EXAMEN - Design Patterns
RATTRAPAGE

ING-2, Génie Informatique, Hiver 2017
Sébastien Rufiange, Julien Mercadal

Durée de l'examen : 2h
Type d'examen : Papier
Conditions d'examen :

- Accès à une feuille A4 recto-verso manuscrite seulement.
- Les machines sont interdites.
- Les tentatives de communication sous toute forme sont interdites durant l'examen.
- Posez des hypothèses en cas de doute.

Mise en situation
Vous travaillez comme stagiaire chez Nintendo France et votre contrat prendra fin sous peu. Votre maître de stage croit en vos capacités et vous permet donc de passer un examen d'entrée en vue d'une embauche. Au cours de ce test, vous devez démontrer votre aptitude à réaliser de bons designs. Si votre candidature est retenue, vous pourrez participer au développement d'un jeu de rôle, prévu pour la plateforme Android en 2018. Les diagrammes doivent respecter la notation UML et les extraits de code doivent être écrits en Java. Vous devez fournir des réponses complètes et cohérentes (*i.e.*, le diagramme UML doit correspondre au code Java).

Question 1 - Personnage - 4 points
Le joueur débute l'aventure en créant son personnage, composé d'un nom et d'une classe.
Chaque personnage possède aussi des points d'attaque, des points de défense, une rapidité (vitesse d'attaque), des points de vie, des points de magie. À la création du personnage, celui-ci possède des caractéristiques de base : une attaque de 100, une défense de 100, une vitesse de 100, 1000 points de vie et 200 points de magie.
Pour le moment, il y a seulement trois classes (guerrier, ninja, mage) mais cela pourrait changer dans le futur. À tout moment dans le jeu, le choix de classe du personnage va altérer ses caractéristiques de base à l'exécution et de la manière décrite ci-après.
Guerrier : + 100 Attaque, + 50 défense, - 50 rapidité, + 50 vie, - 200 magie.
Ninja : + 50 Attaque, + 50 défense, + 50 rapidité, - 50 vie, - 50 magie.
Mage : - 50 Attaque, - 50 Défense, - 50 rapidité, - 50 vie, + 200 magie.
Proposez donc un diagramme de classes complet, cohérent et adapté à cette situation

(a) En utilisant de l'héritage. **[2 points]**
(b) En utilisant de la composition. **[2 points]**

Question 2 - Équipement - 4 points
Des équipements peuvent donner lieu à divers bonus ou encore à des pénalités.
Un personnage peut porter plusieurs équipements en même temps. Dans ce cas, les effets peuvent aussi se combiner de manière flexible et on ne sait pas encore quelles combinaisons seront permises dans le jeu.

(a) Illustrez votre design en créant un objet Personnage, puis en lui affectant ces trois équipements avec du code Java **[1 point]** :
(1) une Amulette de Rapidité augmentant la rapidité de 10% de la valeur de base,
(2) une Armure Ancestrale augmentant la défense de 20 points et la magie de 10 points,
(3) un Bâton Magique augmentant la magie de 25% de la valeur totale (prenant en compte les 10 points précédemment ajoutés par l'Armure Ancestrale) et réduisant le nombre de points de vie maximal de 10%.
Affichez les caractéristiques du personnage avant et après le port des équipements.

(b) Tracez un schéma UML complet et clair de votre design. **[2 points]**

(c) Indiquez les noms des design patterns utilisés le cas échéant et justifiez clairement de leur pertinence. **[1 point]**

Question 3 - Inventaire - 4 points
Les objets qui tombent font partie d'un inventaire, associé au personnage.
Cet inventaire comprend des objets, mais aussi des ensembles d'objets, peuvent eux-mêmes contenir d'autres ensembles d'objets. Par exemple, un coffre peut contenir plusieurs pièces d'or, ou encore un sac d'objets. Un objet a aussi un poids. Dans le design du jeu, vous devez opter pour une solution bien adaptée à ce contexte.

Dans la situation présentée ici, vous devez créer un coffre, contenant 100 x 1 pièce d'or ainsi qu'un sac, contenant lui-même un casque et une amulette de longévité.

(a) Quel design pattern vous semble le plus adapté à cette situation ? Justifiez votre réponse. **[1 point]**

(b) Proposez un schéma UML complet et cohérent. **[2 points]**

(c) Fournissez du code Java pour illustrer la situation et afficher le poids total des objets mentionnés. **[1 point]**

Question 4 - Unique - 5 points

Suite aux combats entre le personnage et des ennemis, de l'équipement peut tomber. Par exemple, des armes, des élixirs, des parchemins, etc. Certains de ces équipements sont uniques. S'il s'agit d'un équipement unique, il porte un nom qui doit être unique et il ne doit apparaître qu'une seule fois dans le jeu.

Au cours du jeu, il faut créer ces divers objets. Proposez un design pour gérer la construction de ces objets en prenant en compte la création des objets uniques. Pour cette question, vous devez en outre concevoir un logiciel pour la création d'épées uniques. Votre classe Jeu doit créer trois épées uniques nommées *Dard*, *Excalibur* ainsi que *Durandal*. Il ne doit pas être possible de créer deux fois la même épée (c'est-à-dire ayant le même nom). Vous devez proposer une solution la plus générique possible.

(a) Quel design pattern vous semble le plus approprié ? **[1 point]**
(b) Illustrez votre design par un schéma UML. **[2 points]**
(c) Fournissez le code Java de l'implémentation du pattern. Créez les objets mentionnés dans la question puis ajoutez-les à l'inventaire du personnage *Léo*. **[2 points]**

Question 5 - Patterns - 3 points

Vous devez associer les bons design patterns à chacune des situations ci-dessous. Une bonne réponse donne 0.5 point. Une mauvaise réponse ou une absence de réponse enlève 0.5 point. Vous pourrez obtenir un maximum de 3 points pour cette question et un minimum de zéro.

(a) Gérer un historique d'opérations.
(b) Avoir une seule instance d'objet.
(c) Faire varier des implémentations d'algorithmes.
(d) Accéder à un objet distant de manière transparente.
(e) Rendre un objet compatible avec un autre qu'on ne peut modifier.
(f) Simplifier l'utilisation de plusieurs classes.
(g) Être renseigné lors de changements d'états d'objets.
(h) Modéliser une hiérarchie d'objets.

FIN DE L'EXAMEN