université de BORDEAUX	**ANNEE UNIVERSITAIRE 2018 / 2019** **SESSION D'AUTOMNE DECEMBRE 2018** **MENTION** : MASTER INFORMATIQUE **PARCOURS / ETAPE : Code UE : 4TIN706U** **Epreuve : Approche objet, Examen** **Date :** 13/12/2018 **Heure : 9 h** **Durée :** 3h Documents : autorisés Epreuve de M : Blanc Xavier	**Collège Sciences et technologies** **Masters**

Questions de cours (4 point) :

Q1 Le DDD est-il applicable pour Java uniquement ou est-il possible de l'appliquer à d'autres langages orientés objet ? Justifiez votre réponse.

Les patterns stratégiques du DDD s'appliquent à tout projet informatique. Les patterns tactiques du DDD s'appliquent aux langages OO car ils portent essentiellement sur les concepts d'encapsulation et de couplage.

Q2 Un Value Object peut-il changer dans le temps ? Donnez un exemple.

Non un Value Object ne peut pas changer dans le temps. Le point GPS est l'exemple classique du VO.

Q3 Une commande (dans le CQRS) exprime-t-elle une intention d'une modification des objets métier ou un résultat d'une modification ? Donnez un exemple (sans code).

Une commande exprime une intention de modification d'objet métier, par exemple, l'ajout d'un élément dans un panier électronique.

Q4 Faite un schéma avec les quatre couches de l'architecture octogonale avec des flèches précisant les dépendances entre les couches.

Les quatre couches sont : domain, application, interface et infra. Il faut montrer que application dépend de domain, interface dépend de applicaton et domain. Enfin, infra dépend de domain, application et interface. On parle ici de dépendance de classe (dépendance de code source).

Questions conception objet (12 points) :

Une jeune équipe de développeurs Java est en train de développer une application pour gérer des championnats. Un championnat a les caractéristiques suivantes :

- Un championnat (**Championship**) a un nom et une capacité max de joueurs
- Un championnat est ouvert ou fermé aux inscriptions. Quand il est ouvert, on peut ajouter un nouveau joueur (dans la limite de la capacité max).

- Un championnat fermé peut démarrer. Il n'est alors plus possible de le rouvrir et de changer les inscriptions des joueurs.
- Un joueur (**Player**) a un pseudo
- L'application dispose de l'ensemble de tous les joueurs (**PlayerRepository**)
- Chaque joueur inscrit dans un championnat qui a démarré doit rencontrer tous les autres joueurs dans des matchs.
- Un match (**Match**) oppose deux joueurs d'un championnat. Un match est en cours (même s'il n'a pas encore démarré) ou terminé. Quand il est en cours, il est possible de modifier le nombre de points de chaque joueur. Quand il est terminé, on ne peut plus modifier les points des joueurs, par contre, on peut savoir qui a gagné.
- Lorsque tous les matchs d'un championnat sont joués, le championnat est terminé.
- Lorsqu'un championnat est terminé, on peut obtenir le classement d'un championnat en considérant qu'un match gagné compte 3 points et qu'un match nul compte 1 point.

Q1 Précisez les concepts DDD à mettre en œuvre (Value Object, Entity, Aggregate, Repository) pour **Player** et **PlayerRepository**. Codez **Player** et **PlayerRepository**. Vous préciserez les packages. Enfin pour **PlayerRepository** vous proposerez la classe d'implémentation qui stocke les joueurs en mémoire.

Player est un VO.

PlayerRepository est un Repository.

Ces deux classes vont dans le package domaine.

Enfin, il faut une infrastructure réalisant PlayerRepository.

Voir https://github.com/xblanc33/championship

Q2 Précisez le concept DDD à mettre en œuvre (Value Object, Entity, Aggregate, Repository) pour **Match**. Codez la classe correspondante en ne donnant que ses propriétés et les signatures de ses méthodes. Ne codez pas le corps des méthodes.

Match est un Entity.

Il faut proposer des méthodes pour changer le score, obtenir le gagnant et surtout gérer le cycle de vie (démarrer un match, etc.).

Voir https://github.com/xblanc33/championship

Q3 Précisez le concept DDD à mettre en œuvre (Value Object, Entity, Aggregate, Repository) pour **Championship**. Codez la classe correspondante en ne donnant que ses propriétés et les signatures de ses méthodes. Ne codez pas le corps des méthodes.

Championship est un aggregate car il gère la vie des matchs qu'il contient. Il n'est pas possible de démarrer un match si un de ses joueurs et déjà impliqué dans un autre match qui a démarré (et n'est pas terminé).

Voir https://github.com/xblanc33/championship

Q4 Codez la méthode permettant de calculer le classement d'un championnat terminé.

Il faut mettre cette méthode dans la classe Championship. Il faut retourner un ordre !!!

Voir https://github.com/xblanc33/championship

Q5 On souhaite mettre en place une architecture CQRS pour mettre à jours les matchs d'un championnat. En outre, on veut disposer des commandes suivantes : *UpdateMatchScore* et *CloseMatch* permettant respectivement de mettre à jours le score d'un match et de clore un match. Codez toutes les classes nécessaires à la mise en place de cette architecture.

Voir https://github.com/xblanc33/championship

Test (4 points) :

On considère que la classe **Group** contient un ensemble de joueurs.

La fonction « Set<Group> generateSubGroup(Group group, int size) » permet de générer des sous-groupes de Joueurs à partir groupe de départ. Le paramètre « size » précise la taille maximale des sous-groupes.

Par exemple, si le groupe de départ est : group1 = [« Alice », « Bob », « Brian »] alors generateSubGroup(group1, 2) peut retourner [[« Alice »], [« Bob », « Brian »]] ou même [[« Alice », « Bob »], [« Brian »]], voir même [[« Alice »], [« Bob »], [« Brian »]], etc.

Proposez le code permettant de tester cette fonction.

Le test devra en particulier vérifier :

- que tous les joueurs du groupe de départ sont dans exactement un des sous-groupes,
- qu'il n'existe pas de joueur dans un sous-groupe qui n'appartient pas au groupe de départ,
- qu'il n'existe pas de sous-groupe vide,
- qu'il n'existe pas de sous-groupe dont la taille dépasse la taille maximale.

Vous préciserez la façon de lancer vos tests, et donc quels sont les résultats attendus !