

CES-28 Prova 3 - 2017

Sem consulta - individual - com computador - 3h

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 código 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 confortável 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.
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**.
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.

[1 pt] Para cada uma das responsabilidades de MyInterpolationApp, indicadas com comentários no código e listadas abaixo, indique marcando uma coluna 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 Responsibility).

Resposta:

- 1) Expert, pois as classes Lagrange e CubicSpline, por exemplo, são especialistas no seus modelos de cálculo.
- 2) Creator, pois para atender o requisito 2, foi preciso utilizar o modelo de Singleton para a instanciação do Presenter, logo, apenas essa classe é responsável por criar instâncias dela mesma.

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*.

Resposta:

- 1) Facade: através de uma fachada, posso acessar diferentes códigos a partir de uma única classe. Além disso, evito o acesso direto do usuário a um certo conjunto de classes.
- 2) Observer: através da divisão entre observer e subject posso utilizar uma informação em diferentes classes ao mesmo tempo, reutilizando o código dessa forma. Também separo os interesses, pois os observers são incumbidos apenas de atualizar seus dados a medida que o subject os informa de alguma mudança no seu estado.

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.

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

Resposta:

Singleton é utilizado para evitar a instanciação de mais de um objeto de uma certa classe. Entretanto, caso o programador tenha consciência da necessidade de apenas um objeto, ele pode facilmente criar apenas um de forma muito menor (em relação a tamanho de código). Por exemplo, se precisamos criar apenas um objeto da classe house, ao invés de aplicar Singleton, posso apenas criar um objeto com nome ONLY_house, dessa forma serei constantemente lembrado que não é correto fazer outra instanciação dessa classe.

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.

Resposta:

Em um banco, por exemplo, algumas taxas de juro só podem ser atualizadas em uma hora específica do dia. Dessa forma, se tivermos uma classe TaxaX, que contém informações sobre a taxa e seu tipo de cálculo, seria muito importante instanciar apenas um objeto para garantir que todos os clientes estão pagando a mesma taxa. Além disso, o objeto precisa ser tipo "final" para ser imutável naquele período.