CES-28 Exame - 2017

Ítalo Tabatinga Braga – COMP19

Com consulta - individual - com computador - 3h

Obs.:

1. Para a resolução da prova, podem ser usados 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.
2. Questões com itens diversos, favor identificar claramente (inserir comentário) pela letra que representa o item, para que se saiba precisamente a que item corresponde a resposta dada!
3. Só precisa implementar deve ser usado o Eclipse, onde essas questões ou itens serão 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.
4. **Submissão -> VIA TIDIA e GITHUB**
    a. Código completo e funcional da questão **[IMPLEMENTAÇÃO] -** deve ser gerado um arquivo zip ou rar contendo o projeto eclipse, assim como todas as bibliotecas**;**
    b. Arquivo PDF com respostas, código incluso no texto para as outras questões. Use os números das questões para identificá-las.
    c. Arquivo texto contendo nome completo e repositório GITHUB.
5. 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!!!!

UNMANNED AIRCRAFT SYSTEM (UAS) TRAFFIC MANAGEMENT (UTM)

A inserção de aeronaves remotamente pilotadas (ARP), ou como elas são mais conhecidas - drones, em espaços aéreos urbanos é uma realidade que estará presente em um futuro muito próximo. Várias empresas ao redor do mundo vêm buscando forçar os órgãos reguladores, responsáveis pelo gerenciamento e controle do espaço aéreo, a se posicionarem e apresentarem quais os rumos que a normatização irá tomar, visando dessa forma, garantir o entendimento de quais são as tecnologias candidatas, bem quais são os critérios de operação e padrões de segurança a serem adotados.

Uma vez que operações de pequenos drones já são bastante comuns de serem vistos nas cidades, principalmente em missões de transmissão de imagens reais de grandes acontecimentos, se faz necessário caracterizar essa porção do espaço aéreo, definindo, por exemplo, quais os critérios de acesso a esse espaço, buscando evitar que colisões entre aeronaves, danos à pessoas e patrimônio, bem como garantir isonomia entre empresas que pretendem desenvolver novos planos de negócio baseado no uso dessa tecnologia. Para isso, os países têm trabalhado na construção de um novo espaço aéreo e será utilizado apenas por aeronaves não pilotadas (ou remotamente pilotadas) – o Unmanned Traffic Management (UTM). O UTM tem por objetivo identificar serviços, funções, responsabilidades, arquitetura de informações, protocolos de troca de dados, funções de software, infraestrutura e requisitos de desempenho para permitir o gerenciamento de operações de aeronaves remotamente pilotadas não controladas em espaços aéreos de baixa altitude (< 500 Ft). Ou seja, o UTM é um ecossistema de "gerenciamento de tráfego" para operações não controladas que é independente do sistema convencional de controle de tráfego aéreo (ATM), mas que é complementar ao mesmo.

Um dos problemas relacionados a sistemas UTM é como prover consciência situacional a todos atores em tempo real, uma vez que muitas vezes eventos diversos e dinâmicos precisam ser compartilhados, por terem grande impacto no desenvolvimento da operação.

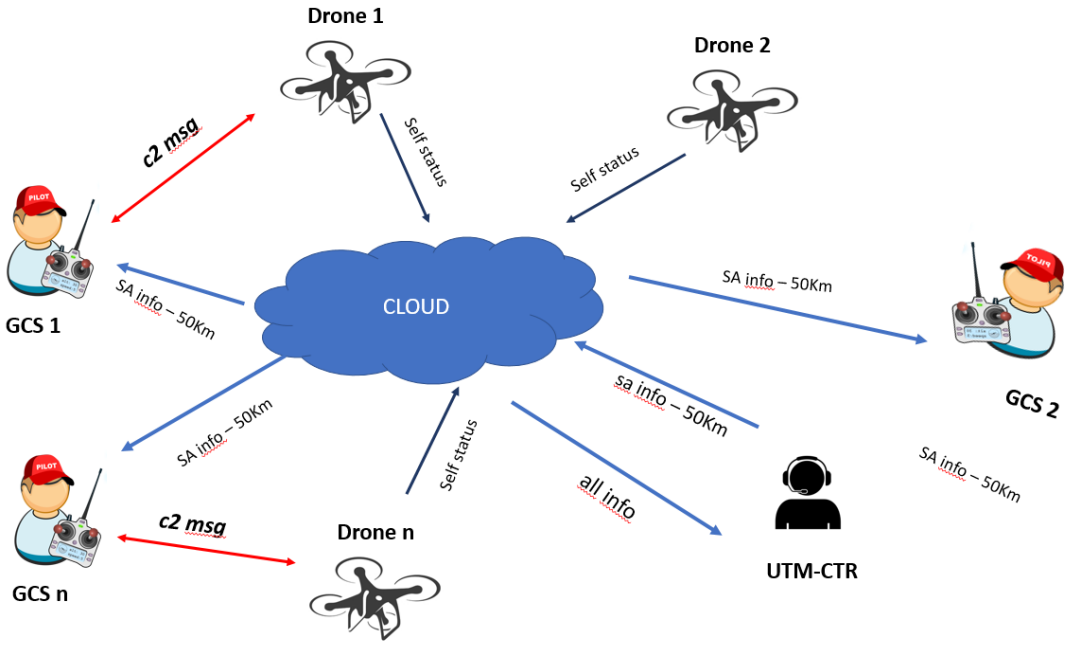
Basicamente, essa operação possui os seguintes atores, conforme apresentado na figura abaixo:

a) Drone (ARP): plataforma aérea que possui uma inteligência embarcada que é capaz de detectar obstáculos e outras aeronaves, assim como evitar que a aeronave venha a colidir com os mesmos (função *avoid-colision*). Ela envia de 10 em 10 seg, via CLOUD, suas informações de posicionamento (***self status***) para a UTM-CTR. É importante que cada drone possui uma GCS associada e este possui um link de comunicação ponto-a-ponto (***c2 msg***) – **NÃO CONSIDERAR ESSE CANAL PARA EFEITO DESSA PROVA**.

b) GCS: estação de controle de um ARP específico, conforme legislação pertinente, que pode interferir através de um piloto habilitado no que o drone está a realizar (mudar altura, prumo, etc.), através do link de comando e controle que ele tem com o drone (***c2 msg***) – **NÃO CONSIDERAR ESSE CANAL PARA EFEITO DESSA PROVA**. Essa estação recebe informações diretamente do drone que ela controla, assim como envia informações para o mesmo (***c2 msg***) – **NÃO CONSIDERAR ESSE CANAL PARA EFEITO DESSA PROVA**. É importante mencionar que a GCS possui informações do seu ambiente (consciência situacional), ou seja, ela possui um sistema de mapas, que permite com que ela realize sua navegação, que contém informações de posicionamento de outras aeronaves (raio de 50 km), bem como informações meteorológicas, entre outras, que são enviadas de 20 em 20 seg da UTM-CTR (via CLOUD) (***sa info***).

c) UTM-CTR: sistema de controle e vigilância de tudo que ocorre em um espaço UTM. É ela que recebe via CLOUD todas as informações oriundas dos ARPS (***all infos***), bem como informações meteorológicos, trata essas informações (compila) e gera um mapa síntese de informações (mapa de consciência situacional), que apresenta as

informações necessárias para o funcionamento das GCSs em sua área de controle. Para isso, ela envia de 20 em 20seg esse mapa síntese (***sa info***) para as GCS. Uma informação importante, é que esse serviço deve ser único, ou seja, não pode ter mais de um rodando por vez.



PARTE II – QUESTÕES

1) **[1.5 PTOS]** De posse do problema apresentado na Parte I, mapeie 03 requisitos funcionais, os escrevendo por via de estórias de usuários (*user stories*), atentando para que os mesmos atentem para possuir as características do mnemônico INVEST (Independente, Negociável, Valiosa, Estimável, Pequena (Small) e Testável.
    a. Observação: Caso falte alguma das informações requeridas para a construção de estórias de usuários, use sua imaginação para complementar a mesma.
    b. As estórias dos usuários devem ser descritas no Word (Libre Office), não devendo ser usado nenhum sistema como Taiga, etc.

    RESP:

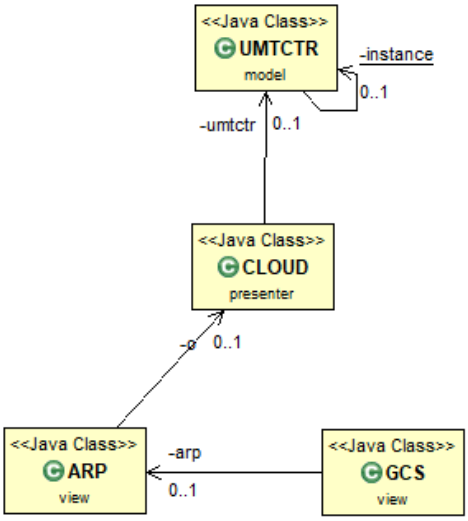
    1. Como um ARP, quero ser capaz detectar obstáculos, para assim evitar a colisão com os mesmos.
    2. Como GCS, quero receber informação da UMT-CTR via CLOUD, para assim navegar sem correr riscos.
    3. Como UTM-CTR, quero receber todas as informações dos ARPs pela CLOUD, para assim poder gerenciar e enviar apenas o necessário para as GCSs.
2) **[0.5 PTOS]** Apresente (explique), como suas palavras, em cada uma das estórias de usuários os atributos contidos em INVEST são obtidos.

RESP: Irei evidenciar as características de INVEST mais marcantes por estória : pode-se perceber a característica **Independente** na estória de usuário 1, pois esta não possui maiores dependências com outras estórias e pode ser feita isolada. A característica **Negociável** pode ser percebida ao notar a prioridade das estórias, a estória 3 depende de que informações os ARPs devem fornecer, portanto pode ser feita ao decorrer do projeto; já a estória 1 possui menor correlação com outras e pode ser feita no começo do projeto. Estórias 1 e 2 mostram característica **Valiosa** para o consumidor-usuário, dado que garantem uma certa segurança em relação ao produto (ARP). As características **Estimável** e **Pequena** estão presentes nas 3 estórias, dado que estas demonstram pequenas funcionalidades do todo. A característica **Testável** pode ser notada na estória 1, dado que o caso sucesso é não haver colisões.

3) **[1.0 PTOS]** Perceba que na descrição do UTM-CTR informa que ele deve ser implementado de forma única, ou seja, não pode existir mais de uma instância dele rodando por vez. Qual o padrão de projeto que resolve essa questão? Explique.

RESP: O padrão de projeto Singleton. Esse padrão garante que essa classe só terá uma instância e provê acesso global a esta pois o seu construtor está encapsulado, dessa forma só é possível receber uma instância da classe através de um método static. Esse método checa se já existe uma instância da classe, se não existir (caso da primeira requisição) ele cria a instância e armazena que esta já foi criada, e depois retorna a instância armazenada.

4) **[2.0 PTOS]** O problema apresentado na Parte I, claramente nos remete a uma série de padrões aprendidos no ano de 2017 em CES 28. Entretanto, de forma bem explícita podemos ver o padrão arquitetural MVP (*Model View Presenter*) e o Observer. Usando o problema, construa um diagrama de classe que apresente uma solução para o problema.
public
    a. Obs1: Lembre-se de que uma das vantagens do MVP é seu baixo acoplamento, bem como uma separação clara de interesses. Dessa forma, atente para definir que um conceito é uma classe (concreta / abstrata) ou uma interface.
    b. Obs2: Identifique de forma clara, como os padrões se apresentam em seu diagrama de classe.



Pode-se perceber dois padrões evidentes: **Singleton** na classe UMCTR e **MVP** no conjunto. UMTCTR funciona como Model analisando a informação por completo que CLOUD. CLOUD é o Presenter, faz a interação entre as duas partes. Já a view pode-se dizer composta por ARP e GCS, pois na descrição do problema ficou claro que ARP enviava sua posição para CLOUD porém GCS que recebia as informações condensadas de UMTCTR, através de CLOUD, dessa forma ambos fazem parte da View. Além disso, o padrão **Observer** foi implementado para que mais de 1 GCS possa ter conhecimento do estado de CLOUD, dessa forma GCS é o Observer e CLOUD o Observable.

5) **[IMPLEMENTAÇÃO]** De posse do diagrama do exercício 4, usando linguagem Java implemente o mesmo. Construa uma classe principal (Main), que demonstre o correto funcionamento dos padrões usados em sua solução. Por exemplo, no caso do MVP deve ser apresentadas que as responsabilidades entre as entidades da arquitetura (model, view, controller) funcionam adequadamente.
   a. Obs1: Atente que não é necessário implementar nenhuma interface gráfica, onde na demonstração, deve-se imprimir na console, as mensagens de forma a entender o funcionamento do código gerado.
   b. Obs2: Para efeito do exemplo devem ser instanciados ao menos 3 instâncias de GCS, consequentemente 03 instâncias de Drones.
   c. Obs3: A solução deve tratar o problema apresentado na questão 3 (instância única do UTM-CTR).
   d. Salve todo o código gerado num pacote utm_v0.

CORREÇÃO:

i. **[2.0 PTOS] – IMPLEMENTAR CORRETAMENTE A ARQUITETURA MVP**
ii. **[1.0 PTOS] – IMPLEMENTAR UMA CLASSE MAIN QUE DEMONSTRE O MVP**
iii. **[1.0 PTOS] – TRATAR O PROBLEMA DA QUESTÃO 5.C**

RESP: Na solução temos a Main que demonstra as funcionalidades do MVP implementado', todo o processo é implementado no Console simulando a interface gráfica com que View interage. São instaciadas 3 GCSs, os ARPs são instanciados

no construtor de GCS, assumindo que um ARPs cadastrado sempre está associado a um GCS, ou é feita a associação de um ARP a um GCS. A Classe UMTCTR é implementada com o padrão de projeto Singleton para garantir sua unicidade.

6) **[IMPLEMENTAÇÃO]** Transforme seu código, de forma a substituir sua classe principal (Main) por testes projetados em JUNIT e MOCKITO.
    a. Obs1: Lembre-se um teste, uma funcionalidade.
    b. Obs2: Os nomes dos testes, devem lembrar o que está sendo testado.
    c. Obs3: Em cada teste, deve-se ter um comentário (JAVADOC) do que está se fazendo.
    d. Salve todo o código gerado num pacote utm_v1.

CORREÇÃO:

i. **[1.0 PTOS] – USO CORRETO DO MOCKITO (NÃO USAR CLASSES FUNCIONAIS, QUANDO SE PODE MOCAR)**
ii. **[2.0 PTOS] – ESTRATÉGIA CORRETA DE TESTES -> DEMONSTRAR EM TESTES SEPARADOS O MODEL, O VIEW, O CONTROLLER**
iii. **[1.0 PTOS] – DEMONSTRAR EM TESTES A SOLUÇÃO DO PROBLEMA DA QUESTÃO 5.C**

RESP: Foram elaborados 3 arquivos de teste, uma para cada Classe do modelo MVP. Os testes foram feitos baseados nas funções principais desses arquivos para o sistema funcionar. Vale ressaltar que o quesito iii está no arquivo de teste de Model.