
b) 3 – 2 – 5 – 1 – 4
c) 2 – 3 – 4 – 1 – 5
d) 3 – 2 – 1 – 5 – 4

PROGRAMAÇÃO ORIENTADA A OBJETOS

2017

87 – Um dos três pilares da Programação Orientada a Objetos (POO) é a Herança. A seguir, estão alguns termos relacionados com esse pilar da POO. Relacione a coluna da esquerda com a da direita e, em seguida, assinale a alternativa que contém a sequência correta.

1 – Herança
2 – “É um”
3 – Especialização
4 – Delegação
5 – Classe Folha

( ) Descreve o relacionamento em que uma classe é considerada do mesmo tipo de outra.
( ) É o processo de uma classe “filha” ser projetada em termos de como ela é diferente de sua progenitora.
( ) É um mecanismo que permite basear uma nova classe na definição de uma classe previamente existente.
( ) É o processo de um objeto passar uma mensagem para outro objeto, a fim de atender, a algum pedido.
( ) É uma classe sem filhas.

a) 2 – 3 – 1 – 5 – 4
b) 3 – 2 – 1 – 6 – 4

PROGRAMAÇÃO ORIENTADA A OBJETOS

2017

87 – Um dos três pilares da Programação Orientada a Objetos (POO) é a Herança. A seguir, estão alguns termos relacionados com esse pilar da POO. Relacione a coluna da esquerda com a da direita e, em seguida, assinale a alternativa que contém a sequência correta.

1 – Herança
2 – “É um”
3 – Especialização
4 – Delegação
5 – Classe Folha

( ) Descreve o relacionamento em que uma classe é considerada do mesmo tipo de outra.
( ) É o processo de uma classe “filha” ser projetada em termos de como ela é diferente de sua progenitora.
( ) É um mecanismo que permite basear uma nova classe na definição de uma classe previamente existente.
( ) É o processo de um objeto passar uma mensagem para outro objeto, a fim de atender, a algum pedido.
( ) É uma classe sem filhas.

a) 2 – 3 – 1 – 5 – 4
b) 3 – 2 – 1 – 6 – 4

PROGRAMAÇÃO ORIENTADA A OBJETOS

2017

88 – Selecione a alternativa que completa corretamente a afirmação:
_____ é a característica da programação orientada a objetos de ocultar partes independentes da implementação.

a) Interface
b) Abstração
c) Polimorfismo
d) Encapsulamento

PROGRAMAÇÃO ORIENTADA A OBJETOS

2017

88 – Selecione a alternativa que completa corretamente a afirmação:
_____ é a característica da programação orientada a objetos de ocultar partes independentes da implementação.

a) Interface
b) Abstração
c) Polimorfismo
d) Encapsulamento

PROGRAMAÇÃO ORIENTADA A OBJETOS

2018

61 – A maioria das linguagens orientadas a objetos (OO) suporta três níveis de acesso. Com base nesses três principais níveis, relacione-os com as suas características, sabendo que, das numerações abaixo, devem ser escolhidas 3 das 5 existentes.

1 – pacote
2 – público
3 – restrito
4 – protegido
5 – privado

( ) Garante acesso apenas para a instância, ou seja, aquele objeto.
( ) Garante acesso a todos os objetos.
( ) Garante acesso a instância e a todas as subclasses.

a) 1 – 2 – 3
b) 5 – 2 – 4
c) 3 – 1 – 2
d) 4 – 1 – 5

PROGRAMAÇÃO ORIENTADA A OBJETOS

2018

61 – A maioria das linguagens orientadas a objetos (OO) suporta três níveis de acesso. Com base nesses três principais níveis, relacione-os com as suas características, sabendo que, das numerações abaixo, devem ser escolhidas 3 das 5 existentes.

1 – pacote
2 – público
3 – restrito
4 – protegido
5 – privado

( ) Garante acesso apenas para a instância, ou seja, aquele objeto.
( ) Garante acesso a todos os objetos.
( ) Garante acesso a instância e a todas as subclasses.

a) 1 – 2 – 3
b) 5 – 2 – 4
c) 3 – 1 – 2
d) 4 – 1 – 5

PROGRAMAÇÃO ORIENTADA A OBJETOS

2018

67 – “Na década de 1990, o modelo baseado na orientação a objeto foi aplicado também aos bancos de dados, criando um novo modelo de programação conhecido como bancos de dados orientados a objeto. Os_____são valores definidos segundo_____, ou tipos de _____ complexos, com seus próprios operadores (métodos).”.

a) objetos – classes – dados
b) classes – dados – objetos
c) objetos – dados – classes
d) classes – objetos – dados

PROGRAMAÇÃO ORIENTADA A OBJETOS

2018

67 – “Na década de 1990, o modelo baseado na orientação a objeto foi aplicado também aos bancos de dados, criando um novo modelo de programação conhecido como bancos de dados orientados a objeto. Os_____são valores definidos segundo_____, ou tipos de _____ complexos, com seus próprios operadores (métodos).”.

a) objetos – classes – dados
b) classes – dados – objetos
c) objetos – dados – classes
d) classes – objetos – dados

PROGRAMAÇÃO ORIENTADA A OBJETOS

2018

76-Tendo por base a programação orientada a objetos (POO), analise as afirmativas abaixo e responda V para verdadeiro e F para falso.

( ) Interfaces são métodos usados para inicializar objetos durante sua instanciação. Inicializam um objeto durante sua criação.
( ) Uma interface lista os serviços fornecidos por um componente. É o contrato com o mundo exterior, que define exatamente o que uma entidade externa pode fazer com o objeto, informando com detalhes como o componente fará seu trabalho.
( ) O encapsulamento permite que se forneça um a implementação mais eficiente ou se corrijam erros, porém não permite que se atualize seu componente.
( ) A herança permite à classe que está herdando redefinir qualquer comportamento herdado de que não goste.

a) F – V – F – V
b) V – V – V – F
c) F – F – F – V
d) V – F – F – F

PROGRAMAÇÃO ORIENTADA A OBJETOS

2018

76-Tendo por base a programação orientada a objetos (POO), analise as afirmativas abaixo e responda V para verdadeiro e F para falso.

( ) Interfaces são métodos usados para inicializar objetos durante sua instanciação. Inicializam um objeto durante sua criação.
( ) Uma interface lista os serviços fornecidos por um componente. É o contrato com o mundo exterior, que define exatamente o que uma entidade externa pode fazer com o objeto, informando com detalhes como o componente fará seu trabalho.
( ) O encapsulamento permite que se forneça um a implementação mais eficiente ou se corrijam erros, porém não permite que se atualize seu componente.
( ) A herança permite à classe que está herdando redefinir qualquer comportamento herdado de que não goste.

a) F – V – F – V
b) V – V – V – F
c) F – F – F – V
d) V – F – F – F

A primeira assertiva é falsa, pois está fazendo a definição de construtores e não de interfaces.
Os construtores são métodos usados para inicializar objetos durante sua instanciação. Inicializam um objeto durante sua criação.

A segunda assertiva é falsa, pois uma interface **não** informa: como o componente fará seu trabalho. Em vez disso, a interface oculta a implementação do mundo exterior. Isso libera o componente para alterações na sua implementação a qualquer momento. As mudanças na implementação não mudam o código que usa a classe, desde que a interface permaneça inalterada. As alterações na interface necessitarão de mudanças no código que exerce essa interface.

A terceira assertiva é falsa, pois o encapsulamento permite que você atualize seu componente, forneça uma implementação mais eficiente ou corrija erros - tudo isso sem ter de tocar nos outros objetos de seu programa. Os usuários de seu objeto se beneficiarão automaticamente de todas as alterações que você fizer.

PROGRAMAÇÃO ORIENTADA A OBJETOS

2018

78 – Sobre a programação orientada a objetos, informe se a assertiva é falsa (F) ou verdadeira (V) e assinale a alternativa com a sequência correta.

( ) As três características de encapsulamento eficaz são: abstração, ocultação da implementação e divisão de responsabilidade.

( ) Abstração é o processo mais complexo de um problema.

( ) O nível de acesso Público garante o acesso apenas para a instância.

( ) O encapsulamento permite que você divida o programa em várias partes menores e independentes.

a) F – V – F – V
b) F – V – V – F
c) V – F – F – F
d) V – F – F – V

PROGRAMAÇÃO ORIENTADA A OBJETOS

2018

78 – Sobre a programação orientada a objetos, informe se a assertiva é falsa (F) ou verdadeira (V) e assinale a alternativa com a sequência correta.

( ) As três características de encapsulamento eficaz são: abstração, ocultação da implementação e divisão de responsabilidade.

( ) Abstração é o processo mais complexo de um problema.

( ) O nível de acesso Público garante o acesso apenas para a instância.

( ) O encapsulamento permite que você divida o programa em várias partes menores e independentes.

a) F – V – F – V
b) F – V – V – F
c) V – F – F – F
d) V – F – F – V

PROGRAMAÇÃO ORIENTADA A OBJETOS

2020

50 – Considerando os conceitos de introdução a UML, assinale a alternativa cuja assertiva é verdadeira.

a) Um estereótipo descreve um relacionamento ‘é um’ entre duas classes.

b) Uma linguagem de modelagem é uma notação hexadecimal para descrever projetos de software.

c) Uma agregação é um tipo especial de associação que modela o reclacionamento ‘tem um’ entre pares.

d) Uma composição é menos rigorosa do que a agregação, pois é um relacionamento entre pares completamente independentes

PROGRAMAÇÃO ORIENTADA A OBJETOS

2020

50 – Considerando os conceitos de introdução a UML, assinale a alternativa cuja assertiva é verdadeira.

a) Um estereótipo descreve um relacionamento ‘é um’ entre duas classes.

b) Uma linguagem de modelagem é uma notação hexadecimal para descrever projetos de software.

c) Uma agregação é um tipo especial de associação que modela o relacionamento ‘tem um’ entre pares.

d) Uma composição é menos rigorosa do que a agregação, pois é um relacionamento entre pares completamente independentes

PROGRAMAÇÃO ORIENTADA A OBJETOS

2020

62 – Considere as representações UML das classes abaixo e assinale a alternativa que possui dois atributos privados.

a)

MilitarAeronautica
idade : int + posto : String # especialidade : String

b)

MilitarAeronautica
- idade : int - posto : String + especialidade : String

c)

MilitarAeronautica
+ idade : int - posto : String # especialidade : String

d)

MilitarAeronautica
+ idade : int + posto : String # especialidade : String

PROGRAMAÇÃO ORIENTADA A OBJETOS

2020

62 – Considere as representações UML das classes abaixo e assinale a alternativa que possui dois atributos privados.

a)

MilitarAeronautica
idade : int + posto : String # especialidade : String

b)

MilitarAeronautica
- idade : int - posto : String + especialidade : String

c)

MilitarAeronautica
+ idade : int - posto : String # especialidade : String

d)

MilitarAeronautica
+ idade : int + posto : String # especialidade : String

B

PROGRAMAÇÃO ORIENTADA A OBJETOS

2020

64 – Tratando-se dos tipos de teste de software, assinale a alternativa que fala sobre testes de integração.

a) Examinam todo o sistema.

b) Verificam apenas um recurso por vez.

c) Verificam se dois ou mais objetos funcionam em conjunto corretamente.

d) Examinam as alterações nas partes do sistema que foram validadas anteriormente.

PROGRAMAÇÃO ORIENTADA A OBJETOS

2020

64 – Tratando-se dos tipos de teste de software, assinale a alternativa que fala sobre testes de integração.

a) Examinam todo o sistema.

b) Verificam apenas um recurso por vez.

c) Verificam se dois ou mais objetos funcionam em conjunto corretamente.

d) Examinam as alterações nas partes do sistema que foram validadas anteriormente.

PROGRAMAÇÃO ORIENTADA A OBJETOS

2020

82 – Considere as representações UML das classes abaixo e assinale a alternativa que apresenta uma sobrecarga de métodos.

a)

ComparadorNumerico
+ max(x : int, y : int) : int + max(x : long, y : long) : long

b)

ComparadorNumerico
+ max(x : int, y : int) : int + min(x : int, y : int) : int

c)

ComparadorNumerico
+ max(x : int, y : int) : int + min(x : int) : void

d)

ComparadorNumerico
+ max(x : int, y : int) : int + maior(x : int, y : int) : int

PROGRAMAÇÃO ORIENTADA A OBJETOS

2020

82 – Considere as representações UML das classes abaixo e assinale a alternativa que apresenta uma sobrecarga de métodos.

a)

ComparadorNumerico
+ max(x : int, y : int) : int + max(x : long, y : long) : long

b)

ComparadorNumerico
+ max(x : int, y : int) : int + min(x : int, y : int) : int

c)

ComparadorNumerico
+ max(x : int, y : int) : int + min(x : int) : void

d)

ComparadorNumerico
+ max(x : int, y : int) : int + maior(x : int, y : int) : int

A

PROGRAMAÇÃO ORIENTADA A OBJETOS

2020

86 – Sobre o padrão de projeto de software MVC (Model View Controller), relacione a coluna da esquerda com a da direita e assinale a alternativa correta.

1 – Modelo
2 – Controlador
3 – Padrão Observer
4 – Modo de visualização

( ) Exibe informações para o usuário.
( ) Gerencia o comportamento básico e o estado do sistema.
( ) Atua como a cola entre o modo de visualização e o modelo.
( ) Fornece ao projeto um mecanismo de publicação/assinatura entre objetos.

a) 4 – 1 – 2 – 3
b) 1 – 3 – 2 – 4
c) 4 – 1 – 3 – 2
d) 1 – 4 – 2 – 3

PROGRAMAÇÃO ORIENTADA A OBJETOS

2020

86 – Sobre o padrão de projeto de software MVC (Model View Controller), relacione a coluna da esquerda com a da direita e assinale a alternativa correta.

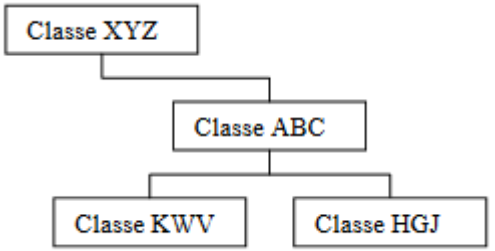
1 – Modelo
2 – Controlador
3 – Padrão Observer
4 – Modo de visualização

( ) Exibe informações para o usuário.
( ) Gerencia o comportamento básico e o estado do sistema.
( ) Atua como a cola entre o modo de visualização e o modelo.
( ) Fornece ao projeto um mecanismo de publicação/assinatura entre objetos.

a) 4 – 1 – 2 – 3
b) 1 – 3 – 2 – 4
c) 4 – 1 – 3 – 2
d) 1 – 4 – 2 – 3

PROGRAMAÇÃO ORIENTADA A OBJETOS

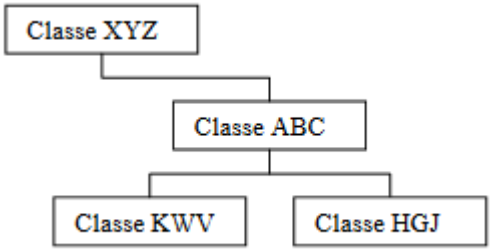
10- Considerando os conceitos relacionados à Programação Orientada a Objetos, observe a figura abaixo e assinale a afirmativa correta a respeito das classes representadas.



a) A classe KWV não é uma classe folha.
b) A classe HGJ é uma classe descendente da classe XYZ.
c) Considerando a hierarquia inteira representada pela figura, a classe ABC é uma classe raiz.
d) Considerando a hierarquia inteira representada pela figura, a classe XYZ é uma classe ancestral da classe ABC.

PROGRAMAÇÃO ORIENTADA A OBJETOS

10- Considerando os conceitos relacionados à Programação Orientada a Objetos, observe a figura abaixo e assinale a afirmativa correta a respeito das classes representadas.



a) A classe KWV não é uma classe folha.
b) A classe HGJ é uma classe descendente da classe XYZ.
c) Considerando a hierarquia inteira representada pela figura, a classe ABC é uma classe raiz.
d) Considerando a hierarquia inteira representada pela figura, a classe XYZ é uma classe ancestral da classe ABC.