M1 : Ingénierie du Logiciel - UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Examen 2eme Session

7 Juin 2017 (2 heures avec documents : tous SAUF ANNALES CORRIGEES).

Barème <u>indicatif</u> sur 21,5 points (donne le poids relatif des questions) (max 20/20).

Questions de cours [4 Pts]

Répondez de façon précise et concise aux questions.

Question Cours (QC):

QC1) Expliquez la fonction et le rôle d'un serveur d'intégration continue.

QC2) Pourquoi préconise-t-on de n'utiliser que des types simples (Bool, Int, String) dans les signatures d'opérations d'interface ?

QC3) Expliquez la notion de raffinement qui existe entre les diagrammes de séquence d'analyse et les diagrammes de séquence inter-composants réalisés à l'étape conception architecturale dans l'approche en V de l'UE.

QC4) Quel est le rôle d'un composant bouchon dans un test d'intégration ? Est-on toujours obligé d'en définir ?

2. Problème: Analyse StoneHearth [Barème sur 8 Pts]

Votre compagnie a décidé d'investir sur le segment des jeux pour téléphone mobiles "free to play/pay to win", et veut se lancer en force avec un nouveau jeu de cartes ultra-original et inédit. L'équipe chargée du design graphique promet d'innover avec des dragons, des robots, des lasers, des mages... De quoi plaire à tous les publics.

Le Jeu "StoneHearth" est un jeu à deux joueurs, où les joueurs s'affrontent en jouant chacun leur tour des cartes. La particularité du jeu par rapport aux jeux de cartes classiques (tarot, belote...) est que les cartes sont toutes différentes et dotées d'effets particuliers. Chaque carte a un nom et une description qui explique informellement ses effets.

Chaque joueur a donc une collection de cartes, à partir de laquelle il devra composer son "deck", c'est-à-dire choisir les 30 cartes (sans doublon) qu'il utilisera pour affronter son adversaire. Il dispose d'une interface lui permettant de mémoriser 3 decks qu'il peut utiliser en partie, mais il peut débloquer d'autres emplacements de decks pour 2eu ou 3$ chacun. Les cartes sont rangées en quatre catégories de "rareté" croissante : "basique", "commune" puis "rare" puis "légendaire". Les cartes les plus puissantes sont aussi les plus rares.

Le modèle commercial est que les cartes ne peuvent pas être échangées entre les joueurs. Pour ajouter des cartes à sa collection, le joueur doit ouvrir des "packs", contenant 5 cartes aléatoires dont au moins une "rare". Les nouveaux joueurs ont droit automatiquement à toutes les cartes "basique" ce qui leur permet de composer un deck, plus 5 packs gratuits pour se lancer.

Les packs peuvent être achetés dans le jeu pour 1,39eu ou $1.99. Les joueurs assidus se voient également attribuer un pack toutes les 10 parties jouées, dans la limite de deux packs par 24h.

Comme les cartes issues des packs sont aléatoires, un joueur ayant des cartes en double peut détruire une carte ce qui lui donne des "joyaux". Les joyaux sont une monnaie dans le jeu, qui ne sert qu'à acheter des cartes, mais au choix de l'utilisateur cette fois.

La seule façon d'en obtenir est de détruire des cartes de sa collection (typiquement les cartes obtenues en double, mais pas forcément). Détruire une carte commune rapporte 2 joyaux, une carte rare 5 joyaux, une carte légendaire rapporte 20 joyaux.

Les cartes "basique" ne peuvent pas être détruites ni obtenues dans les packs. Pour acheter une carte commune il faut 20 joyaux, une carte rare 50 joyaux, une carte légendaire coûte 200 joyaux.

Les joueurs doivent créer un compte pour se connecter au jeu et accéder à leur collection de cartes et leurs decks mémorisés. Ils doivent y renseigner leurs coordonnées bancaires pour pouvoir acheter des cartes, mais ce n'est pas obligatoire de le faire immédiatement. Ils peuvent accéder aux options du compte par la suite, et les saisir quand ils le souhaitent.

Les joueurs connectés peuvent alors directement décider de jouer contre un joueur de même rang. Ils choisissent un deck dans la liste des decks qu'ils ont mémorisé, puis le jeu leur trouve un adversaire de rang similaire (écart <= 3 rangs).

Le rang des nouveaux joueurs est initialement 0. Chaque partie gagnée incrémente le rang (sauf s'il est 100), chaque partie perdue le décrémente de 1 (sauf s'il est à 0).

Les joueurs jouent alors chacun leur tour des cartes jusqu'à ce que l'issue soit décidée par la victoire d'un des joueurs; il n'y a pas de match nul possible.

Dans cet énoncé on ne détaillera pas les règles du jeu lui-même, ni le déroulement de la partie.

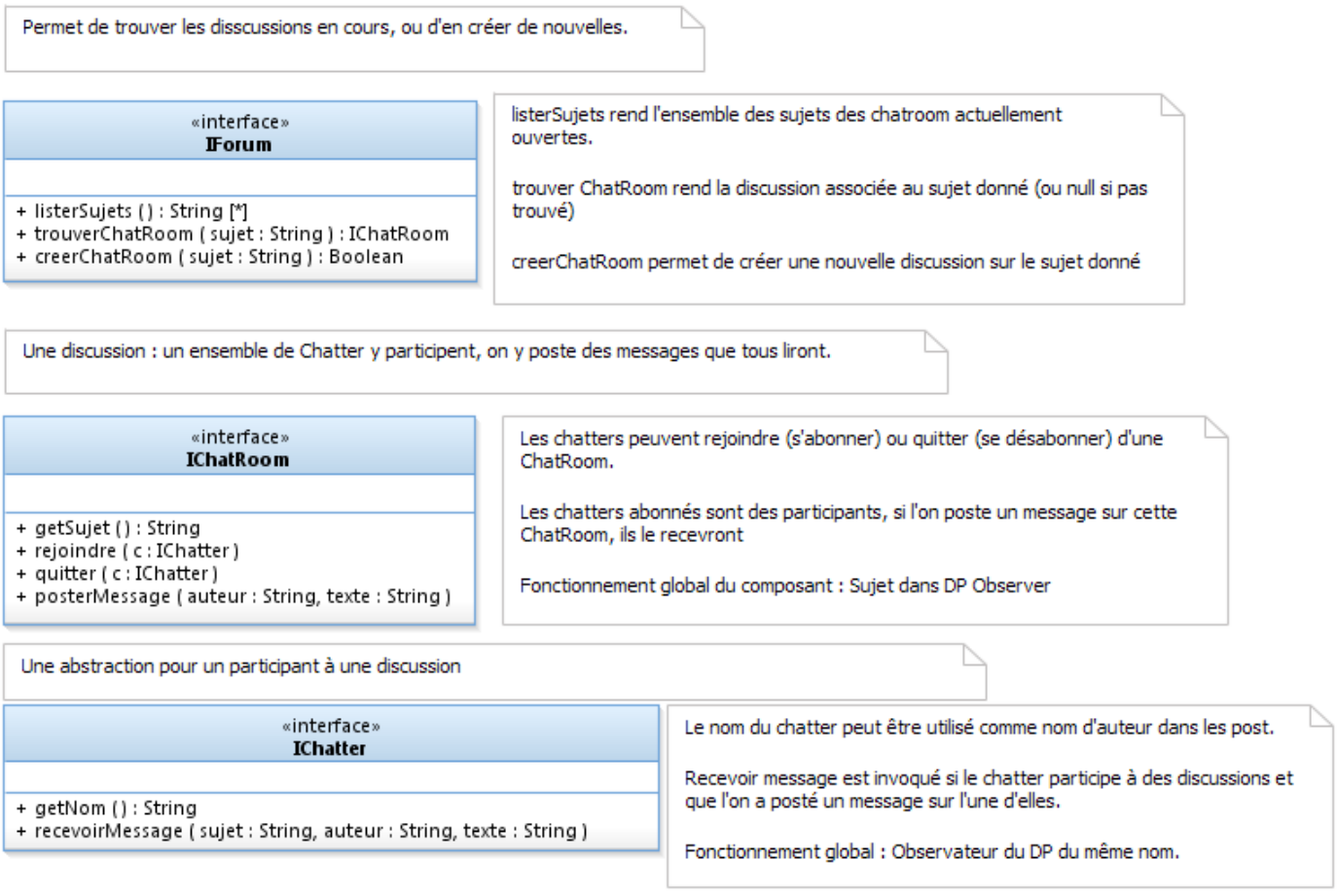
Question 2.1 : (3 pts) Réalisez le diagramme de cas d'utilisation de la phase d'analyse. Vous justifierez tous vos choix, par un texte ou des annotations sur le diagramme.

Question 2.2 : (3 points) Réalisez le diagramme de classes métier de la phase d'analyse. Vous justifierez tous vos choix, par un texte ou des annotations sur le diagramme. On ne représentera pas la classe représentant le « Système », introduite dans l'approche en V du module.

Question 2.3 : (2 pts) Ecrivez un test de validation couvrant l'achat d'une carte avec des joyaux.

3. Problème: Conception ChatRoom [Barème sur 8,5 Pts]

On considère un système de Chat ou discussion instantanée en ligne, on donne les interfaces :



Un forum est un point central de connection qui permet de savoir quelles sont les discussions en cours.

IChatRoom fonctionne sur le modèle du DP Observer : chaque IChatRoom est associée à un ensemble de participants, c'est à dire des IChatter ayant *rejoint* la IChatRoom. Un IChatter participant à une IChatRoom va recevoir des notifications à chaque fois que l'on y poste un message.

Donc posterMessage("toto","Hello") sur une IChatRoom nommée "UML" doit provoquer l'invocation sur tous les participants de recevoirMessage("UML","toto","Hello").

Question 3.1 : (3 pts)

On considère trois composant : CSimpleChatter, CSimpleChatRoom, CSimpleForum qui réalisent chacun de façon ***minimale*** mais fonctionnellement correcte une des interfaces introduites (par exemple *recevoirMessage* peut se contenter d'écrire sur stdout).

a) (1,5) Représentez sur un diagramme de composant ces trois composants (interfaces requises et offertes).
b) (1,5) Proposez à l'aide d'un diagramme de classes une conception détaillée possible du composant CSimpleForum.

Question 3.2 : (1 pts)

Le composant CClientChat est une application complète qui tourne sur les machines des chatters. Il permet de se connecter à un IForum, d'y trouver ou d'y créer une IChatRoom, de la rejoindre, puis de poster et de recevoir des messages.

Sur un diagramme de composants représentez ce composant CClientChat.

Question 3.3 : (3,5 pts)

On considère la situation suivante : un forum qui contient deux chatroom de sujets respectifs « UML » et « Java ». « Bob » et « Alice » sont deux utilisateurs utilisant un client de chat. Bob a rejoint à la fois « UML » et « Java », mais Alice n'a rejoint que « UML ».

a) (1,5 pts) Modélisez cette situation sur un diagramme de structure interne.
b) (2 pts) Dans un diagramme de séquence de niveau intégration (une ligne de vie par composant) mettant en jeu les mêmes instances de composant, modélisez le scenario :
Alice crée et rejoint une chatroom "UML". Bob cherche et trouve UML puis rejoint la discussion. Alice envoie un message "bonjour" dans la chatroom « UML » (qui doit donc être reçu par Bob et Alice).

Question 3.4 : (2 pts)

Dans l'esprit du DP Decorator (cf Annexe), on souhaite permettre plus de flexibilité au niveau de la définition des IChatRoom, c'est-à-dire pouvoir doter a posteriori (après son instanciation) une IChatRoom de propriétés supplémentaires.

On considère un composant CChatRoomFiltre, qui empêche de poster des messages avec des gros mots sur un IChatRoom donné (poster ne fait rien si on détecte des gros mots dans le texte du message).

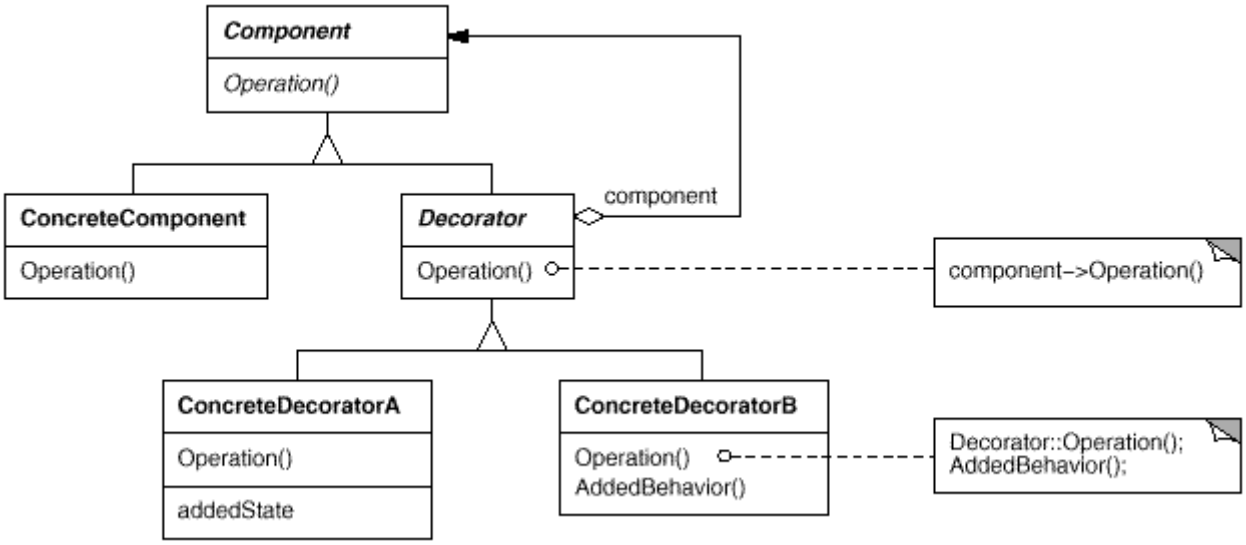
On considère également un composant CChatRoomPrivée, qu'on initialise avec une liste de noms d'utilisateurs, et qui seront les seuls à pouvoir rejoindre la ChatRoom concernée.

a) (1 pt) Sur un diagramme de composants, modélisez ces deux nouveaux composants.
b) (1 pt) Représentez sur un diagramme de structure interne une instanciation d'une ChatRoom protégée par un filtre sur les utilisateurs **et** sur les gros mots.

Annexe : DP Decorator (extrait du GOF)

Intention : Attacher dynamiquement de nouvelles responsabilités aux objets. Les décorateurs offrent une alternative flexible à l'héritage pour étendre des fonctionnalités.

Structure :



Participants :

- Component : une interface qu'implantent les objets auxquels il faut attacher de nouvelles responsabilités
- ConcreteComponent : un objet (simple) auquel on veut attacher des responsabilités
- Decorator (abstrait): détient une référence à un Component, et implante cette interface par délégation sur cette instance. Permet de factoriser le code des décorateurs concrets.
- DecorateurConcret : redéfinit les opérations souhaitées, pour ajouter du comportement à l'objet décoré.