UML – TP noté (jeudi 14 avril 2011)

Durée : 1H30 en binôme (maximum)

Travail à rendre : un document papier ou PDF contenant l'analyse UML du problème suivant en utilisant OCL et les diagrammes UML suivants : classes, use cases et séquences.

Descrition du problème :

(A : à traiter par tous | B : uniquement en binôme | C : n'est pas à traiter)

1. **Introduction**
   - **A.** On s'intéresse à un jeu vidéo contenant des personnages devant effectuer des tâches. Certains personnages peuvent être aidés par des lutins pour exécuter une tâche, ce qui leur permet d'effectuer une sieste pendant que le lutin travaille à leur place.

2. **Les joueurs**
   - **A.** A chaque joueur est assigné un personnage. Un joueur particulier sera le maître du jeu ; il pourra effectuer les mêmes actions qu'un joueur normal mais il aura la responsabilité de la bonne conduite du jeu hors du logiciel.

3. **Les éléments du jeu**
   - **A.** Un personnage a un nom et un avatar permettant de le représenter visuellement. Il peut exécuter une tâche, travailler pendant un temps donné et dormir pendant une certaine durée. N'importe qui peut lui demander d'exécuter une tâche mais seulement lui décidera de travailler ou de faire la sieste.
   - **A.** Une tâche a un libellé et une durée exprimée en nombre d'heures. Une tâche est initialement à faire puis passera en statut en cours et sera finalement marquée comme finie une fois exécutée.
   - **A.** Un lutin est un être qui peut rendre service à un personnage (et un seul). Il a une certaine capacité exprimée en points. A chaque service rendu, le nombre de points est diminué d'une quantité fixe et propre à chaque lutin. Cette quantité est alors retournée au bénéficiaire du service. Bien entendu si le nombre de points du lutin est insuffisant, le lutin garde son capital et ne rendra aucun service (le demandeur recevra 0 point). Un lutin retrouve une partie de sa capacité en dormant ; il se renfloue alors d'un montant fixe et toujours propre à lui ; il ne peut toutefois dépasser une capacité maximale qui est commune à tous les lutins.
   - **B.** Un lutin facétieux est un lutin un peu spécial qui de temps en temps va porter préjudice à celui qui lui a demandé service. Il lui renvoie à ce moment là une valeur négative correspondant à l'opposé de ce qu'un lutin normal aurait fourni. Mais comme il est malin, il se rattrape en étant plus efficace les autres fois, en rendant un service qui vaut deux fois le nombre de points que cela lui coûte. Un lutin facétieux a les mêmes caractéristiques qu'un lutin de base, avec en plus un taux de mauvaises blagues (fixe par lutin). Par exemple si ce taux est de 5, le lutin rendra 4 fois un super service positif, une fois un service négatif puis à nouveau 4 positifs, un négatif, etc …

4. **Fonctionnement du jeu**

B. Le maître du jeu initialise une partie en donnant au jeu un nombre de joueurs, un nombre de tâches par joueur et un nombre de lutins. Le nombre de lutins doit être inférieur au nombre de joueurs. Le jeu crée alors un personnage par joueur et lui associe au hasard un lutin ou un lutin facétieux ou aucune aide suivant le nombre total de lutins disponibles. Le jeu crée ensuite le nombre nécessaire de tâches à faire. Les personnages ont un ordre de passage pour le jeu qui maintient une référence vers le personnage correspondant au joueur courant. Le jeu démarre avec le 1er personnage créé.

C. Les noms des personnages, les libellés et durées des tâches sont puisées dans des catalogues et seront étudiées dans une phase ultérieure de l'analyse.

A. A chaque tour de jeu, le maître du jeu s'assure qu'un joueur différent utilise le logiciel. Le joueur peut alors lancer son tour. Le logiciel choisit une tâche encore à faire et demande au personnage courant de l'exécuter. Si le personnage a la chance d'avoir un lutin associé, il lui demande un service qui lui permettra d'exécuter une partie de la tâche. Le nombre de points reçus est converti en nombre d'heures qui sera déduit de la durée de réalisation de la tâche. S'il reste encore une partie de la tâche à effectuer, le personnage décide alors de travailler pendant le temps nécessaire à compléter la tâche.

B. Bien entendu, si le personnage a affaire à un lutin facétieux qui lui donne un nombre de points négatifs, le personnage mettra d'autant plus de temps à réaliser sa tâche. Le personnage profitera d'une éventuelle déduction de temps de travail pour faire la sieste avant ou après avoir travaillé.

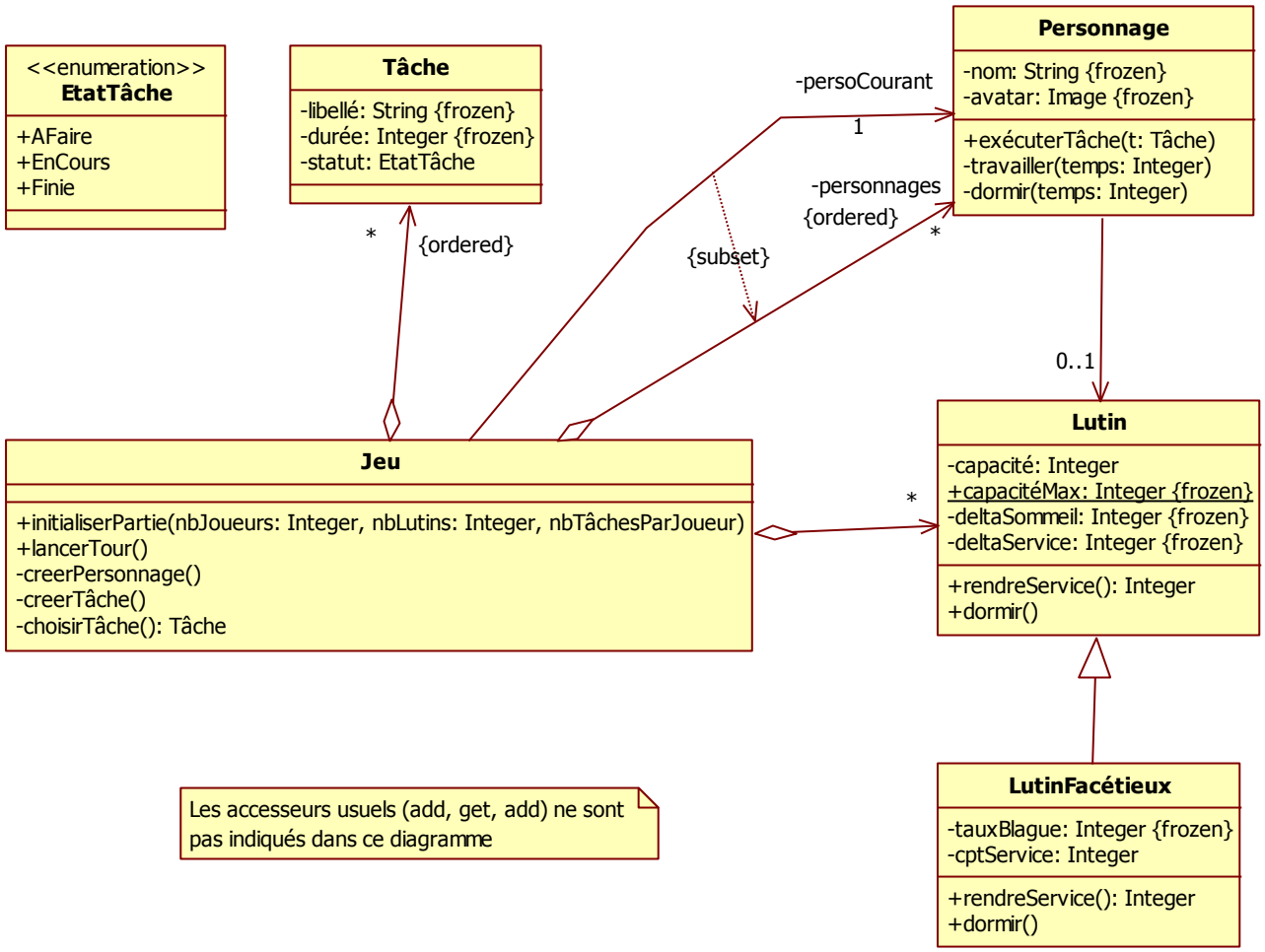
C. Les autres aspects du jeu ne sont pas à traiter : IHM du jeu, sommeil des lutins, passage au personnage suivant, etc …

NB : Les lignes vertes sont des ajouts à l'énoncé d'origine

Eléments de correction

Diagramme de classes

On considère en plus des types UML de base, le type Image.



Contraintes OCL

context Tâche::statut : EtatTâche
init : EtatTâche::AFaire

context Jeu::initialiserPartie(nbJoueurs : Integer, nbLutins : Integer, nbTâchesParJoueur : Integer)
pre : nbLutins < nbJoueurs
post : self.personnages→size() = nbJoueurs
post : self.lutin→size() = nbLutins = self.personnages→collect(lutin)→select(l|l→notEmpty())→size()
post : self.personnages→collect(lutin)→select(l|l→notEmpty())→includesAll(self.lutin)
post : self.lutin→includesAll(self.personnages→collect(lutin)→select(l|l→notEmpty()))
post : self.tâche→size() = nbJoueurs * nbTâchesParJoueur

context Jeu::choisirTâche() : Tâche
pre : self.tâche→exists(statut = EtatTâche::AFaire)
post : result.statut = EtatTâche::AFaire

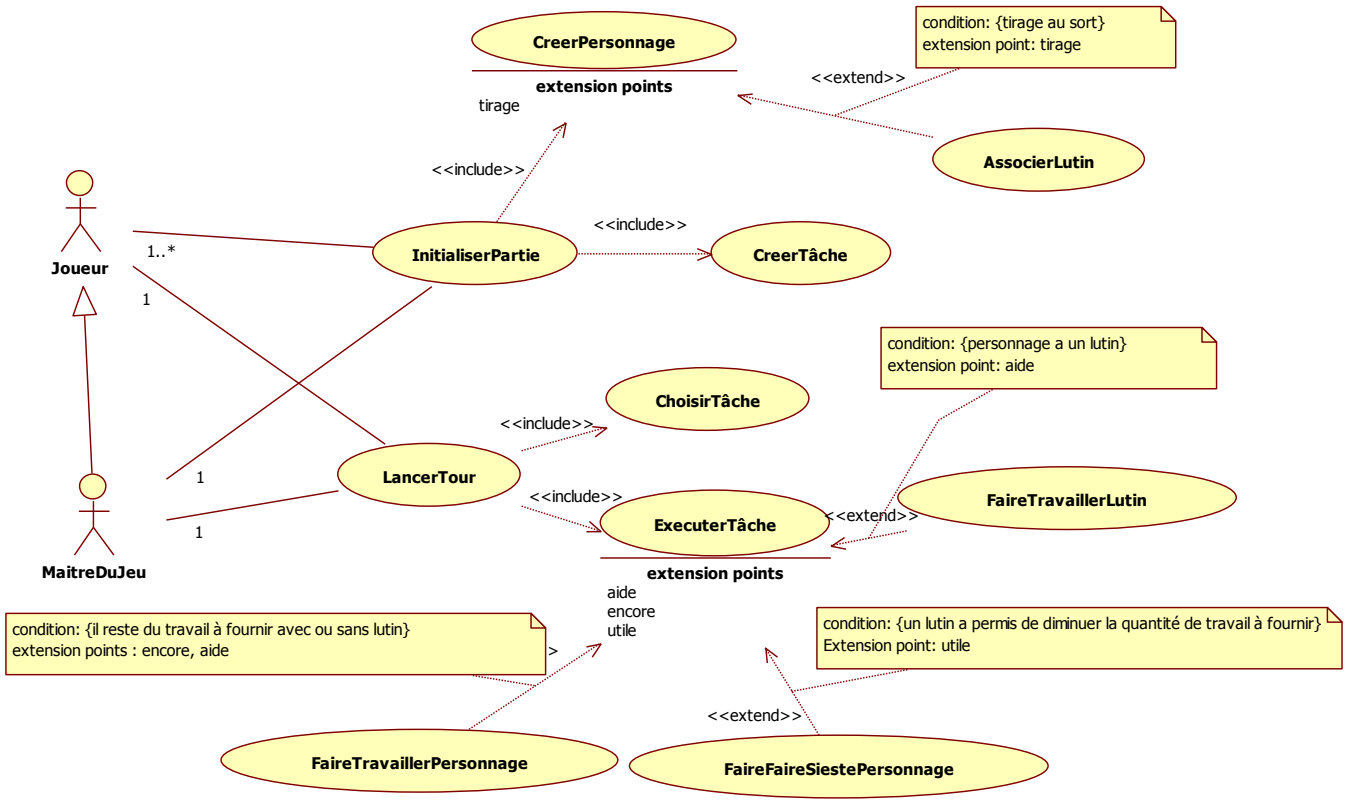
context Personnage::exécuterTâche(t : Tâche)
pre : t.statut = EtatTâche::AFaire
post : t.statut = EtatTâche::Finie

context Lutin
inv : 0 ≤ capacité ≤ Lutin.capacitéMax

context Lutin::rendreService() : Integer
post : result =
if self.capacité≥self.deltaService
then self.deltaService
else 0
endif
post : self.capacité = Max(self.capacité@pre – self.deltaService, 0)

context Lutin::dormir()
post : self.capacité = Min(self.capacité@pre + self.deltaSommeil, Lutin.capacitéMax)

Diagramme de Use Cases



UC InitialiserPartie
Acteurs : 1 MaîtreDuJeu (primaire), n Joueurs (Secondaire)

UC LancerTour
Acteurs : 1 Joueur (primaire), 1 MaîtreDuJeu (Secondaire)
Précondition : une partie a été crée, il reste au moins une tâche à faire ; le personnage courant correspond au joueur s'apprêtant à jouer (contrôle du Maître du Jeu)
Postcondition : une tâche supplémentaire est finie

Diagramme de sequences

Ils correspondent aux scenarii des UC ci-dessus. On y retrouve les différents points d'extension dans des fragments **alt** ou **opt**.

(cf annexe)