Examen Analyse Orientée Objet - ING1

Lundi 3 juin 2013

Durée : 2h

Partie 1 : Diagramme de cas d'utilisation et diagramme de classes

CityPASS est un système de vente de tickets qui permettent de visiter les attractions touristiques les plus connues dans une ville avec un prix plus économique. Chaque ville correspond à une liste des attractions dont on connait le nom, l'adresse, la description et les horaires d'ouverture. Ces horaires dépendent de la saison et du jour de la semaine. Par exemple, le Metropolitan Museum of Art à New York est ouvert tous les dimanches, mardis, mercredis et jeudis entre 9h30 et 17h30 et tous les vendredis et samedis entre 9h30 et 21h, hiver comme été. Attention, pour le même jour et la même attraction, l'heure de fermeture d'hiver doit être toujours plus tôt que celle d'été.

Le nombre d'attractions d'une ville est limité entre 3 et 5. Par contre, on peut avoir des attractions alternatives : à New York, on peut visiter Top of the Rock ou le musée Guggenheim. Ceci donne éventuellement une liste de 7 attractions parmi lesquelles il faut en choisir 5 pendant la visite.

En ce qui concerne les clients, ils peuvent aller sur le site internet de CityPASS pour effectuer plusieurs actions. Pour consulter les informations concernant les attractions d'une ville, il faut d'abord sélectionner la ville. Quand le client décide d'acheter, il choisit la quantité de CityPASS pour les adultes et celle pour les enfants, puis les ajoute au panier. Le paiement peut être effectué tout de suite ou le client peut revenir sur le site plus tard pour payer, car son panier est toujours sauvegardé pour un temps donné.

Pour gérer les clients, il faut stocker leur nom, leur prénom, leur email et leur adresse. Chaque achat correspond à une date et à un prix total qui est calculé à partir du prix unitaire pour un CityPASS adulte et celui d'un CityPASS enfant. Bien sûr, ces prix unitaires changent en fonction de temps.

Question 1. Proposer le diagramme de cas d'utilisation pour le logiciel en ligne de CityPASS.

Question 2. Élaborer un diagramme de classes pour décrire au mieux le système. Ajouter toutes les contraintes OCL nécessaires pour compléter le diagramme.

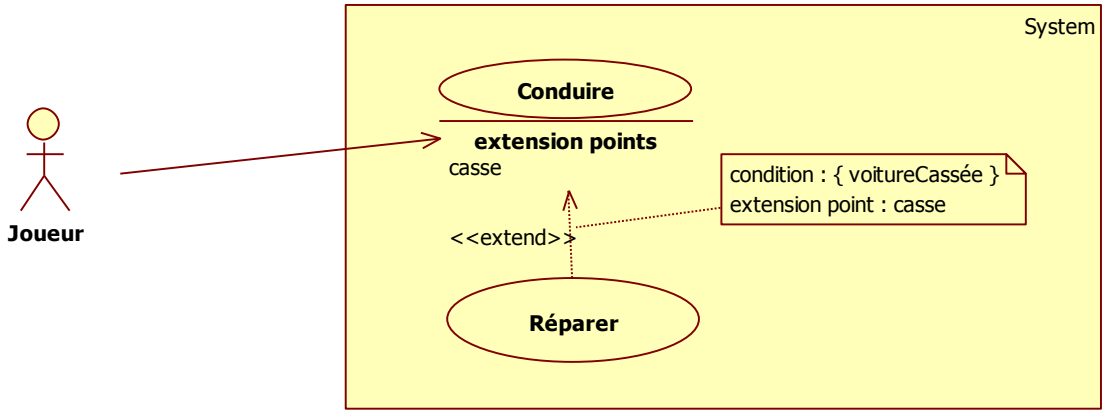
Partie 2 : Diagramme d'activité

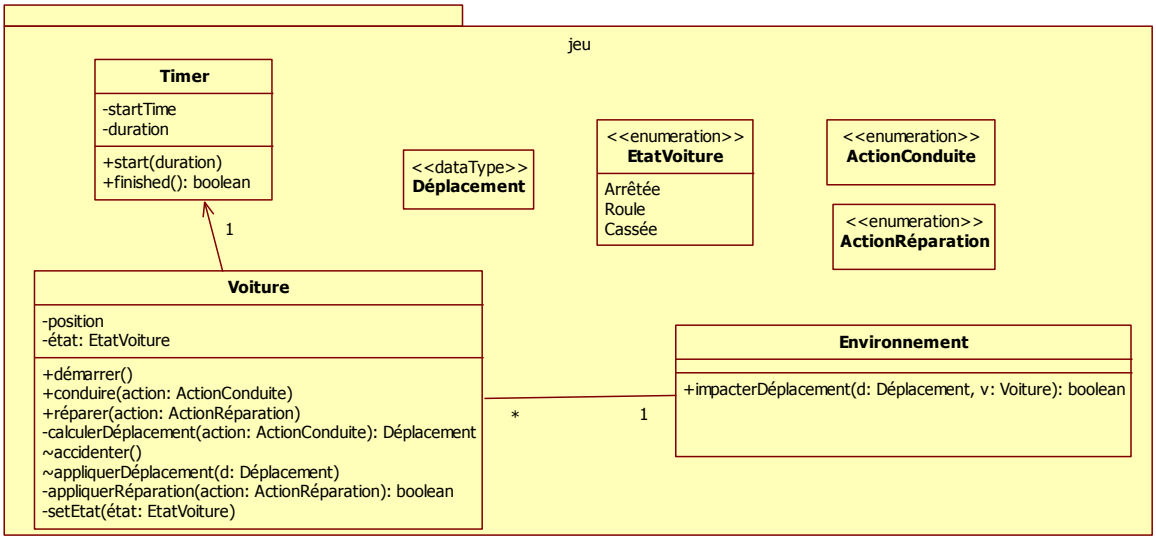
Le processus d'envoi de colis est le suivant. L'expéditeur vient déposer son colis à La Poste. Un employé de La Poste l'enregistre. Le colis est transporté au bureau de poste le plus proche du destinataire. Le facteur distribue le colis. Si, lors de son passage, le destinataire est présent, ce dernier réceptionne le colis. Dans le cas contraire, le facteur dépose un avis de passage et le colis est stocké à La Poste pour une durée maximale de 15 jours. Si durant ces 15 jours le destinataire vient avec son avis de passage, alors un employé de La Poste valide la réception du colis et le destinataire peut récupérer son colis. Sinon, le colis est retransféré chez l'expéditeur.

Question 3. Construire un diagramme d'activité pour modéliser le processus de l'envoi d'un colis avec les différents acteurs.

Partie 3 : Diagramme de séquence et d'état

On considère un logiciel de simulation de conduite de voiture pédagogique. En cas de mauvaise conduite et d'accident, le joueur doit réparer son véhicule pour reprendre le jeu. On vous donne une vue partielle du diagramme des cas d'utilisation et du diagramme de classes de ce logiciel.





Le scénario du cas d'utilisation **Conduire** est le suivant :

1. le joueur démarre sa voiture
2. la voiture passe alors à l'état *Roule* et le timer est démarré pour une durée de 60 secondes
3. le joueur peut alors passer une série d'ordre de conduite ou de réparation à sa voiture tant qu'elle n'est pas à l'arrêt
4. si la voiture roule, un ordre de réparation est ignoré et un ordre de conduite entraîne un calcul interne de déplacement suivant l'action demandée puis une demande à l'environnement de l'impact de ce déplacement sur la voiture.
5. la demande d'impact provoque sur la voiture soit une application de ce déplacement si le déplacement est correct vis-à-vis de l'environnement soit un accident de la voiture dans le cas contraire.
6. un accident passe la voiture dans l'état *Cassée*.
7. si la voiture est cassée, le scénario *Réparer* est suivi.

8. suite au traitement d'un ordre du joueur (conduite ou réparation), le timer est interrogé sur sa terminaison
9. si le temps est terminé, la voiture passe à l'arrêt.

Une condition d'initiation de ce use case est qu'une voiture dans l'état initial *Arrêtée* est attribuée à chaque joueur. On ne détaille pas ici le type d'accident sur la voiture ni le processus de décision de l'environnement vis-à-vis d'un déplacement.

Le scénario du cas d'utilisation **Réparer** est le suivant :

1. un ordre de conduite est ignoré et un ordre de réparation entraîne l'application de cette réparation sur la voiture
2. la réparation peut se terminer correctement et repasse la voiture en état *Roule*
3. dans le cas contraire, la voiture reste dans l'état *Cassée.*

Question 4. Etablir les 2 diagrammes de séquence correspondant aux 2 cas d'utilisations ci-dessus.

Question 5. Etablir le diagramme d'états de la classe Voiture.