Conception Logicielle - TD 3

‡ ‡ ‡

Jeu des personnages – V2.0 : Patrons de conception

Notre jeu de tests bien en place, nous pouvons maintenant, de manière plus sereine, faire évoluer notre cahier des charge du jeu des personnages.

I - Les patterns de comportement

Intéressons nous tout d'abord au concept de *troupe*.

Exercice I.1

Troupes. . . En marche !

À l'aide du pattern observateur, nous souhaitons ici modéliser le concept de troupe : plus précisément nous souhaitons regrouper l'ensemble des personnages et éléments pouvant se déplacer (i.e. l'ouvrier et le guerrier qui marche, le dragon qui vole et la catapulte qui roule si elle est reliée à un guerrier).

1. Modifier votre diagramme de classes UML pour faire apparaître ce nouveau concept.
   On réfléchira également :
   - ↬ à la possibilité de rendre une troupe déplaçable
   - ↬ et donc à la possibilité de manipuler une troupe au sein d'une autre troupe. . .
2. Implémenter ces évolutions dans votre projet java
3. Tester.

Exercice I.2

Évolution. . . des personnages et de leurs comportements

Nous souhaitons maintenant apporter quelques modifications concernant les comportements de déplacement et d'attaque de nos personnages.

Précisément, les personnages et objets pourront évoluer au cours du jeu de manière à ce que, au départ :

- ↬ un ouvrier
  - ⇝ se déplace toujours en marchant
  - ⇝ n'attaque pas et n'attaquera jamais
- ↬ un guerrier
  - ⇝ se déplace en courant ou en marchant
  - ⇝ attaque avec un couteau
- ↬ un petit nouveau, le mage
  - ⇝ se déplace en marchant
  - ⇝ n'attaque pas tant qu'il ne connait pas de sort
- ↬ un dragon
  - ⇝ se déplace toujours en volant (donc pas de modification)
  - ⇝ attaque en crachant du feu

et, **au cours du jeu :**

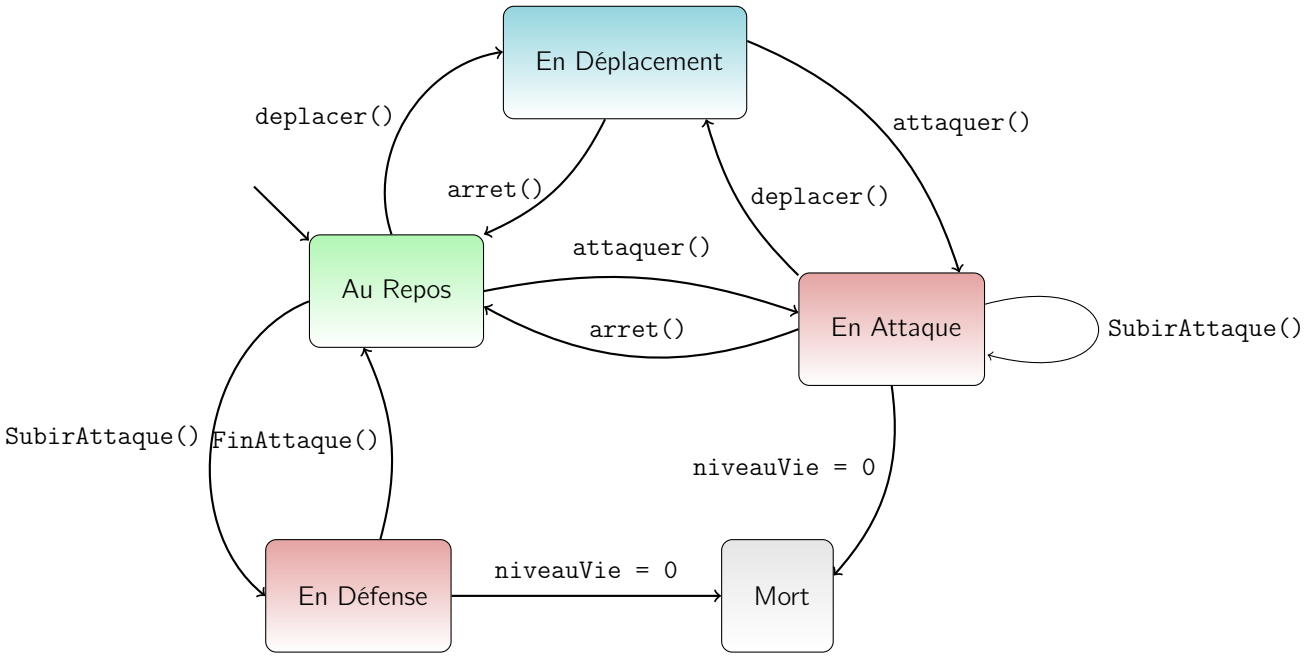
- ↬ un ouvrier :
  - ⇝ pourra évoluer pour se déplacer en courant
- ↬ un guerrier :
  - ⇝ pourra évoluer pour se déplacer en courant
  - ⇝ pourra évoluer pour attaquer avec une hache
- ↬ un mage :
  - ⇝ pourra évoluer pour se déplacer en flottant
  - ⇝ pourra évoluer pour attaquer en lançant un sort

1. Modifier votre diagramme de classe UML afin de prendre en compte cette évolution. Vous mettrez en œuvre le pattern strategy pour la modélisation des nouvelles fonctions de déplacement et d'attaque des personnages.
2. Implémenter, dans votre projet Java, les modifications apportées précédemment dans la conception UML.

Exercice I.3

Gestion des états

Nos personnages étant maintenant pourvus de comportement d'attaque et de déplacement, il nous faut gérer les états de ces derniers.
L'automate définissant les différents états pris par un personnage et les transitions associées est donné ci-après.



1. En utilisant notamment le pattern état, modifier votre diagramme de classe UML afin de prendre en compte la gestion des états des personnages.
2. Implémenter votre nouveau diagramme de classes. On réfléchira à l'implémentation à prévoir lorsqu'une transition n'est pas tirable depuis un état donné.

II - Les patterns de structure

Exercice II.1

L'équipement du guerrier ...

Afin de remplir la lourde tâche qui leur incombe, les guerriers peuvent, au cours du jeu, gagner ou acheter des options. On décide ici d'implémenter les options suivantes :

- une armure permettant de diminuer par 2 les dommages des coups reçus par le personnage
- un casque permettant de diminuer par 2 les dommages des coups reçus par le personnage
- une amulette doublant la puissance d'attaque
- un pouvoir de double attaque donnant la possibilité d'attaquer avec deux armes à la fois.

1. *Conception UML :* modifier votre diagramme de classes afin de prendre en compte ces nouvelles fonctionnalités en vous inspirant du pattern décorateur.
2. *Implémentation Java :* ajouter les classes java nécessaires à la gestion de ces options au jeu des personnages.

III - Les patterns de création

Nous allons maintenant essayer de contrôler et/ou faciliter la création d'instance des différents éléments de notre jeu. L'objectif du jeu des personnages est d'être le premier joueur à trouver et mettre la main sur le fabuleux et *unique* trésor, objet de toutes les convoitises.

Exercice III.1

Un unique trésor

1. *Conception UML :* modéliser cette fonctionnalité dans votre diagramme de classes.
2. *Implémentation Java :* À l'aide du pattern singleton, implémenter une classe modélisant le trésor telle qu'elle respecte, par construction, la règle énoncée ci-dessus : le trésor doit être *unique*.
3. À l'aide des tests unitaires, assurez-vous que cette règle est bien respectée.

Exercice III.2

La fabrique... de personnages !

Tout est dans le titre ! !
En effet, nous souhaitons faciliter la vie des utilisateurs de nos différents personnages.

1. *Conception UML :* en utilisant les pattern fabrique et fabrique abstraite, proposer une nouvelle version de votre diagramme de classes afin de prensre en compte cette fonctionnalité.
2. *Implémentation Java :* mettre en œuvre cette évolution dans votre code java.

À suivre...
‡ ‡ ‡