CES-28 Prova 3 - 2017

Sem consulta - individual - com computador - 3h

Ítalo Tabatinga Braga – COMP19

Obs.:

1. Qualquer dúvida de codificação Java só pode ser sanada com textos/sites oficiais da Oracle ou JUnit.
    a. Exceção são idiomas (ou 'macacos') da linguagem como sintaxe do método .equals(), ou sintaxe de set para percorrer collections, não relacionados ao exercício sendo resolvido. Nesse caso, podem procurar exemplos da sintaxe na web.
2. Sobre o uso do mockito, podem usar sites de ajuda online para procurar exemplos da sintaxe para os testes, e o próprio material da aula com pdfs, exemplos de codigo e labs, inclusive o seu código, mas sem usar código de outros alunos.
3. Questões com itens diversos, favor identificar claramente pela letra que representa o item, para que eu saiba precisamente a que item corresponde a resposta dada!
4. Só precisa implementar usando o Eclipse ou outro ambiente Java as questões ou itens indicados com o rótulo **[IMPLEMENTAÇÃO]**! Para as outras questões, você pode usar o Eclipse caso se sinta mais confortavel digitando os exemplos, mas não precisa de um código completo, executando. Basta incluir trechos de código no texto da resposta.
5. Submeter: a) Código completo e funcional da questão **[IMPLEMENTAÇÃO]; b)** arquivo PDF com respostas, código incluso no texto para as outras questões. Use os números das questões para identificá-las.
6. No caso de diagramas, vale usar qualquer editor de diagrama, e vale também desenhar no papel, tirar foto, e **incluir a foto no pdf dentro da resposta, não como anexo separado**. Atenção: use linhas grossas, garanta que a foto é legível!!!!

Joãozinho programa Interpolação **[IMPLEMENTAÇÃO]**

O *package* InterpV0 inclui uma aplicação de interpolação numérica. Há duas classes que implementam métodos de interpolação (não precisa lembrar os detalhes de CCI22, basta lembrar o conceito de interpolação). E há outra classe MyInterpolationApp que realiza todo o trabalho. A proposta principal desta questão é transformar o package de Joãozinho em 3 *packages* Model, View e Presenter que implementam o padrão arquitetural MVP.

Deve incluir uma view funcional, mas que imprime no console, e com métodos que simulam entrada do usuário humano. Por exemplo, se o usuário humano deveria digitar um inteiro, basta haver um método set(int value). Quando a main() chamar este método, simulamos entrada de usuário.

Deve garantir que:

1. **[2 pt] O conceito de camadas seja seguido estritamente, e cada camada esteja em um package separado**.
   RESP: No código.
2. **[2 pt]** Que seja possível adicionar outras implementações da camada View, com as mesmas responsabilidades, e usar várias instâncias de Views diferentes ao mesmo tempo com a mesma instância de Presenter e Model, **sem necessitar mudar o código de Presenter ou Model**.
   RESP: Presenter é um Observable e as View são Observers, basta associar cada view ao Presenter no construtor que o sistema funciona. Caso a responsabilidade da view seja satisfeita, pode-se mudar o código pois sua função é passar os dados pro Presenter.

3. **[2 pt] SUBQUESTÃO [IMPLEMENTAÇÃO]:** (esta parte envolve um padrão de projeto além do MVP). Seja possível implementar e escolher outros algoritmos de interpolação, **sem precisar mudar nada no código além de uma chamada de método para registrar o novo algoritmo.** *As camadas superiores apenas precisam escolher uma String correspondendo ao nome do método de interpolação desejado.*
   RESP: Atualmente existe um switch-case no Presenter para selecionar qual o algoritmo para ser usado. A solução para isso é usar um HashMap<String, InterpolationMethod> em Presenter que armazena os códigos dos algoritmos(value) associados a uma String (key). No método bind() chamado no construtor são designados os algoritmos padrões e caso queria registrar um novo basta criar uma nova classe com o método calculateResult(double t, Vector<Double> xx, Vector<Double> yy).

[1 pt] Para cada uma das responsabilidades de MyInterpolationApp, indicadas com comentários no código e listadas abaixo, indique marcando uma colunas entre M, V ou P neste documento em qual camada deve ser incluída CADA responsabilidade. **DEVE CORRESPONDER AO SEU CÓDIGO**:

	M	V	P
1. RESPONSABILITY: DEFINIR PONTO DE INTERPOLACAO (LEITURA ENTRADA DE USUARIO HUMANO)		X	
2. RESPONSABILITY: DEFINIR QUAL EH O ARQUIVO COM DADOS DE PONTOS DA FUNCAO (LEITURA ENTRADA DE USUARIO HUMANO)		X	
3. RESPONSABILITY: ABRIR E LER ARQUIVO DE DADOS			X
4. RESPONSABILITY: IMPRIMIR RESULTADOS		X	
5. RESPONSABILITY: DADO O VALOR DE X, EFETIVAMENTE LER O ARQUIVO			X
6. RESPONSABILITY: DADO O VALOR DE X, EFETIVAMENTE CHAMAR O CALCULO			X

7. RESPONSABILITY: CRIAR O OBJETO CORRESPONDENTE AO METODO DE INTERPOLACAO DESEJADO			X
8. RESPONSABILIDADE: EFETIVAMENTE IMPLEMENTAR UM METODO DE INTERPOLACAO	X		

GRASP x SOLID

[1pt : 0.5 por princípio] Para a solução do exercício da interpolação, explique como a solução final promove 2 princípios GRASP ou SOLID (não vale os princípios que apenas definem menor acoplamento e separação de responsabilidades, High Coesion, Low Coupling, Single Responsability).
RESP: 1) Open/Closed Principle pois o código é acessível apenas pelos métodos public que não permitem a modificação do mesmo mas possui possibilidade de expansão de algoritmos por dois meios: registrando o algoritmo feito pelo programador-usuário e registrando pelo método criado na questão passada(o que foi pedido) ou alterando o código para ter mais algoritmos padrões criando classes que implementam a interface que já existe (não foi pedido).

2) Controller pois no código original o MyInterpolationApp tratava todos os eventos, desde de instanciar classes de Algoritmos até leitura de Dados, depois da solução a classe Presenter possui a maioria das responsabilidades, já que está é que faz o tratamento dos eventos de todas as classes View, funcionando como o sistema, recebendo e enviando dados para View e requisitando resultados de Model.

DPs são tijolos para construir Frameworks

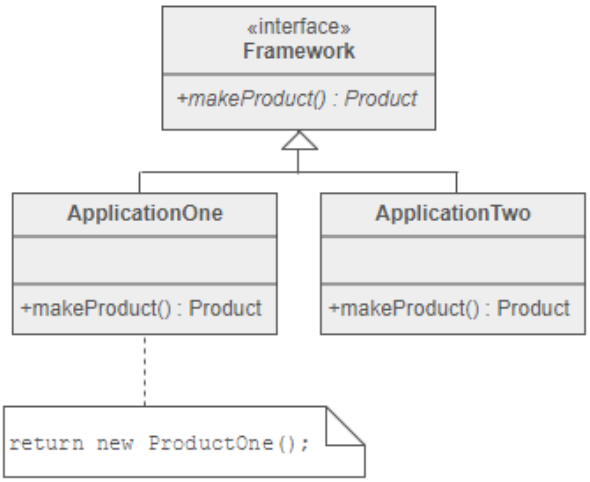
[2 pt: 2 * { a) [0.5] b [0.5] }]
Escolha **2 (dois)** DPs que ao serem aplicados como parte do código de um Framework, promovam:

a) o **reuso de código**
b) a **separação de interesses** (separation of concerns), entre o código do framework e o código do programador-usuário do framework.

Explique conceitualmente como cada um 2 DPs promove os 2 conceitos a) e b). Vale usar diagramas UML na explicação, mas *deixe claro o que deve ser implementado pelo framework e o que deve ser implementado pelo programador-usuário do framework*.
RESP:

Factory Method:

a) Esse DP promove a reutilização do código do Framework pois esse possui uma estrutura definida do padrão de criação de objetos com alguns resultados padrões.

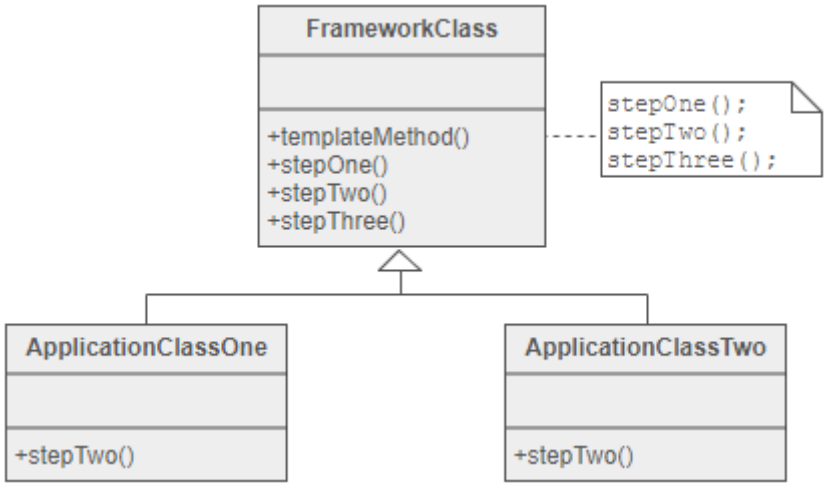


Pode-se ver no UML acima que o Framework originalmente possui alguns padrões já organizados por aplicação de makeProduct(), caso o programador-usuário desejasse utilizar o framework para outra aplicação bastaria criar o método makeProduct() de uma ApplicationThree, por exemplo.

b) Como mostrado pelo diagrama anterior, o código do framework seria para definir a estrutura de criação do objeto(Classe Framework) e, usualmente, alguns casos mais gerais dessa criação(Classes ApplicationOne e ApplicationTwo), já o programador usuário definiria uma classe (Classe ApplicationThree) que vai reescrever os métodos definidos na estrutura de criação do framework.

Template Method:

a) Semelhante ao Factory Method, porém o Template Method tem aplicação na estrutura de um algoritmo, não na criação de uma instância. Esse DP define a estrutura do algoritmo de um método específico. Essa estrutura é formada por Hook Methods.



No UML,acima, o templateMethod() chama os métodos stepOne(), stepTwo() e stepThree(). O método stepTwo() é chamado de HookMethod pois este depende da Classe que chamou(ApplicationClassOne ou ApplicationClassTwo). Promove

o reuso pois caso o programador-usuário desejasse ele poderia definir sua Classe ApplicationClassThree que reescreveria os Hook Methods.

b) Como mostrado pelo diagrama, o código do Framework será para definir a estrutura do algoritmo de um método(FrameworkClass), onde essa estrutura é compostos por alguns HookMethods, e, possivelmente, algumas implementações gerais dos HookMethods(ApplicationClassOne e ApplicationClassTwo). O programador-usuário definiria uma nova Classe (ApplicationClassThree) com suas implementações dos Hook Methods.

Abusus non tollit Usum

Conceito	Consequência do Abuso do conceito Marque o número apropriado conforme lista abaixo
Singleton DP	1 2 3
Dependency Injection	1 2 3
Getters and Setters	1 2 3

1. Excessiva quantidade de código e classes auxiliares para inicializar objetos
2. Acoplamento excessivo e código difícil de entender devido à proliferação de Dependências e conflitos de nomes.
3. Confusão semântica dependendo da ordem de chamada de métodos, resultando em objetos com estado inválido.

a) **[0.5]** Associe cada conceito à consequência do seu abuso, marcando os números apropriados na a tabela acima, conforme a lista acima.

RESP: Está marcado na tabela.

b) **[1]** Escolha Singleton ou Dependency Injection e explique a causa da consequência, explicando o contexto do abuso do conceito.

RESP: Singleton. Normalmente tem uma situação onde é necessária que apenas uma instância de uma Classe exista, para chamar métodos. Sem usar o DP, teremos que passar como parâmetro a instância que criamos inicialmente para todas as chamadas desses métodos, isso gera as consequências citadas.

c) **[0.5]** Para o mesmo conceito escolhido em b), explique um contexto de uso apropriado, em que há razões claras para se utilizar o conceito sem incorrer nas consequências negativas.

RESP: Quando é necessária que exista apenas uma instância de uma Classe na aplicaão deve-se tratar essa Classe como um Singleton. Pode-se citar como um exemplo real um

Banco de Dados, pois para uma aplicação costuma-se ter somente um Banco de Dados para o mesmo conjunto de funções, não é vantajoso criar várias instâncias do mesmo banco de dados já que eles devem acessar o mesmo.