M1 : Ingénierie du Logiciel

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Partiel de novembre 2005

L'équipe enseignante du module

2 heures avec documents

0. Préambule : A lire avant de commencer le partiel

Attention ! Toute réponse à une question de l'épreuve sans justification ou rédigée en ne respectant pas la syntaxe d'UML sera notée pour 0 point.

Bonne chance !

1. Questions de cours [5 Pts]

Répondre à chaque question de manière précise et concise. Votre réponse ne doit pas dépasser une dizaine de lignes. Chacune des questions a le même poids.

Q1.1

Est-il possible, uniquement avec un diagramme de cas d'utilisation, de spécifier qu'un cas d'utilisation doit s'exécuter obligatoirement avant un autre cas d'utilisation ? Précisez votre réponse.

Non, cela n'est pas possible car aucune relation entre cas d'utilisation (héritage, inclusion et extension) ne permet d'établir un ordre de séquence entre les cas. Pour spécifier un ordre, il faut utiliser les fiches détaillées (pré-condition).

Q1.2

UML est-il adapté au cycle en V ? Quelles sont les étapes de ce cycle qu'il ne couvre pas ?

UML est adapté au cycle en V. Les diagrammes de cas d'utilisation, de classe et de séquence sont utilisés pour la phase d'analyse. Les diagrammes de cas d'utilisation de classe, de séquence, d'activité sont utilisés pour la phase de conception. Les diagrammes de classes, d'états sont utilisés pour la phase de réalisation. UML peut être utilisé pour rédiger les tests d'intégration et les tests unitaires.

UML ne peut pas être utilisé pour rédiger les tests de validation. En effet ceux-ci doivent être lu par le client. Les notes de code doivent être intégrées dans le modèle pour permettre la génération de code. On peut donc considérer que les notes de code ne sont pas UML.

Q1.3

Est-il possible de suivre le cycle en V sans utiliser UML ?

Evidemment, le cycle en V est un processus qui n'a pas besoin d'UML. D'autres langages de modélisation ou de spécification peuvent être utilisés.

Q1.4

Donner un exemple réaliste et original d'héritage entre acteurs dans un diagramme de cas d'utilisation.

Pour justifier un héritage entre acteurs il faut évidemment disposer de deux acteurs mais il faut aussi disposer de 2 cas d'utilisation (un cas sera lié aux deux acteurs alors que l'autre cas ne sera lié qu'à un acteur).

Q1.5

La phase d'analyse est-elle indépendante ou dépendante de la plate-forme d'exécution sur laquelle s'exécutera l'application ?

Une analyse reprend les informations données dans le cahier des charges. Nous avons précisé en cours que nous ne traitions que les informations fonctionnelles. Donc, nous pouvons considérer qu'une analyse ne dépend pas de la plate-forme d'exécution.

2. Problème: SpeedBoat [15 Pts]

La pratique du bateau à voile est difficile car il faut savoir composer avec vents et courants pour faire avancer le bateau. Grâce aux techniques actuelles de prévision météorologique, au système GPS et à la précision des cartes du SHOM, la société MyBoat a décidé de construire un logiciel, nommé SpeedBoat, permettant de calculer automatiquement la route du voilier. Il vous est demandé de réaliser une partie de la phase d'analyse ainsi qu'une partie de la phase de rédaction des tests de validation de cette application grâce aux extraits du cahier des charges suivants :

« Le logiciel SpeedBoat devra pouvoir fonctionner dans n'importe quelle région navigable du globe terrestre. Le système de carte nautique devra donc être indépendant de tout système de carte nautique existant. Ce système de carte devra supporter les notions suivantes : coordonnées GPS de tous points de la carte à une précision de l'ordre du mètre (longitude et latitude), zone (terre – mer – zone découvrante), lignes de sonde (2, 5, 10, 20 et 50m de profondeur), bouées (cardinales, danger isolé, eau saine, zone militaire, chenal), amères (phares, églises, château d'eau). Toutefois, un système d'importation des cartes nautiques du SHOM devra être développé supportant le format électronique SHOM en vigueur à date de livraison du logiciel. Les cartes importées par ce module ne devront pas pouvoir être modifiées. »

« Le logiciel SpeedBoat devra pouvoir intégrer les prédictions de marées ainsi que les forces de courant sur n'importe quelle carte nautique. Il devra donc être possible d'associer des tableaux de prédictions de marées aux cartes nautiques. Un tableau de prédictions de marées contient, pour chaque jour de l'année, les heures de marée haute et basse, le coefficient de marée ainsi que la hauteur minimale de la mer. Il devra aussi être possible de préciser des vecteurs courants sur certaines zones des cartes nautiques en fonction des périodes des marées. Une zone sur une carte se définit à l'aide d'au moins trois points GPS. Un vecteur courant se définit par une direction (en degré) et une force (en nœud). »

« Le logiciel SpeedBoat devra pouvoir intégrer automatiquement les informations météorologiques fournies par le site web de Météo France (http://www.meteofrance.com/FR/mer/selectCote.jsp). Ces informations météorologiques sont attachées à une carte nautique pour une durée de quatre jours au maximum et par période de 6 heures. Les informations météorologiques consistent à spécifier les vents, les vagues ainsi que la situation nuageuse sur certaines zones de la carte. Les vents sont définis par leur direction et leur force (en Beaufort). Les vagues sont définies par leur hauteur. La situation nuageuse par le type de nuages présents dans la zone (cumulus, stratus). Il devra aussi être possible de préciser s'il y a lieu, la position de fronts (chaud, froid et occlus) par un ensemble de coordonnées GPS.

« Le calcul d'une route fond par le logiciel SpeedBoat se fait en plusieurs étapes. L'utilisateur doit d'abord sélectionner la carte nautique du trajet parmi les cartes actuellement disponibles. Puis il doit préciser la date du trajet. Puis il peut effectuer, si cela est possible, une intégration des informations météorologiques. Puis il doit saisir le point de départ ainsi que le point d'arrivée. Ces informations sont suffisantes pour demander à SpeedBoat de calculer la route fond ainsi que les caps à suivre. La route fond est définie par un ensemble de points GPS. Les caps associés à la route fond sont définis par une direction (en degrés) entre deux points GPS successifs de la route. Si aucune information météo n'a été récupérée (direction et force du vent), SpeedBoat ne sera pas en mesure de calculer les caps à suivre, il ne fournira alors que la route fond. Si aucune information n'a été donnée sur les marées ainsi que sur les courants, SpeedBoat considérera que ces informations sont négligeables. »

« Afin de pouvoir être utilisé aussi bien en croisière qu'en compétition, il devra être aussi possible de saisir des points de passage lors de la définition d'un trajet, avant de demander le calcul de route. La route fond proposée par SpeedBoat passera alors par les points de passage. »

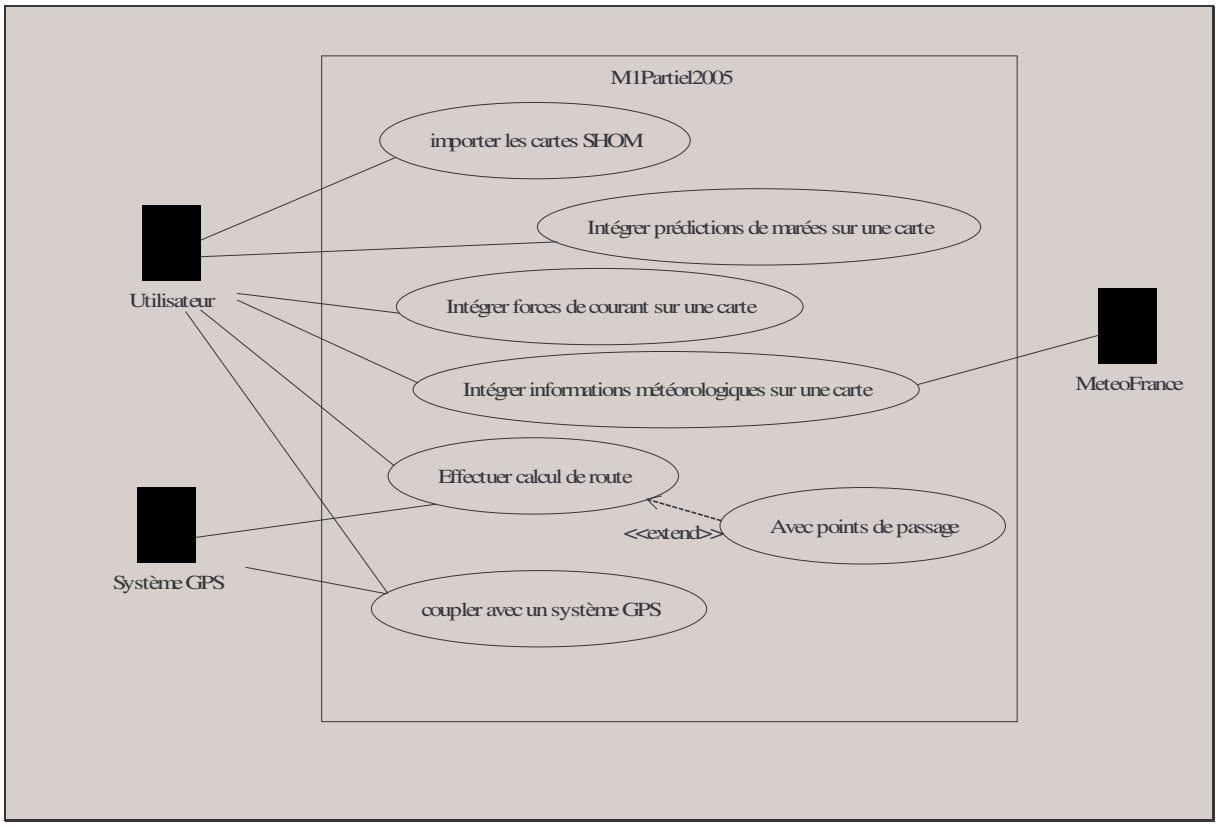
« SpeedBoat devra pouvoir être couplé au système GPS des bateaux afin d'effectuer une synchronisation de la position du bateau sur une carte nautique. Si SpeedBoat est couplé au GPS, il n'est alors pas nécessaire de saisir le point de départ d'un trajet lors d'un calcul d'une route fond ; celui-ci est remplacé par la position du bateau rendue par le GPS. Lors d'un couplage, il est aussi possible de demander une surveillance de la position du bateau par rapport à la route calculée. Si l'écart excède 1 mile nautique, SpeedBoat émet un message d'alerte et propose une correction de cap pour revenir sur la route. Une telle surveillance ne peut être demandé que si une route fond a été calculée. Il n'est alors pas possible d'effectuer un nouveau calcul de route fond sans avoir, au préalable, désactivé la surveillance. »

« SpeedBoat devra pouvoir être installé sur n'importe quel ordinateur Mac, PC Windows ou PC Linux. Il ne devra pas consommer plus de 256Mo de mémoire vive en exécution. Il ne devra pas demander plus de 100Mo en place disque (hors du poids de chaque carte nautique). »

« La communication de SpeedBoat avec le système GPS du bateau devra se faire soit en Bluetooth, soit par cable USB, soit par cable série. ».

Q 2.1 (4 Pt)

Faites le diagramme de cas d'utilisation de SpeedBoat. Vous justifierez tous les acteurs, les cas d'utilisation et les relations entre ceux-ci.



Le cahier des charges n'est pas très explicite sur l'acteur qui bénéficie des fonctionnalités offertes par SpeedBoat. Seul une phrase parle de l'utilisateur. On considère alors qu'il n'y a qu'un acteur qui bénéficie des fonctionnalités : l'utilisateur.

Le système GPS interagit avec le système. C'est donc lui aussi un acteur.

MétéoFrance interagit avec le système. C'est donc lui aussi un acteur.

L'importation des cartes nautiques du SHOM est clairement précisé dans le premier paragraphe du cahier des charges (« un système d'importation des cartes nautiques du SHOM … »).

L'intégration des prédictions de marée est clairement précisé : « SpeedBoat devra pouvoir intégrer les prédictions de marée … »

L'intégration des forces de courant est précisé : « … ainsi que les forces de courant »

L'intégration des informations météorologiques est précisé « intégrer automatiquement les informations météorologiques … »

Le calcul de route fond est le cas d'utilisation central de SpeedBoat. Nous avons fait le choix de ne pas faire apparaître

Q 2.2 (2 Pt)

Faites la fiche détaillée (présentée en cours et en TD) du cas d'utilisation correspondant au calcul de la route fond.

Fiche Détaillée.

Nom : Effectuer calcul de route fond

Acteurs : Utilisateur, Système GPS

Description : Un utilisateur demande à SpeedBoat de calculer une route fond après avoir fourni toutes les informations nécessaires (carte, départ, arrivé, courant, météo, ...)

Pré-condition : Aucune

Hypothèse : au moins une carte est présente dans le système

Suite des interactions :

1) Selectionner une carte nautique
2) Préciser la date du trajet

3) intégration des informations météo (si cela est possible)

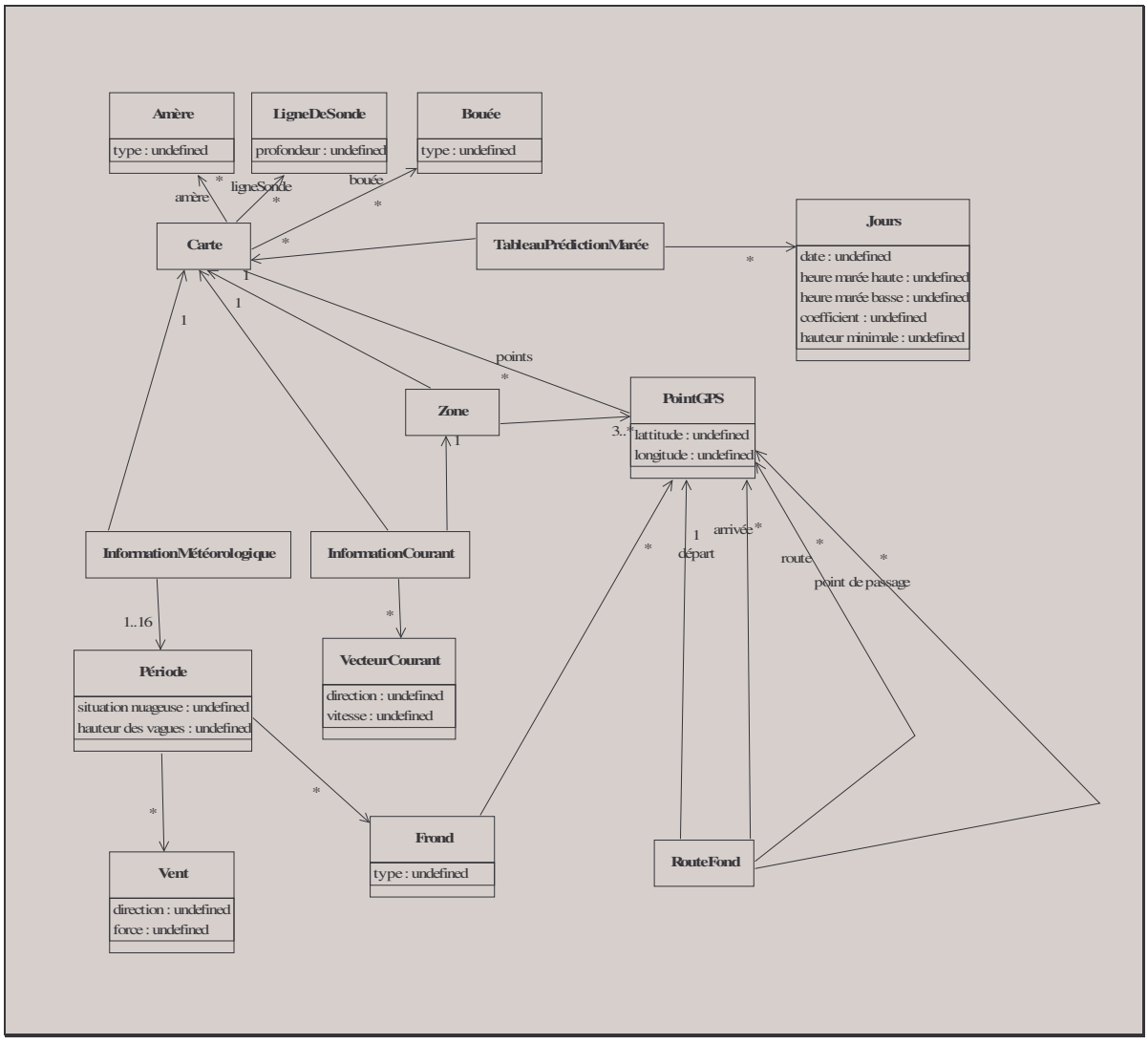
5) Définir point de départ (ou acquisition à l'aide du système GPS)
6) Définir point d'arrivé
7) Demander calcul de la route fond

Post-Condition :
La route fond est construite

Q 2.3 (5 Pts)

Faites le diagramme de classe d'analyse de SpeedBoat. Vous justifierez toutes les classes, tous les attributs, toutes les opérations et toutes les associations.

.



Chaque concept du cahier des charges est représenté soit par une classe soit par un attribut soit par une association :

Concepts représenté sous forme de classes :

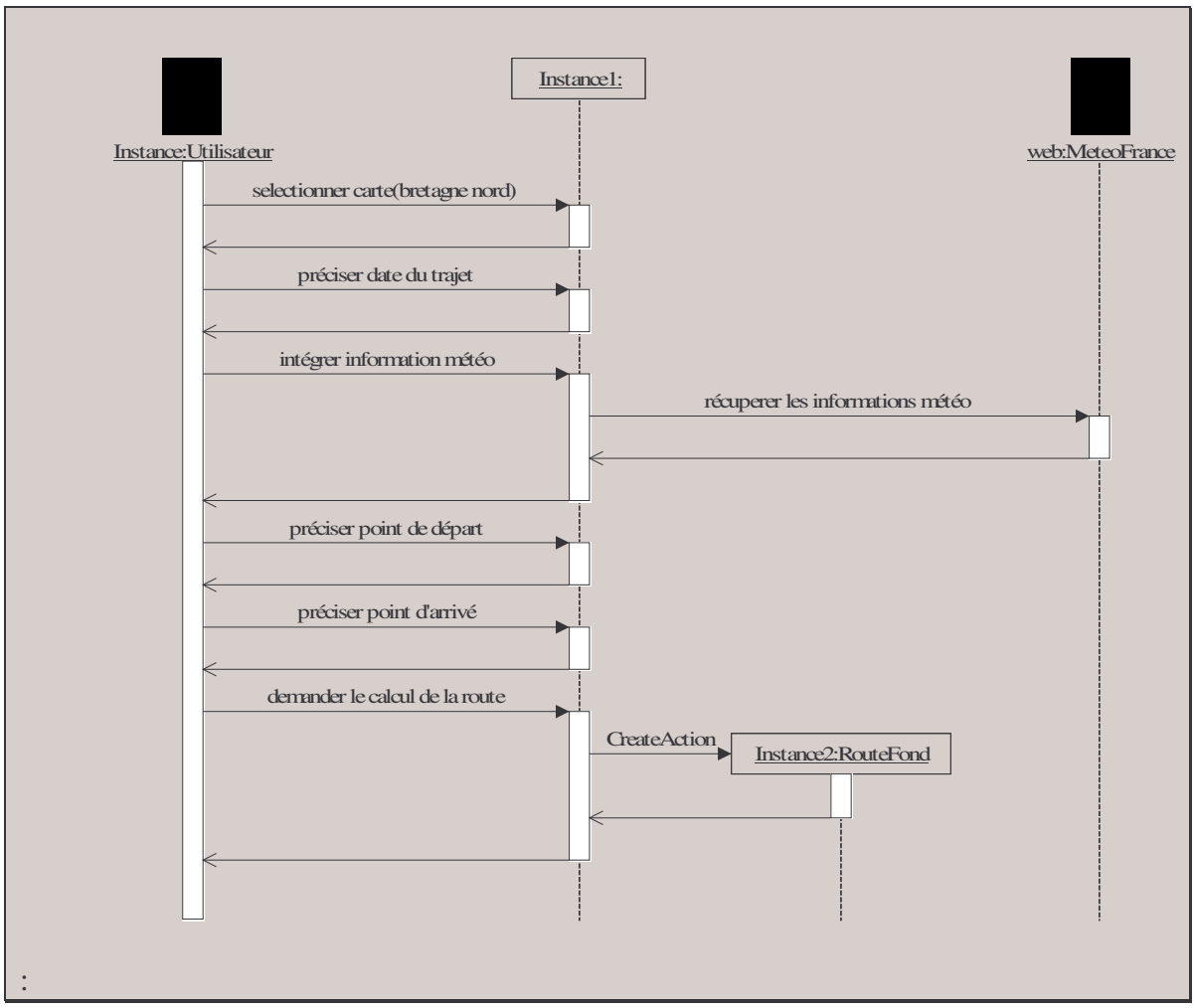
Carte, PointGPS, LigneDeSonde, Bouée, Amère, InformationMétéorologique, TableauPrédictionMarée, Jours, Zone (d'une carte), InformationCourant, VecteurCourant, Période (pour la météo), Vent, Frond.

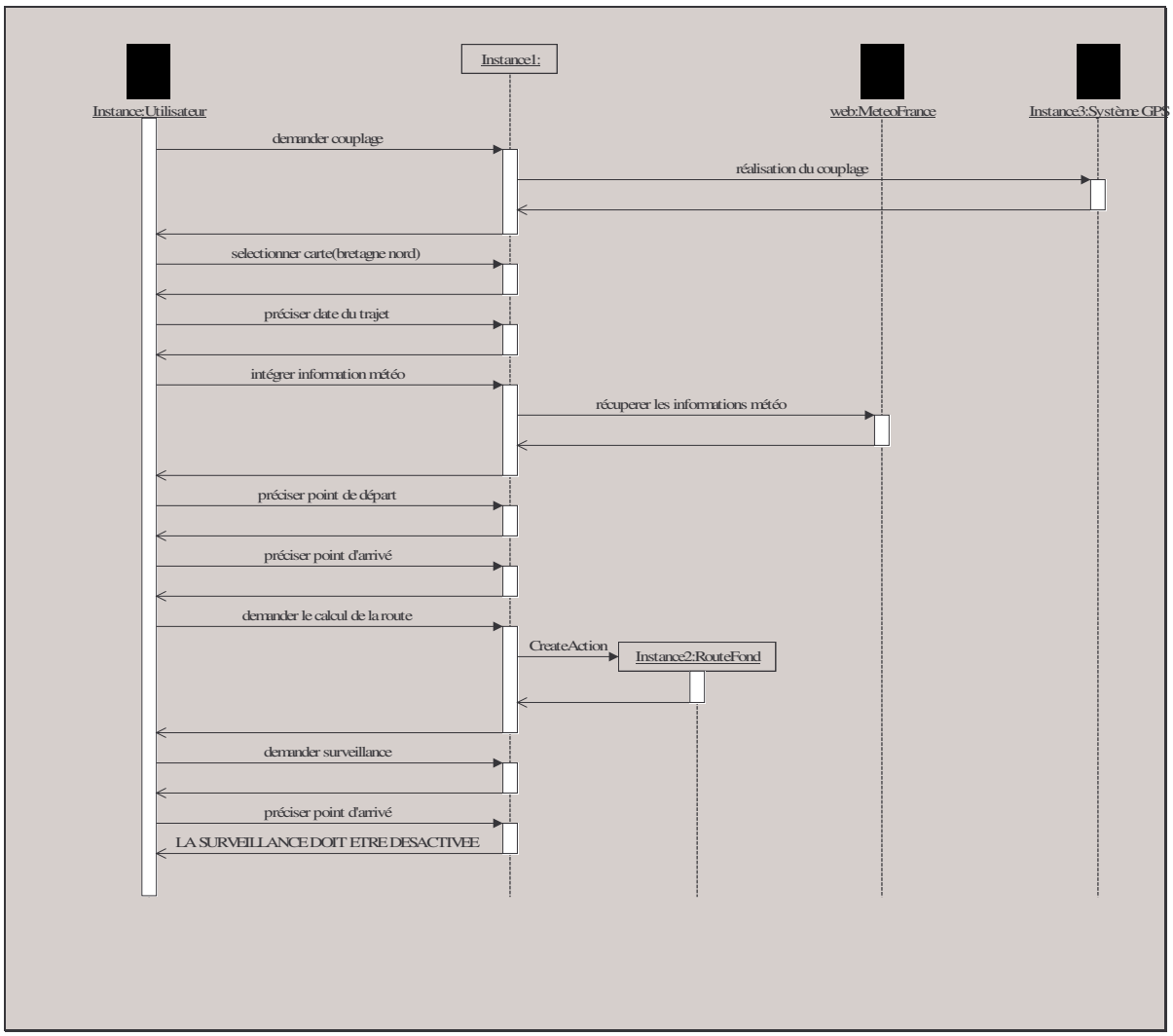
Il faut pouvoir retrouver les informations telles que :

Une carte possède tous les points GPS. Une route fond a un point de départ, un point d'arrivée et des points de passages…

Q 2.4 (2 Pt)

Faites deux diagrammes de séquences décrivant le cas d'utilisation correspondant au calcul de la route fond (un avec erreur et un nominal).


Instance1:
Instance:Utilisateur
web:MeteoFrance
selectionner carte(bretagne nord)
préciser date du trajet
intégrer information météo
récuperer les informations météo
préciser point de départ
préciser point d'arrivé
demander le calcul de la route
CreateAction
Instance2:RouteFond



Q 2.5 (2 Pts)

Faites deux tests de validation avec erreurs et deux tests de validation nominaux de l'application SpeedBoats.

1 test de validation et 1 test d'erreur suffisent ☺ Cette boulette avait pourtant été levée par Olivier mais elle subsiste.

Titre : Calcul de route fond

Context : l'utilisateur veut construire une route fond sur la carte nommée « nord cotentin » entre Cherbourg et Aurigny. SpeedBoat doit posséder la carte nommée « nord cotentin ». SpeedBoat doit pouvoir se connecter au site web de FranceMétéo.

Entrée : le coordonnées GPS de Cherbourg et les coordonnées CPS d'Aurigny.

Scénario :

L'utilisateur démarre SpeedBoat

L'utilisateur choisi de calculer une nouvelle route fond

L'utilisateur sélectionne la carte du nord cotentin

L'utilisateur saisie la date de son trajet (date du jours)

L'utilisateur demande à SpeedBoat d'intégrer les informations météorologiques

SpeedBoat lui indique que les informations météorologiques ont été saisies.

L'utilisateur précise son point de départ (cherbourg)

L'utilisateur précise son point d'arrivé (aurigny)

L'utilisateur demande à SpeedBoat de calculer la route fond.

SpeedBoat lui présente la route fond calculée.

Moyen de vérif : Visuel

Titre : Calcul de route fond, couplage et surveillance et re-calcul de route fond

Context : l'utilisateur veut construire une route fond sur la carte nommée « nord cotentin » entre Cherbourg et Aurigny. SpeedBoat doit posséder la carte nommée « nord cotentin ». SpeedBoat doit pouvoir se connecter au site web de FranceMétéo. L'utilisateur doit posséder un système GPS qui peut être couplé avec SpeedBoat

Entrée : le coordonnées GPS de Cherbourg et les coordonnées CPS d'Aurigny.

Scénario :

L'utilisateur démarre SpeedBoat

L'utilisateur choisi de calculer une nouvelle route fond

L'utilisateur sélectionne la carte du nord cotentin

L'utilisateur saisie la date de son trajet (date du jours)

L'utilisateur demande à SpeedBoat d'intégrer les informations météorologiques

SpeedBoat lui indique que les informations météorologiques ont été saisies.

L'utilisateur précise son point de départ (cherbourg)

L'utilisateur précise son point d'arrivé (aurigny)

L'utilisateur demande à SpeedBoat de calculer la route fond.

SpeedBoat lui présente la route fond calculée.

L'utilisateur demande à SpeedBoat de se coupler au système GPS

L'utilisateur demande ensuite une surveillance de la part de SpeedBoat

L'utilisateur reprécise son point d'arrivé.

Resultat avec Erreur : SpeedBoat indique qu'il n'est pas possible de modifier le point d'arrivé tant que la surveillance n'est pas désactivée.

Moyen de vérif : Visuel

M1 : Ingénierie du Logiciel

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Partiel de novembre 2006 (2 heures avec documents)

1. Questions de cours [5 Pts]

Q1.1 : Pourquoi est-il déconseillé de spécifier les opérations des classes métiers (autre que les opérations d'accès aux propriétés des classes) ?

Les classes métiers représentent les caractéristiques communes des concepts métiers définis dans le cahier des charges. Il ne faut pas confondre classe métier avec la classe d'analyse qui correspond au système. De ce fait, les classes métiers ne réalisent pas de traitements. Elles ne sont pas responsables. L'assignation de la responsabilité se fera en conception.

Barème :

100% si mention de la notion de responsabilité. Les classes métiers ne sont pas responsable.

0% sinon

Q1.2 : Comment exprimer une relation d'ordre entre deux cas d'utilisation (lorsqu'il faut absolument réaliser un cas avant de pouvoir en réaliser un autre) ?

La seule façon de le faire est d'utiliser les pré-conditions des fiches détaillées des cas d'utilisation. Il ne faut surtout pas utiliser l'import ou l'extends.

Barème :

50% si fiche

50% si pré-condition

0% sinon

Q1.3 : Pourquoi n'est-il pas raisonnable d'écrire les tests de validation en UML ?

Les tests de validation seront lus par le client et par la personne qui les exécutera. Ces personnes ne sont pas des informaticiens et encore moins des concepteurs UML. Ces tests ne doivent donc pas être écris en UML.

Barème :

100% si mention des lecteurs (client ou testeur)

0% si autre explication sur la lisibilité

Q1.4 : En quoi consistent les phases de « définition des besoins » et de « spécification des besoins » ? Pourquoi considérons-nous la phase de « spécification des besoins » comme étant la phase d'analyse ?

La définition des besoins consiste à la rédaction du cahier des charges. C'est le moment où l'on est en train de réfléchir à bien définir quel est notre besoin. La spécification des besoins consiste à formaliser (spécifier) la définition des besoins. Nous utilisons UML pour formaliser la définition des besoins.

Pour nous la définition des besoins n'est pas du ressort de l'équipe de développement mais du ressort du client. Par contre la spécification des besoins est complètement du ressort de l'équipe de développement. Voilà pourquoi nous considérons que la définition des besoins n'appartient pas à la phase d'analyse alors que la spécification des besoins appartient.

Barème :

50 % si distinction claire entre définition et spécification

50 % si explication sur l'appartenance à la phase d'analyse (ressort de l'équipe de développement ou pas).

Q1.5 : Faut-il spécifier les besoins non-fonctionnels lors de la phase d'analyse ? Est-ce possible en UML ?

La phase d'analyse doit spécifier tous les besoins et pas uniquement les besoins fonctionnels. Il faut donc, en théorie, spécifier les besoins non-fonctionnels. Par contre, cela n'est pas réellement possible avec UML. Voilà pourquoi, en cours, nous ne faisons pas de spécification de besoins non-fonctionnels.

Barème :

50 % si explication sur le fait que la spécification des besoins non-fonctionnels doit appartenir à l'analyse.

50 % sur l'explication avec UML

Le commentaire sur le fait que cela ne soit pas fait en cours ne compte pour rien.

2. Problème: Analyse de eB6 [15 Pts]

Une société désire mettre en place un système de ventes aux enchères, nommé « eB6 » permettant à des millions d'acheteurs et de vendeurs d'acheter et de vendre n'importe quel objet à travers le monde. Vous avez la charge de réaliser la phase d'analyse de ce système. Le cahier des charges est le suivant :

N'importe quel utilisateur peut parcourir le site web d'eB6 sans être inscrit. En effet, les utilisateurs ont uniquement besoin de s'inscrire pour acheter et pour vendre des objets.

Avant de commencer à vendre sur eB6, les utilisateurs doivent ouvrir un compte vendeur pour qu'eB6 vérifie leur sérieux. eB6 propose deux façons d'ouvrir un compte vendeur :

- une façon instantanée sur internet : il suffit de fournir les informations de la carte de crédit du vendeur (la carte ne sera pas débitée, mais uniquement utilisée à des fins de vérification de l'identité du vendeur sauf si vous demandez à eB6 de prélever également les frais de vente sur cette carte) ;

- une méthode par courrier postal qui prend 2 à 3 jours et permet de recevoir un code de confirmation par courrier à la maison.

Quelque soit la façon d'ouvrir un compte vendeur, il faut fournir les coordonnées personnelles du vendeur (nom, prénom, adresse postale, téléphone, email) ainsi que les coordonnées bancaires (soit les informations de la carte bancaire soit un RIB).

Pour vendre un objet, le vendeur doit remplir le formulaire de mise en vente qui donnera les informations utiles à l'acheteur pour savoir s'il souhaite acheter l'objet. Ce formulaire contient:

- un titre clair et accrocheur ;
- une description complète indiquant aussi bien les qualités que les défauts pour éviter les questions et les litiges ;
- une photo fidèle de l'objet ;
- un prix attractif ;
- dans le cas d'une vente par enchère, la durée de la vente (3, 5, 7 ou 10 jours) en pensant à inclure un week-end pour toucher les acheteurs du week-end.
- Les catégories correspondantes à l'objet.
- Les mots-clés correspondants à l'objet

Il existe deux façons de mettre en vente un objet :

- une vente directe qui permet à n'importe quel acheteur d'acheter directement l'objet
- une vente par enchère qui permet aux acheteurs de placer des enchères jusqu'à la fin de la durée de la vente. A la fin de la durée de la vente, l'acheteur ayant placé la dernière enchère pourra alors acheter l'objet.

Le site web eB6 propose deux moyens aux utilisateurs pour trouver un objet :

- Naviguer par catégories : Depuis la page d'accueil du site, il est possible de naviguer dans les catégories et les sous catégorie des objets afin de trouver des objets.
- Naviguer par mot-clé : Depuis la page d'accueil du site, il est possible de taper un mot-clé dans le moteur de recherche. Les objets correspondants au mot-clé seront alors présentés.

Une fois que les utilisateurs ont trouvé l'objet de leurs rêves, ils peuvent, s'ils sont des acheteurs, soit enchérir dessus, soit l'acheter immédiatement en fonction du format de vente choisi par le vendeur de l'objet. Pour s'inscrire, un acheteur doit obligatoirement donner ses informations personnelles et choisir un login et un mot de passe.

Pour enchérir, l'acheteur doit saisir le montant maximum qu'il est prêt à payer pour cet objet. eB6 va enchérir pour l'utilisateur jusqu'à ce montant maximum en cas de surenchère d'un autre acheteur.

A la fin d'une vente (directe ou par enchère), le vendeur peut accepter ou refuser la vente. Si celle-ci est accéptée, elle pourra être considérée comme un contrat qui lie l'acheteur et le vendeur.

Pour acheter un objet immédiatement, si le vendeur propose ce format, il suffit d'accepter le prix du vendeur et l'objet est à acheter immédiatement.

Pour payer un objet que l'acheteur a acheté sur eB6, il faut entrer en contact avec le vendeur, lui demander le montant total (frais de port inclus) s'il n'est pas déjà précisé dans l'annonce et lui envoyer directement l'argent via un des moyens de paiement qu'il accepte. Pour contacter le vendeur, il faut utiliser l'e-mail du vendeur présenté en fin d'enchères. Le vendeur n'enverra l'objet qu'une fois le paiement reçu. Une fois l'objet reçu l'acheteur peut laisser une évaluation au vendeur qui en fera de même pour l'acheteur.

Question 2.1 : Réalisez le diagramme de cas d'utilisation de la phase d'analyse. Vous justifierez tous vos choix. Préciser la feuille détaillée du cas d'utilisation correspondant à une mise en vente directe.

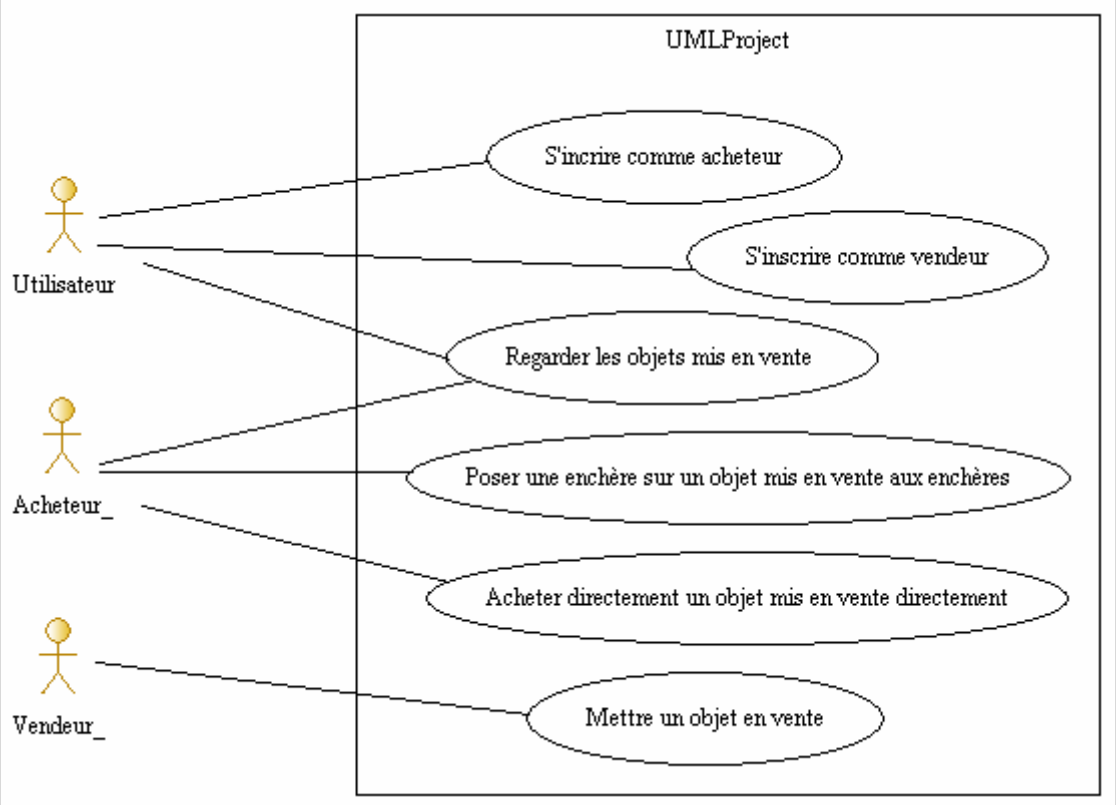


Diagramme de cas d'utilisation :

Les trois acteurs sont l'acheteur, le vendeur et l'utilisateur. Il n'est pas nécessaire de positionner les liens d'héritage entre acheteur, vendeur et utilisateur mais cela n'est pas non plus une erreur.

Les cas d'utilisation absolument fondamentaux : « s'inscrire comme vendeur », « s'inscrire comme acheteur », « parcourir les objets mis à la vente », « déposer un objet à la vente », « acheter un objet », « laisser une évaluation sur le vendeur », « laisser une évaluation sur l'acheteur ».

Il faut soit découper le cas « déposer un objet à la vente » en deux cas « vente directe » et « vente par enchère » soit préciser les différentes variantes dans le texte expliquant le diagramme.

Il faut soit découper le cas « acheter un objet » en deux cas « achat direct » et « achat par enchère » soit préciser les différentes variantes dans le texte.

La communication entre le vendeur et l'acheteur n'est pas un cas d'utilisation.

Il n'y a pas d'administrateur.

L'authentification ne doit pas apparaître.

Barème :

-20% par oubli de cas d'utilisation

-20% par oubli d'acteur

-30% par erreur

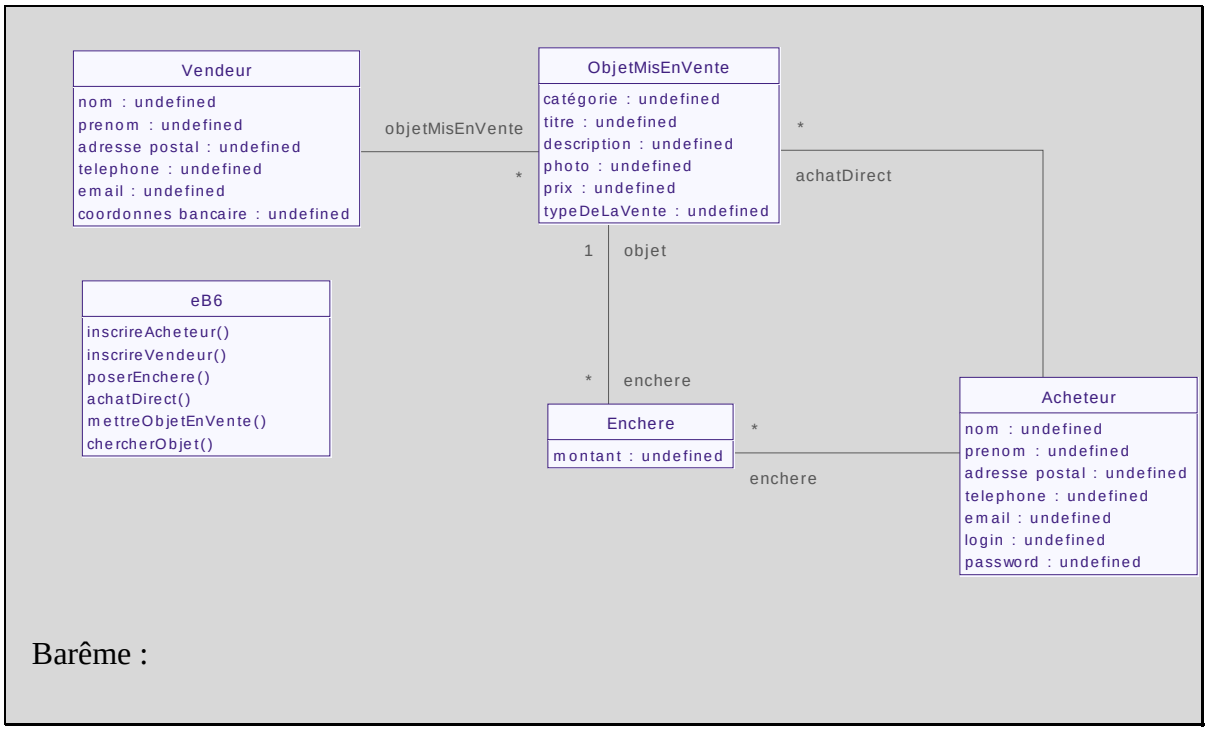
Fiche :

Nom du cas d'utilisation	Mise en vente directe
Acteurs	Vendeur
Description	Mise en vente directe d'un produit
Pré-conditions	Vendeur enregistré et authentifié
Suite des interactions	• Remplissage du formulaire de mise en vente (titre, description, photo...) • Choix vente directe
Post conditions	L'objet apparaît dans la liste des objets en vente sous les catégories et mots-clés choisis par le vendeur
Hypothèses	-
Exigences non fonctionnelles	-
Exceptions / alternatives	-

Barême :

-10% par erreur (pré-condition oublié, étape manquante dans le scénario, …)

Question 2.2 : Réalisez le diagramme de classes de la phase d'analyse. Vous justifierez tous vos choix.



Barême :

Il faut absolument voir apparaître une classe pour l'acheteur et une classe pour le vendeur. La classe utilisateur ne sert à rien (il ne faut donc pas la mettre). (-20 % par omission)

Une classe représentant une mise en vente doit apparaître (-30% sinon)

Cette classe doit posséder un type (vente directe ou enchère) (le type peut être réifié) (-10% sinon)

Cette classe doit posséder tous les attributs du formulaire (-10% s'il en manque 1)

Les vendeurs doivent être associés avec la mise en vente (pour 1 vendeur * mises en vente) (-10% si l'association n'est pas précisée)

Une association ou une classe représentant l'achat direct doit apparaître (-20% sinon)

Une classe représentant une enchère doit apparaître (-30% sinon)

La classe doit être associée à la mise en vente

La classe doit être associée à l'acheteur

La classe eB6 doit apparaître (-20% sinon)

Avec les méthodes correspondant grosso-modo aux cas d'utilisation (-10% si il manque des méthodes)

Le diagramme doit être commenté (-20% sinon)

Si les associations sont navigables (-5%)

Si la notation UML est douteuse (-20 %)

Question 2.3 : Réalisez un diagramme de séquence présentant une mise en vente par enchère d'un objet par un vendeur ainsi que l'achat par deux acheteurs.

Il faut voir apparaître les trois instances des acteurs (2 acheteurs et un vendeur) (-10% par instance manquante)

Il faut voir apparaître l'instance eB6 (-20% sinon)

Il faut voir apparaître la création de la vente (-20 % sinon)

Il faut voir chaque acheteur poser son enchère (-20% sinon)

Il faut voir le système (eb6) réaliser lui-même la résolution des enchères (-30% sinon)

Notation UML (instance, message avec paramètres, …) (-20%)

Question 2.4 : Réalisez 4 tests de validation couvrant la séquence que vous avez réalisée en question 2.3. Pour chaque test vous préciserez le cas d'utilisation correspondant.

Titre : Mise en vente enchere	**id 1**
Contexte :	vendeur enregistré et authentifié
Entrée :	Annonce : titre : VTT TBE description : VTT 21 vitesses, fourche avant téléscopique, VBrake, bleu, compteur avec fil, fourche avant téléscopique photo : Il est bo le vélo prix : 50€ durée : 10j catégories : vélo, sport, loisir mots clé : VTT
Scénario :	• Choix de l'action mettre en vente enchère • entrée des caractéristiques de l'objet (cf entrée) •
Résultat attendu :	Objet ajouté à la liste
Moyens de vérification :	affichage de la vente lors de l'affichage des ventes par catégorie (maison) ou par mot-clés (décoration)

Titre : Recherche d'un objet par catégorie	**id 2**
Contexte :	Arrivée sur le site et recherche d'un objet à partir d'une catégorie. Il y a déjà des objets en vente (id 1)
Entrée :	Catégorie : vélo
Scénario :	L'utilisateur parcours les objets dont la catégorie est « vélo » affichage des résultats
Résultat attendu :	affichage l'objet précédemment enregistré doit apparaître dans la liste
Moyens de vérification :	Visuelle

Titre : Proposer une enchère	**id 3**
Contexte :	Objet sélectionné (id 2), ajout d'une proposition de vente
Entrée :	Prix = 70
Scénario :	• L'acheteur s'authentifie • Selectionne le VTT (test id2) • Propose une enchere dont le montant max est 70^{e}€
Résultat attendu :	Affichage que l'enchère est posée
Moyens de vérification :	Visuel

Titre : Valider une vente	**id 4**
Contexte :	vente en cours (id 1), enchères proposées (id 3) et arrivée à la date butoir
Entrée :	confirmation du vendeur
Scénario :	• Le vendeur confirme la vente sur le site • Les informations du vendeur et de l'acheteur sont envoyés par mail
Résultat attendu :	suppression de l'objet de la liste
Moyens de vérification :	recherche de l'objet (test id 2) ne doit plus afficher cet objet.

Il y a bien sur d'autres possibilité

Barême :

-75% si pas de paramètre précis

-50% si pas de chainage

-10% par erreur/inprécision

-30% par incohérence

M1 : Ingénierie du Logiciel

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Exam de janvier 2006

L'équipe enseignante du module

2 heures avec documents

0. Préambule : A lire avant de commencer le partiel

Attention ! Toute réponse à une question de l'épreuve sans justification ou rédigée en ne respectant pas la syntaxe d'UML sera notée pour 0 point.

Bonne chance !

1. Questions de cours [5 Pts]

Répondre à chaque question de manière précise et concise. Votre réponse ne doit pas dépasser une dizaine de lignes. Chacune des questions a le même poids.

Q1.1

L'emploi de la notation UML en conception impose-t-elle d'utiliser une plate-forme à objets en réalisation ?

Q1.2

Qu'apporte l'itération dans les cycles de développement de génie logiciel ? Quels sont les projets sur lesquels l'itération est plus intéressante voire nécessaire ?

Q1.3

L'eXtreme Programming préconise l'écriture des tests avant la rédaction du code, quel est l'intérêt et quelle est la difficulté de cet exercice ?

Q1.4

A quoi servent les interfaces offertes des composants de conception ? Doit-on obligatoirement les retrouver en réalisation ?

Q1.5

A quoi servent les diagrammes d'états lors de la rédaction des tests unitaires ? Est-ce qu'un test unitaire peut porter sur deux classes ?

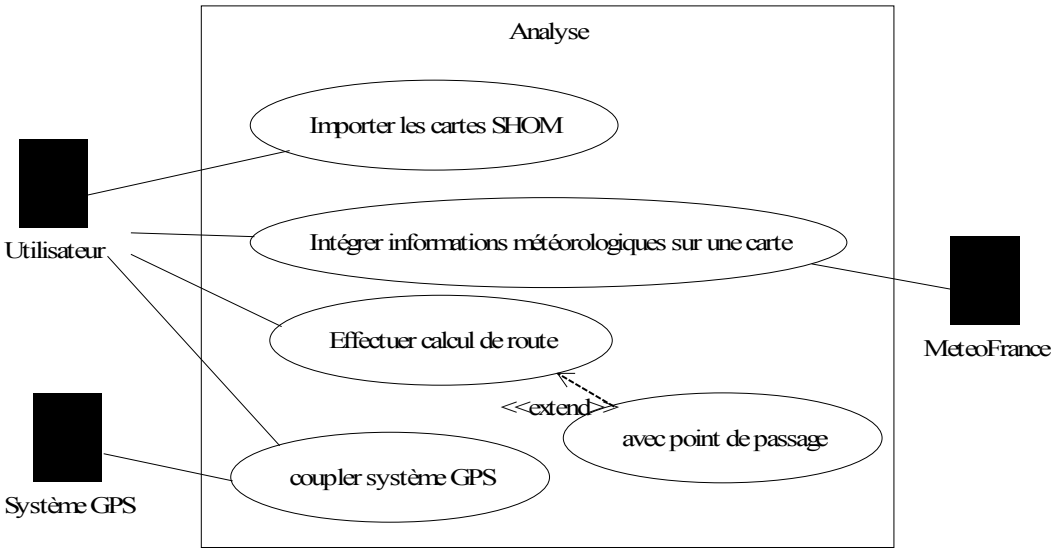
2. Problème: Conception de SpeedBoat [15 Pts]

Après leur partiel de IL, des étudiants de master ont décidé de réaliser la phase de conception du logiciel SpeedBoat.

Ils ont décidé de ne pas refaire une analyse et de ne se baser que sur les diagrammes suivants

Analyse

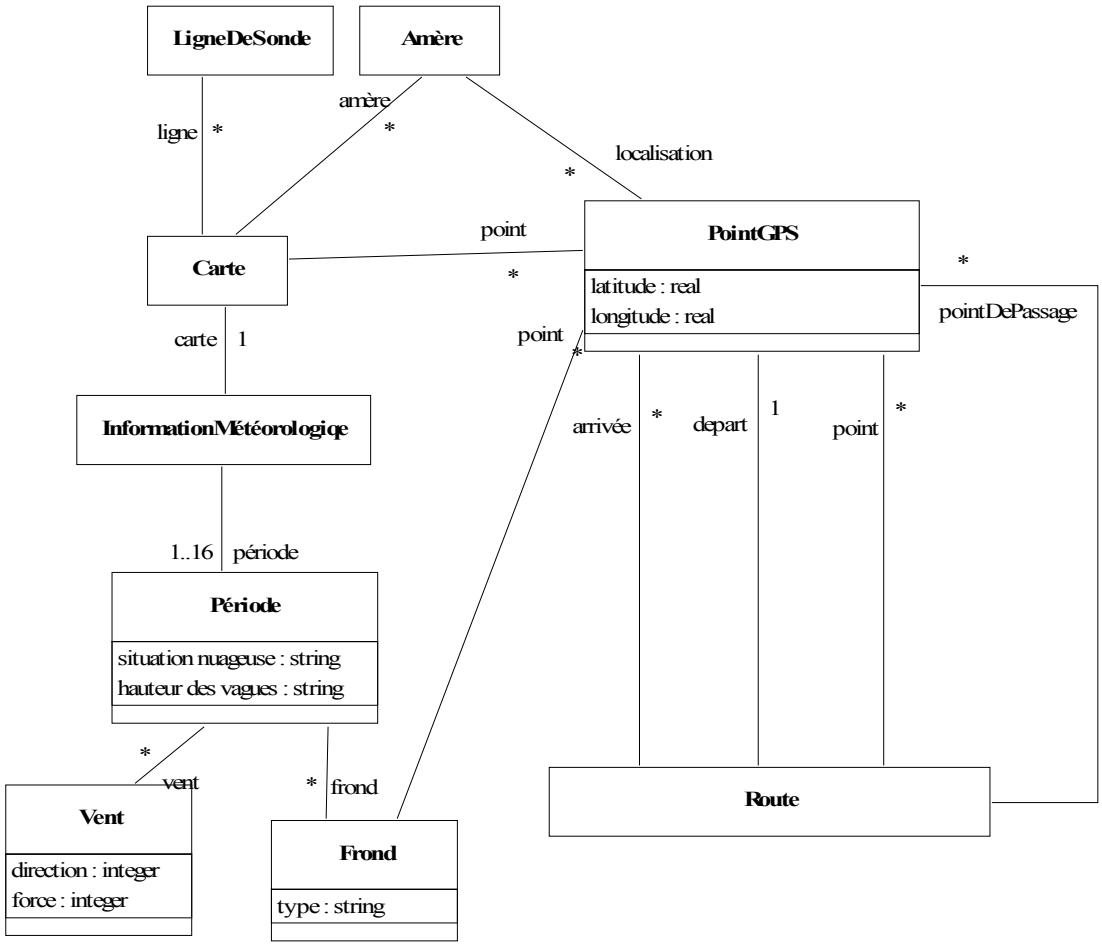
Diagramme de Cas d'utilisation :



Description des cas d'utilisation :

1. Importer les cartes SHOM : permet d'importer dans le logiciel les cartes maritimes du SHOM.
2. Intégrer information météorologiques sur une carte : permet d'obtenir les informations météorologiques contenues sur le site de MétéoFrance et de les intégrer dans le logiciel.
3. Effectuer calcul de route : permet d'effectuer un calcul de route entre un point de départ et un point d'arrivée.
4. Avec points de passage : ce cas est une extension du calcul de route et permet d'insérer des points de passage dans la route.
5. Coupler avec un système GPS : permet de coupler le logiciel avec un système GPS afin que le logiciel puisse récupérer les coordonnées de la position courante du bateau.

Diagramme de classes :



Conception : Découpe en composants

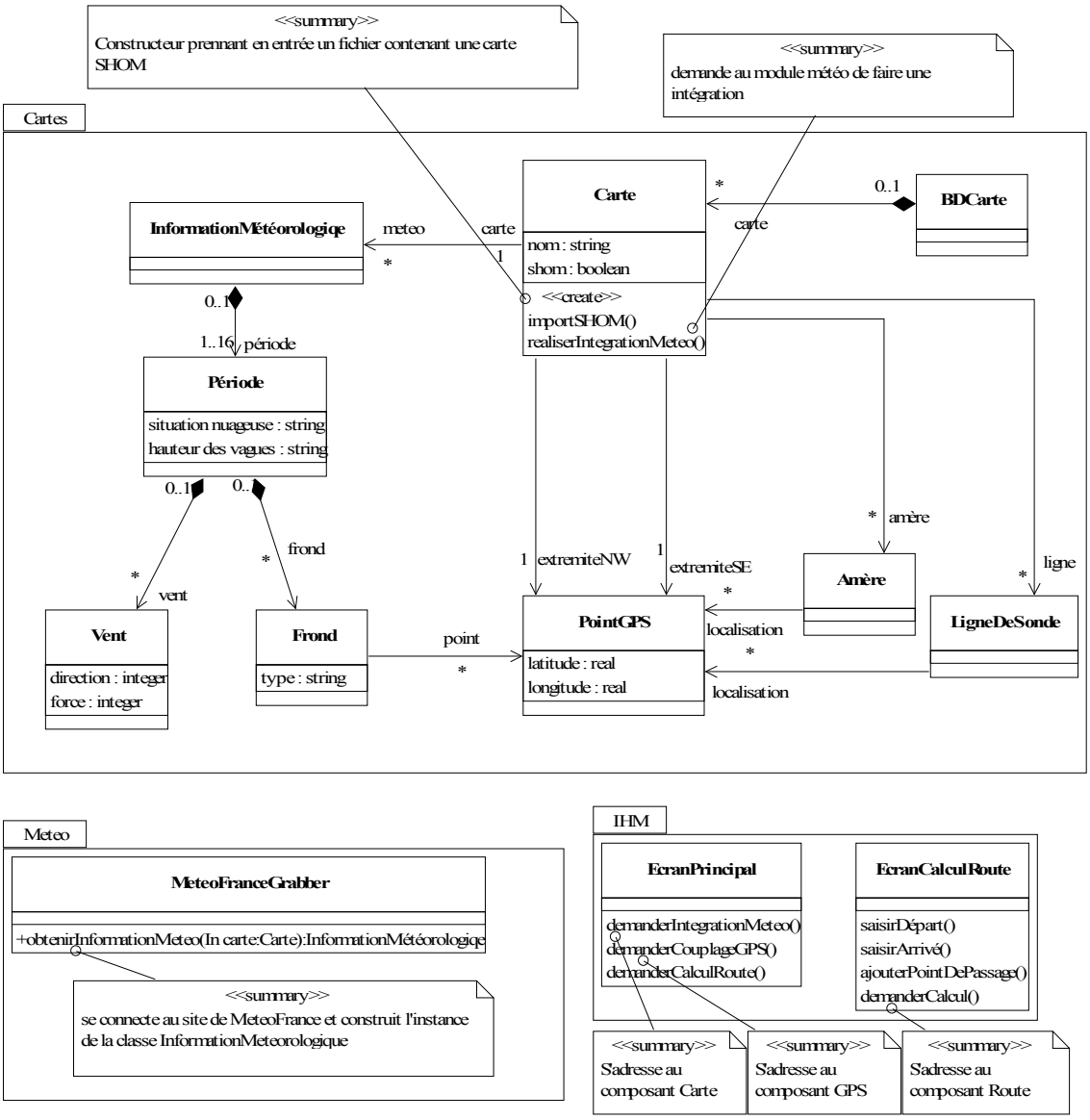
Sur la base de cette analyse, les étudiants proposent la découpe suivante en composants :

- **Carte** : Un composant permettant de gérer les cartes dans SpeedBoat. Ce composant permet à l'utilisateur de créer de nouvelles cartes, d'en supprimer et d'en importer à partir du format SHOM.
- **Meteo** : Un composant gérant l'interaction avec le site web de Météo France et gérant l'intégration des informations météo dans les cartes.
- **Route** : Un composant effectuant le calcul de la route (avec prise en compte des points de passage).
- **GPS** : Un composant réalisant le couplage avec le système GPS.
- **IHM** : Un composant représentant l'interface homme / machine du logiciel.

Question 2.1 : Présentez, à l'aide d'un diagramme de package, les dépendances que vous auriez établies entre les composants. Présentez, à l'aide d'un diagramme de cas d'utilisation par composant, votre interprétation des fonctionnalités de chaque composant. Que pensez-vous de cette décomposition ?

Conception : Structuration interne des composants

La figure suivante présente une partie de la conception réalisée par les étudiants.



Cette conception devrait être différente de votre interprétation (celle que vous avez présentée dans la question 2.1). En effet, cette conception souffre d'un défaut majeur et devrait d'ailleurs être interdite car elle ne respecte pas les principes de base de conception !

Question 2.2 (4 Pt) : Identifiez et expliquez clairement le défaut majeur de cette conception. Corrigez ce défaut et expliquez votre correction.

Conception Abstraite : Composant Météo

Cela n'est pas un défaut majeur mais le composant Météo n'est pas très bien conçu. Il ne contient qu'une seule classe possédant une seule opération réalisant l'intégralité du traitement (recherche de l'information dans le site, décodage de la page HTML et construction de l'instance de la classe InformationMétéorologique).

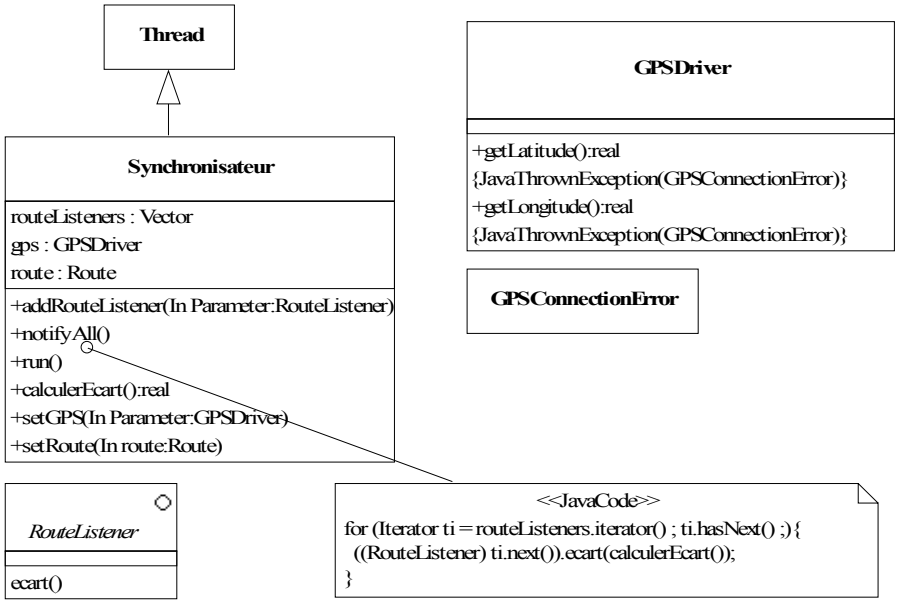
De ce fait, ce composant souffre des inconvénients suivants :

- Il n'est pas possible de récupérer les informations météo d'autres sites.
- Il faut changer la classe si le site de MétéoFrance change.
- Il faut changer la classe si l'on veut intégrer une autre source d'information météo (type web services ou SMS.

Question 2.3 (4 Pts) : Proposez une nouvelle conception permettant de résoudre ces inconvénients. Exprimez l'interface offerte du composant Météo et précisez quels sont les composants qui utilisent cette interface.

Réalisation : Synchronisation avec le GPS

Pour la synchronisation avec le GPS, les étudiants ont décidé de la réalisation ainsi :



Le synchronisateur (classe Synchonisateur) s'exécute tout le temps (hérite de la classe Thread de Java). Il calcule indéfiniment l'écart entre la position du bateau (via GPSDriver) et la route à suivre. L'écart est alors renvoyé à n'importe quel objet dont la classe réalise l'interface RouteListener.

Avant de démarrer le synchronisateur, il faut s'assurer qu'il possède bien une référence vers un driver de GPS. Il faut aussi s'assurer qu'il possède bien la route suivie !

Si lors de son exécution, le synchronisateur récupère 3 fois de suite des exceptions de la part du driver de GPS, il s'arrête et attend qu'on lui réaffecte un nouveau driver. Si aucun nouveau driver ne lui est affecté (on considérera qu'on lui donne « null » comme référence), le synchronisateur envoie un message à tous les objets qui l'écoutent indiquant que la synchronisation ne s'effectue plus.

Question 2.4 (4 Pt) : Réalisez le diagramme d'états du synchronisateur. Présenter deux tests unitaires en précisant vos objectifs de test.

M1 : Ingénierie du Logiciel

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Exam de septembre 2006

L'équipe enseignante du module

2 heures avec documents

0. Préambule : A lire avant de commencer le partiel

Attention ! Toute réponse à une question de l'épreuve sans justification ou rédigée en ne respectant pas la syntaxe d'UML sera notée pour 0 point.

Bonne chance !

1. Questions de cours [5 Pts]

Répondre à chaque question de manière précise et concise. Votre réponse ne doit pas dépasser une dizaine de lignes. Chacune des questions a le même poids.

Q1.1

Pourquoi les tests de validation ne sont-ils pas réalisés en UML ?

Q1.2

UML est-il indispensable au suivi d'un cycle de développement de génie logiciel ?

Q1.3

Quelles sont les différences entre les classes d'analyse et les classes de conception ?

Q1.4

Peut-on générer complètement le code d'une application à l'aide de diagrammes de classes et de séquence?

Q1.5

A quoi sert UML pour la réalisation des tests unitaires ?

2. Problème: Conception de ConfMaster [15 Pts]

L'objectif de tout chercheur est de publier ses résultats de recherche dans des conférences scientifiques. Pour ce faire, les chercheurs doivent soumettre des articles de recherche aux conférences. Les comités de programme des conférences scientifiques ont alors pour objectif de sélectionner (ou non) les articles soumis en fonction de l'importance des résultats obtenus.

Les informations suivantes permettent de bien comprendre la façon dont les conférences gèrent la sélection des articles :

- Le comité directeur de la conférence est constitué de chercheurs de grande renommée.
- Les membres du comité de programme sont des chercheurs désignés par le comité directeur de la conférence. Plus précisément, on distingue dans le comté de programme un président parmi tous les membres.
- Dans la première phase de la préparation de la conférence, le président du comité de programme publie un appel à communications comportant une liste de thèmes, une date de soumission, et une date d'avis (acceptation ou refus aux auteurs).
- Les articles sont envoyés au président du comité de programme. Chaque article a un titre et peut concerner un ou plusieurs thèmes de la conférence. Un article peut avoir un ou plusieurs auteurs (des chercheurs appartenant à un laboratoire ou des industriels). L'un de ces auteurs est l'auteur principal, tenu responsable de la communication avec le comité de programme.
- Tout article envoyé avant la date de soumission fixée est dit « soumis ». Les articles arrivant après la date de soumission sont dits « dépassés » et sont retournées à leur auteur principal.
- A l'issue de la période de soumission, chaque article soumis reçoit un numéro et est envoyé à trois membres du comité de programme choisis par le président.
- Chaque membre du comité de programme est responsable d'un nombre d'articles qu'il a le droit de distribuer à son tour à d'autres chercheurs (pour leur compétence relative au domaine) qui sont dit lecteurs du papier.
- Le président fixe une date de retour des évaluations qui marque la fin de la période d'évaluation et une date de réunion du comité de programme dont le but est de clore définitivement la sélection.
- Chaque article est retourné au président en fin d'évaluation avec une appréciation générale (une note entre 0 et 10 par exemple) et des commentaires explicatifs pour le comité de programme et pour les auteurs du papier.
- Lors de sa réunion, le comité décide d'abord pour chaque papier (en fonction de ses trous évaluations) si il est retenu ou non. Cette réunion est terminée par le découpage de la conférence en séances, en prenant soin de regrouper les papiers retenus concernant les mêmes thèmes ou des thèmes très voisions. Une séance a un nom, une durée, un responsable (un membre du comité de programme) et un ensemble de papiers.
- Lorsque la sélection est close, le président communique à l'auteur principal le résultat de l'évaluation de tout papier soumis.

On souhaite réaliser une application, nommée ConfMaster, de gestion de conférences scientifique. Cette application doit faciliter toutes les interactions et toutes les communications entre tous les intervenants de la conférence.

Question 2.1 (4 Pt) : Réalisez le diagramme de cas d'utilisation de l'application et présentez-le?

Question 2.2 (4 Pt) : Un des cas d'utilisation de réponse à la question 2.1 doit représenter la soumission d'un article. Illustrez cette soumission selon deux scénarios à l'aide d'un diagramme de séquence et fournissez les deux tests de validation correspondant.

Question 2.3 (4 Pts) : Proposez une conception permettant de résoudre la fonctionnalité de .

Question 2.4 (4 Pt) : Votre conception correspondant à la question 2.3 doit contenir une classe représentant le concept de « soumission d'article ». Elaborez le diagramme d'état de cette classe. Définissez, à l'aide de diagrammes de séquence de test, deux tests unitaires de cette classe.

M1 : Ingénierie du Logiciel

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Examen de décembre 2006 (2 heures avec documents)

1. Questions de cours [5 Pts]

Q1.1 : Quelles sont les différences entre les dépendances structurelles et les dépendances fonctionnelles ?

Les dépendances fonctionnelles apparaissent entre composants, lorsqu'un composant fait appel à une opération offerte par un autre composant. Il dépend alors des fonctions fournies par l'autre composant. Il n'a pas besoin par contre de connaître la structure interne de ce composant.

Les dépendances structurelles apparaissent lorsqu'un lorsqu'une classe d'un composant dépend de la structure d'une classe (héritage, association, typage des attributs ou des paramètres) d'un autre composant. Le composant a donc besoin de connaître la structure interne de l'autre composant (c'est interdit en IL).

Barème :

+40% par explication

+20% si précision que les dépendances fonctionnelles sont autorisées entre les composants et pas les dépendances structurelles.

0% sinon

Q1.2 : Peut-on dire qu'une bonne conception est une conception qui défini une solution répondant aux besoins spécifiés lors de l'analyse (ni plus, ni moins) ?

Non, il faut aussi qu'elle soit flexible, performante, robuste, efficace … La complétude et la justesse ne sont pas les seuls critères.

Barème :

100% si tout

0% sinon

0% si conception = uniquement la solution d'un problème (cela n'est alors pas une BONNE conception)

Q1.3 : En conception, est-il possible de résoudre plus de besoins que ceux spécifiés en analyse ?

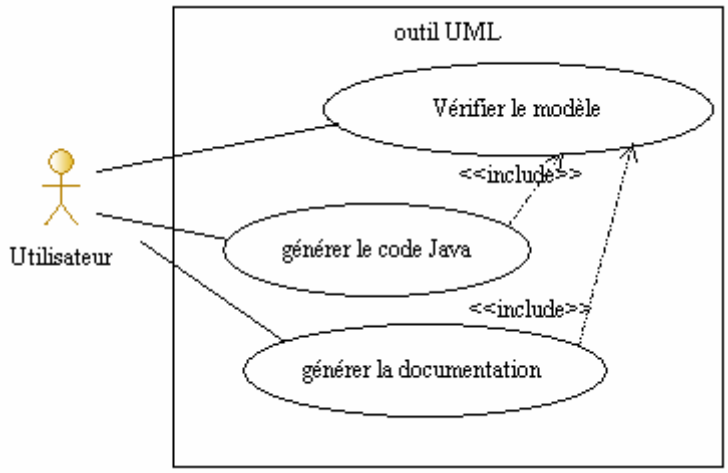
Oui, il est par exemple possible de réutiliser un composant qui fait bien plus que nécessaire. Dans un des TD par exemple, un des composants utilisait une mémoire cache afin d'aller plus vite. La mémoire cache n'est pas nécessaire et pourtant elle améliore la conception.

Barème :

100% si tout

0% si autre explication peu convaincante.

Q1.4 : Le diagramme de cas d'utilisation suivant représente certaines fonctionnalités d'un outil UML. En considérant que la vérification du modèle vérifie que le modèle est bien un modèle UML, discutez des relations d'include.



Ces relations d'inclusion sont tout à fait possible car il est fort à parier qu'une vérification soit faite avant toute génération de code et de documentation. De plus, il est tout à fait envisageable de faire en sorte que l'utilisateur bénéficie de cette fonctionnalité.

Barème :

100 % si tout

50 % si explication correcte sur la non nécessité des includes (ceux-ci ne sont en effet pas obligatoire, on peut s'en sortir avec des pré-condition).

50 % si explication de ce que veut dire le diagramme sans réelle discussion.

0 % si explication douteuse sur les include (genre, non c'est interdit parce que on a dit en cours que c'était interdit).

Q1.5 : Quel est l'intérêt de la phase de conception par rapport à la phase de réalisation ? En d'autres mots, est-il possible de passer directement de l'analyse à la réalisation ?

La phase de conception permet d'être indépendant d'une plate-forme. Si l'indépendance n'est pas un objectif, on peut s'en passer.

Barème :

100 % si tout.

50% si justification assez précise sur l'intérêt d'une découpe en composant (dépendance entre composant et vue abstraite)

0 % sinon.

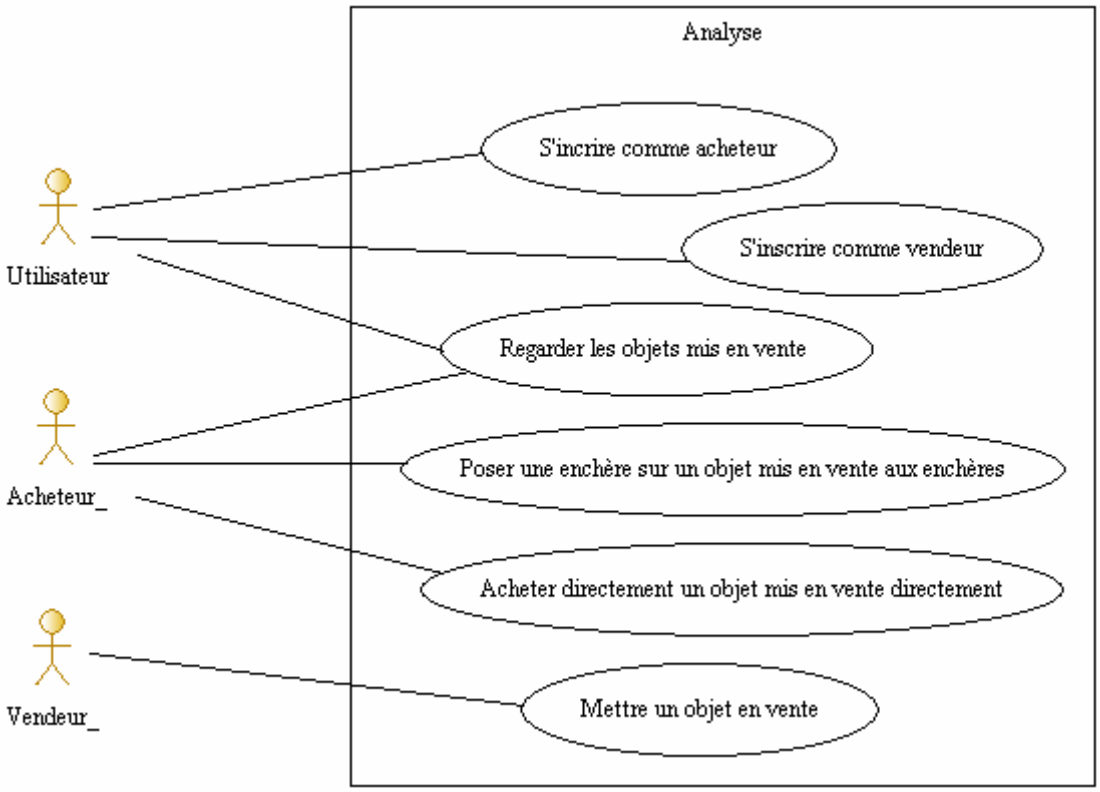
2. Problème: Conception et Réalisation de eB6 [15 Pts]

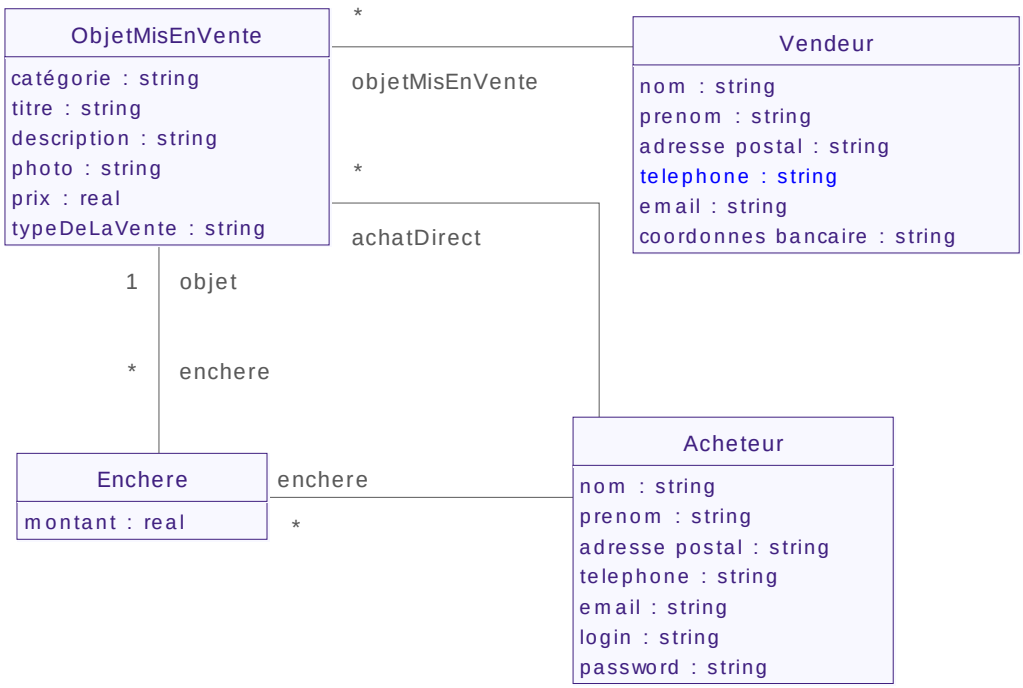
Après leur partiel de IL, des étudiants de master ont décidé de réaliser la phase de conception du logiciel eB6. Ils ont décidé de ne pas refaire une analyse et de ne se baser que sur les diagrammes suivants.

Description des cas d'utilisation :

1. S'inscrire comme acheteur : un utilisateur peut s'inscrire comme acheteur.
2. S'inscrire comme vendeur : un utilisateur peut s'inscrire comme vendeur.
3. Regarder les objets mis en vente : un utilisateur ou un acheteur peuvent regarder tous les objets mis en vente.
4. Poser une enchère sur un objet mis en vente aux enchères : un acheteur peut poser une enchère sur un objet qui a déjà été mis en vente et qui n'est pas encore vendu.
5. Acheter directement un objet mis en vente directement : un acheteur peut acheter directement un objet si celui-ci a été mis en vente directement. Le premier acheteur a acheter directement l'objet peut alors traiter avec le vendeur.
6. Mettre un objet en vente : un vendeur peut mettre un objet en vente soit aux enchères soit en vente directement.

Diagramme de cas d'utilisation d'analyse et diagrammes de classes d'analyse (ne représentant que les classes métiers):

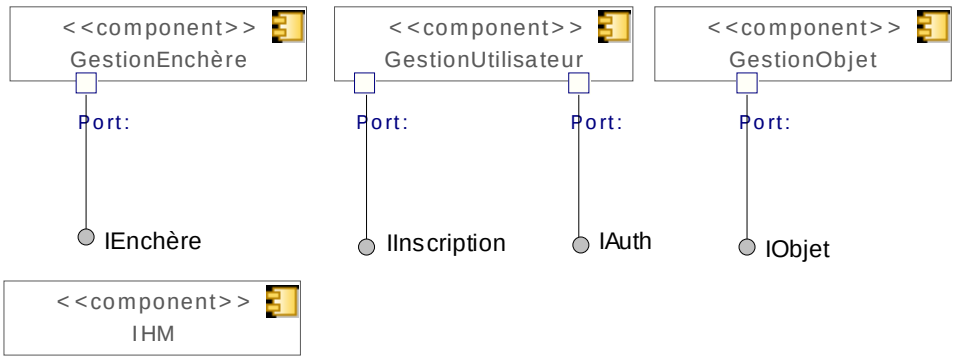




Les étudiants proposent la découpe en composants suivantes :

- Un composant GestionUtilisateur responsable des inscriptions des utilisateurs (en tant qu'acheteur et vendeur) et responsable de l'authentification des utilisateurs.
- Un composant GestionObjet responsable de la gestion des objets mis en vente par les vendeurs. Ce composant est aussi responsable de la recherche des objets en fonction catégories ou de mots clé.
- Un composant GestionEnchère responsable de la gestion des enchères. Ce composant enregistre les enchères et sélectionne la meilleure enchère lors de l'échéance de la vente.
- Un composant IHM qui représente l'interface homme machine de l'application.

Le diagramme suivant ne représente que les interfaces offertes des composants.



Question 2.1 : Spécifiez les opérations des interfaces IObjet et IEnchère.

Dans IObject, il faut voir apparaître une opération pour ajouter un objet et deux opérations pour rechercher les objets (par mots clés et par catégories). Il faut voir les paramètres de ces opérations (pas de dépendance structurelle !)

Dans IEnchère, il faut une opération permettant l'ajout d'une nouvelle enchère et une opération permettant d'obtenir la meilleure enchère pour un objet mis en vente. Il faut voir les paramètres de ces opérations (pas de dépendance structurelle !).

Barème :

50% par interface

25% pour les opérations

25% pour les paramètres sans dépendance structurelle

Question 2.2 : Réalisez un diagramme de séquence de conception pour la mise en vente par enchère d'un objet par un vendeur ainsi que l'achat par deux acheteurs. Il est demandé de ne montrer que les interactions entre composants.

Il faut voir apparaître au moins une instance pour les composants (GestionEnchère,GestionObjet et IHM). 10%

Si l'authentification est traitée il faut voir apparaître une instance de GestionUtilisateur. 10%

Il faut voir apparaître 3 acteurs (un vendeur et deux acheteurs) 10%

Les acteurs ne doivent communiquer qu'avec l'instance de IHM 10%.

Il faut voir apparaître l'ajout d'une nouvelle enchère : communication du vendeur vers l'instance IHM puis communication de l'instance IHM vers l'instance de GestionObject 20%

Il faut voir apparaître les 2 dépôts d'une enchère : communication du vendeur vers l'instance IHM (pour récupérer l'identification de l'objet puis pour déposer une enchère) communication de l'instance IHM vers l'instance GestionEnchère (pour le dépôt effectif de l'enchère) 20%

Les paramètres doivent être précis 20%

Si l'authentification est pleinement modélisée 20%

Toute faute (responsabilité, paramètre, nom d'instance, oubli de message, message superflu, …) -10%

Question 2.3 : Définissez les interfaces requises des composants et précisez leurs dépendances (justifier vos choix).

Idéalement il faut que IHM dépende des autres composants et c'est tout.

Néanmoins, il faut aussi et surtout que les dépendances soit cohérentes avec le diagramme de séquence de la question précédente.

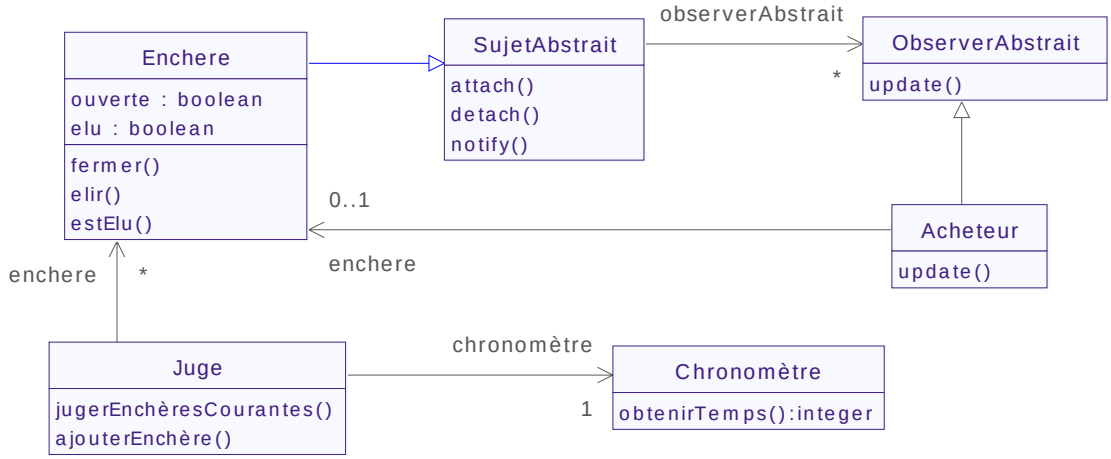
Barème :

100% si que des dépendances de IHM vers les autres et cohérence avec le diagramme de séquence

50% si les dépendances sont cohérentes avec le diagramme de séquence (sans réelle reflexion)

0% sinon, ou si rajout de dépendance structurelle.

Question 2.4 : On considère que le composant GestionEnchère contient une classe représentant les enchères (Enchere) et que le composant GestionUtilisateur contient une classe représentant les acheteurs (Acheteur). Pour gérer la résolution des enchères, c'est-à-dire l'élection de l'enchère la plus haute, les étudiants proposent la conception suivante :



La classe Juge est la classe qui contient l'algorithme d'élection des enchères (dans l'opération jugerEnchèresCourantes). Cette classe référence un unique chronomètre qui lui permet de connaître le temps écoulé et ainsi d'élire les enchères dont la date limite de la vente est arrivée à échéance. Une enchère est ouverte tant que la date limite de la vente n'est pas arrivée à échéance, après cela le juge ferme l'enchère (grâce à l'opération fermer). Ensuite, le juge élit uniquement l'enchère la plus haute (grâce à l'opération elir).

Afin de notifier l'acheteur que son enchère a été élue, les étudiants proposent d'appliquer le pattern Observer. L'enchère est ainsi considérée comme un sujet alors que l'acheteur est considéré comme un observeur.

Expliquez pourquoi cette conception ne suit pas les principes de conception énoncés en cours puis corrigez-la.

L'utilisation de ce pattern est complètement fausse. En effet, il y a ici une confusion entre la classe Acheteur l'acteur Acheteur. Le réel besoin est de notifier l'acteur Acheteur (et non pas la classe Acheteur).

De plus, ce pattern met en place des dépendances structurelles entre les classes des différents composants (Acheteur et Enchere) !

La meilleure chose à faire est de faire supprimer ce pattern (il ne sert à rien).

Une autre possibilité est de modifier ce pattern pour faire disparaître les dépendances structurelles (il est possible d'utiliser des identifiants pour lier les sujets des observateurs (cela laisse néanmoins une double dépendance fonctionnelle entre les composants).

Barème :

100 % si le pattern est supprimé

50 % si les dépendances structurelles sont modifiées en dépendances fonctionnelles.

50 % si identification du problème comme étant une mauvaise application du pattern observer (sans résolution)

25 % si identification du problème comme étant une dépendance structurelle (sans résolution)

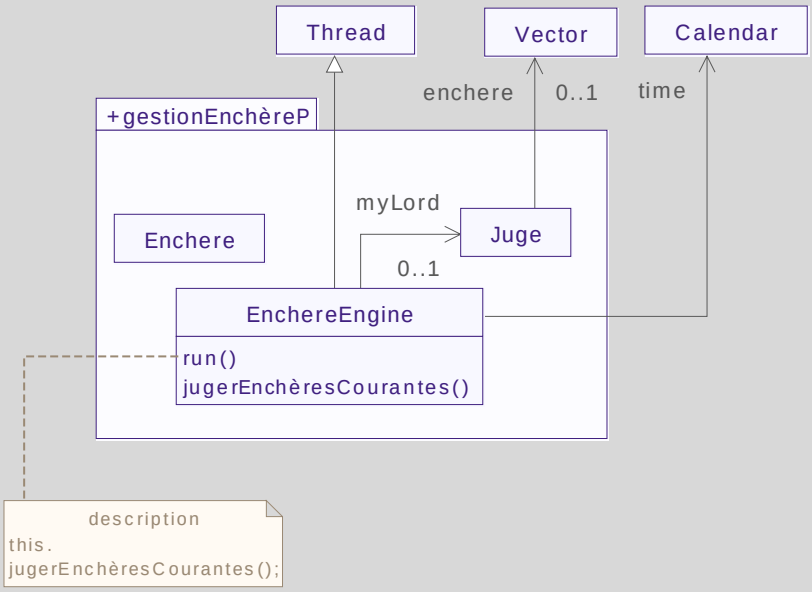
Question 2.5 : Les étudiants souhaitent maintenant effectuer la réalisation de la classe Juge sur la plate-forme Java (vous pouvez faire cette question que vous ayez corrigé ou pas la conception lors de la question 2.4). Etant donné que l'élection des enchères doit s'effectuer tout le temps, les étudiants souhaitent utiliser la classe Java Thread. La classe Thread permet de construire un nouveau processus et de lui affecter un traitement (dans l'opération run() de la classe Thread). Les étudiants souhaitent que le juge construise un nouveau Thread (dès l'instanciation du juge) et que ce Thread réalise en continu l'opération jugerEnchèresCourantes(). De plus, les étudiants souhaitent utiliser la classe Java Calendar qui permet d'obtenir, grâce à l'opération int getTime(), la date précise du moment (en milliseconde). Proposez un diagramme de classe illustrant la réalisation de la classe Juge (en ajoutant le code Java des méthodes que vous jugerez utiles de préciser).

Juge doit être associé avec une classe (EnchereEngine) qui hérite de Thread (30%)

La méthode run de cette classe doit faire appel à l'opération jugerEnchèresCourantes() 20%

Cette classe doit être reliée à la classe Calendar (c'est elle qui a besoin du temps pour savoir quelles enchères calculer). 20%

Cette classe doit pouvoir atteindre les enchères gérée par le Juge (association myLord) (30%)



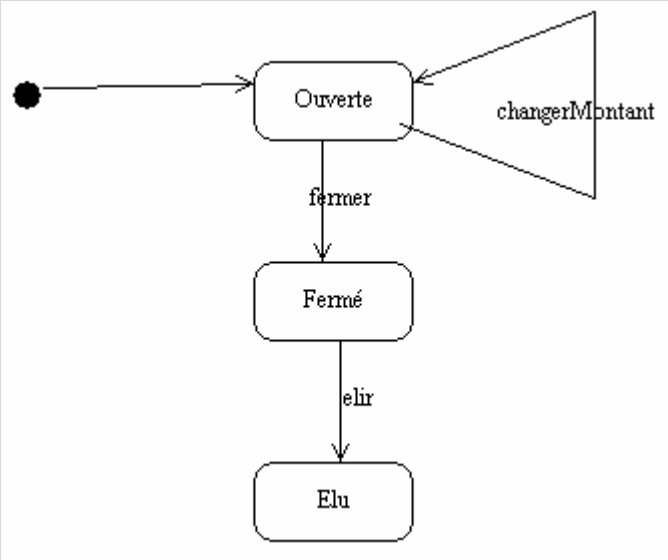
Il y a bien sûr d'autres conceptions qui tiennent la route.

Par contre, une conception qui ne fait que lier Juge à Thread et Calendar mérite 0% (il ne suffit pas de recopier l'énoncé).

Question 2.6 : Etant donné qu'une enchère passe par plusieurs états, définissez la machine à état de la classe Enchère puis identifiez une faute et réalisez un diagramme de séquence de test visant à révéler cette faute.

Les états d'une enchères sont : ouverte (avec un montant), fermé, élu

La machine à état devrait être celle-ci :



A partir de là, il n'est pas très difficile d'identifier des fautes. Par exemple, pouvoir pouvoir élir une enchère alors qu'elle n'est pas fermée.

Barème :

40% pour la machine à état

40% pour la faute

20 % pour le cas de test sous forme d'un diagramme de séquence.

M1 : Ingénierie du Logiciel

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Partiel de octobre 2007 (2 heures avec documents)

1. Questions de cours [5 Pts]

Q1.1 : Quelles sont les différences entre la phase de définition des besoins et la phase de spécification des besoins ? Laquelle de ces deux phases avons-nous appelée phase d'analyse ?

La phase de définition des besoins correspond à la rédaction du cahier des charges. C'est pendant cette phase que les besoins seront identifiés et définis. Dans la phase de spécification des besoins, l'équipe de développement spécifie rigoureusement les besoins. En cours, seule la phase de spécification des besoins est appelée phase d'analyse.

Barème :

100% si tout

0% sinon

Q1.2 : Dans un diagramme de cas d'utilisation, les acteurs liés aux cas d'utilisation sont-ils forcément des bénéficiaires (client des fonctionnalités) ?

Non pas forcément. D'après la définition du standard, les acteurs interagissent avec le système lors de la réalisation de la fonctionnalité.

Barème :

100% si tout

0% sinon

Q1.3 : Pourquoi conseille-t-on de ne pas rendre les associations navigables entre les classes d'analyse ?

Car les navigations servent essentiellement lors du développement des comportements des opérations. Elles induisent des dépendances entre classes. En analyse, spécifier ces dépendances n'est premièrement pas nécessaire, deuxièmement contradictoire avec l'objectif de la phase qui consiste uniquement à spécifier les besoins définis dans le cahier des charges.

Barème :

+50% si blabla sur les dépendances nécessaire uniquement en conception

+50% si blabla sur le fait que cela n'est pas rédiger dans le cahier des charges et donc pas un objectif de la phase.

Q1.4 : Pourquoi faut-il rédiger les tests de validation avant de commencer la conception ?

Car les tests de validation doivent être certifiés par le client qui assure que les besoins ont été bien compris (spécifiés) par l'équipe de développement.

Barème :

100% si tout

0% sinon

Q1.5 : UML est-il absolument nécessaire à la réalisation de la phase d'analyse ?

Non, UML n'est qu'un moyen pour réaliser cette phase.

Barème :

100% si tout

0% sinon.

2. Problème: Analyse de iklinéa [15 Pts]

Une société vendant des meubles désire mettre en place un système de suivi de clientèle, nommé « iklinéa ». Ce système permettra de suivre les achats, livraisons et retours de ses clients. Vous avez la charge de réaliser la phase d'analyse de ce système. Le cahier des charges est le suivant :

N'importe quel vendeur de la société peut ajouter ou mettre à jours une fiche client dans « iklinéa ». Chaque fiche, identifiée par un identifiant unique, contient les informations suivantes :

- Nom et prénom du client ;
- Adresse postale ;
- Adresse électronique ;
- Numéro(s) de téléphone ;

Un client qui souhaite acheter un ou plusieurs meubles doit s'adresser à un vendeur. Le vendeur ajoute dans « iklinéa » la demande d'achat du client. Celle-ci a un identifiant unique, porte la date du jour, référence les meubles à acheter (« iklinéa » contient un unique catalogue des meubles disponibles à la vente), référence la fiche du client, contient le nom du vendeur et une zone de texte permettant au vendeur de laisser un commentaire.

Le client a alors 3 jours pour concrétiser sa demande d'achat. Pour ce faire, il doit s'adresser à un caissier. Les caissiers, après avoir encaissé au moins 10% du prix de la demande d'achat, transforment la demande d'achat en une commande d'achat. La commande d'achat a un identifiant unique, une date de commande, le montant déjà réglé par le client et référence les meubles à acheter ainsi que la fiche du client. Lorsqu'une commande d'achat est enregistrée, la demande correspondante et automatiquement supprimée.

Après avoir effectué sa commande, le client peut alors retirer son produit. Pour ce faire, il doit s'adresser au service de retrait des marchandises. Dans ce service, les magasiniers doivent encaisser la totalité du coût d'achat de la commande avant de remettre au client son

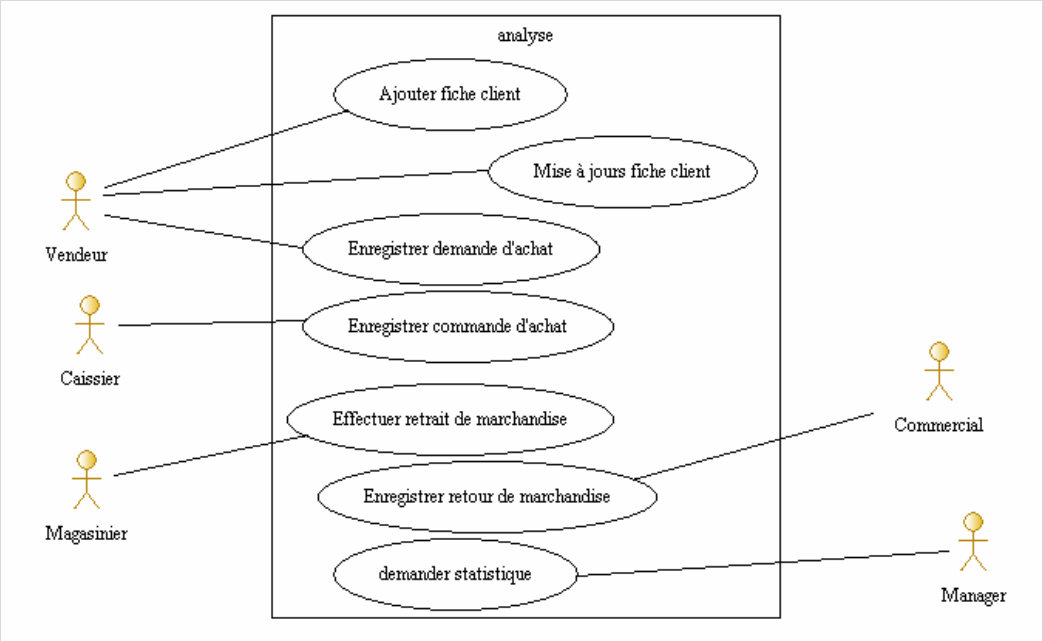
ou ses meubles. Une fois que la remise du ou des meubles a été effectuée, les magasiniers mettent à jours dans « iklinéa » la commande correspondante afin de préciser que le retrait a été fait (la date de retrait doit être enregistrée).

Le client a alors 30 jours après la date de livraison pour retourner le ou les meubles qu'il a achetés et ce quelle que soit la raison. Pour ce faire, il doit s'adresser au service après vente. Là, un commercial, après avoir réceptionné le ou les meubles retournés et après avoir remboursé le client, enregistrera dans « iklinéa » le ou les meubles retournés (par commande, la liste des meubles retournés, la date de retour ainsi que le motif seront enregistrés). Il faut que le retour soit fait avant les 30 jours sinon le retour ne peut pas être enregistré.

Grâce à la saisie de toutes ces informations, les managers peuvent effectuer des études statistiques sur les demandes d'achat, les commandes et les retours. Pour faciliter ces études statistiques, « iklinéa » doit permettre à un manager de connaître l'ensemble des demandes d'achat et l'ensemble des commandes en cours, l'ensemble des demandes d'achat, des commandes, des livraisons et des retours effectués sur une période donnée. De plus, « iklinéa » devra permettre aux managers de retrouver l'ensemble des « super clients » (un super client est un client qui fait plus d'une commande par mois et qui ne retourne jamais ses meubles) et l'ensemble des « clients délicats » (un client délicat est un client qui effectue souvent des demandes d'achat sans faire de commande et qui lorsqu'il fait des commandes retourne très souvent ses meubles). Grâce à ces statistiques facilitées par « iklinéa », les managers pourront alors décider d'effectuer des offres spéciales ou autres initiatives commerciales.

Question 2.1 : Réalisez le diagramme de cas d'utilisation de la phase d'analyse. Vous justifierez tous vos choix.

Dans ma correction, il y a 5 acteurs et chaque acteur n'est lié qu'à un seul cas d'utilisation. Pour autant, il est certainement possible de préciser certains cas d'utilisation (et donc de proposer des cas moins génériques). Par exemple, le cas « enregistrer commande » peut se décomposer en « encaisser totalité » et « transformer demande en commande ».



Barême :

+20% par acteur avec son/ses cas d'utilisation

-30% par acteur supplémentaire (client n'est pas acteur)

-30% par cas d'utilisation en trop (effectuer une offre spéciale n'est pas géré par Iklinéa)

-20% par héritage, include ou extend injustifiable

Question 2.2 : Précisez la feuille détaillée (acteurs concernés, hypothèses, pré et post conditions, scénario) du cas d'utilisation correspondant à un retour de meuble.

Hypothèse : Une commande a été enregistrée et les meubles ont été retirés (je trouve que c'est une hypothèse car sinon, il n'est pas envisageable de faire un retour)

Pré : la commande a été retirée il y a moins de 30 jours (c'est vraiment une pré-condition, sinon, le retour n'est pas possible)

Post : la commande est marquée comme retournée

Scénario : (Toute la question est de savoir si le test des 30j se fait dans le scénario ou pas). Si la pré-condition porte sur les 30j, alors il ne faut pas faire le test dans le scénario.
1) Le commercial enregistre le retour des meubles dans la fiche de la commande

Barême :

Cette question est très délicate à corriger. Il faut donc vérifier les points suivants.

-50% si la post ne fait pas apparaître que le retour des meubles est enregistré dans la fiche de la commande.

-50% si la condition des 30j n'apparaît pas (soit en pré-condition soit en tant que test dans le scénario)

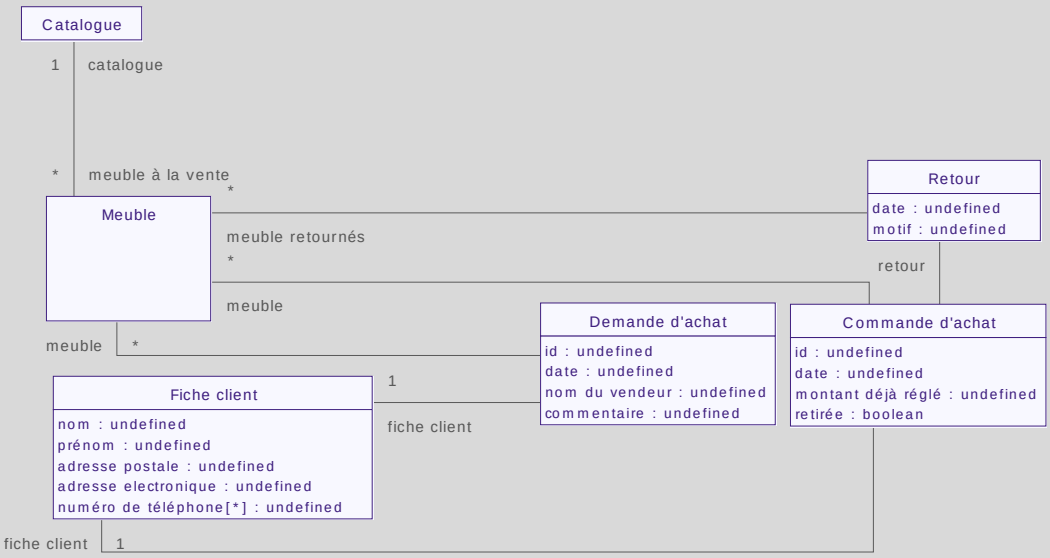
-50% si les pré et post condition ne sont pas cohérente avec le scénario (ex : si les 30j est une pré-condition alors il ne doit pas y avoir de test dans le scénario)

-50% si le scénario fait apparaître des interactions entre des entités externes (ex : si le scénario fait apparaître le remboursement ou le fait que le client s'adresse au commercial)

Pour résumer 100% si la fiche est cohérente et qu'elle fait apparaître la condition des 30j et l'enregistrement du retour.

Question 2.3 : Réalisez le diagramme de classes de la phase d'analyse. Vous justifierez tous vos choix.

Voici ma correction :



Barême :

+10% pour la fiche client (avec ses 5 attributs dont un tableau)

+10% pour le catalogue et les meuble avec l'association et les multiplicités

+20% pour la demande d'achat (attributs et associations)

+20% pour la commande d'achat (attributs et associations)- il faut que cela soit une autre classe

+20% pour le retour

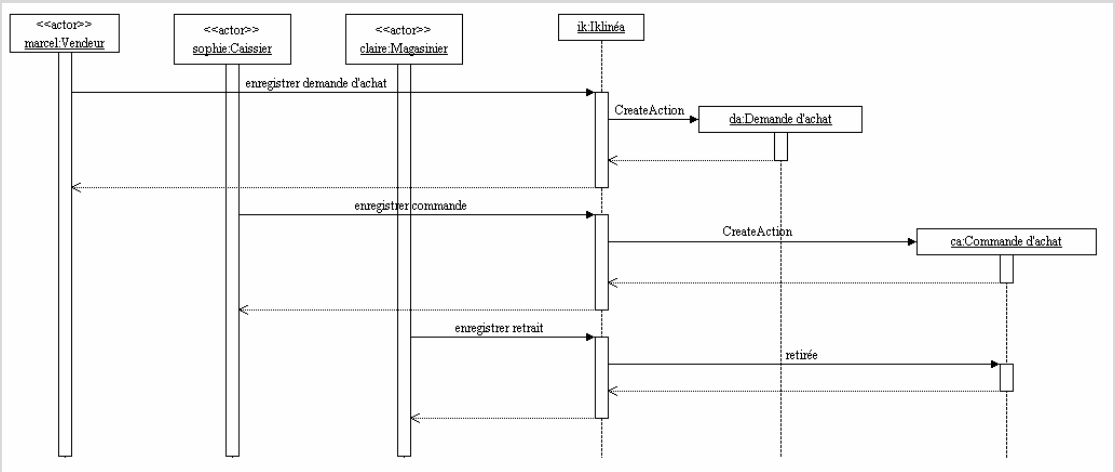
+20% pour la classe iklinéa (avec une opération par cas d'utilisation)

-20% Si il y a une super-classe entre demande d'achat et commande d'achat (cela n'est pas en analyse que l'on fera cela, en conception oui).

-10% à 20% pour toute autre faute

Question 2.4 : Réalisez un diagramme de séquence présentant tout le cycle d'achat d'un meuble par un client (scénario nominal).

Voici ma correction



Barême :

+30% pour le vendeur, le msg vers Iklinéa et la création de la demande d'achat

+30% pour le caissier, le msg vers Iklinéa et la création de la commande d'achat

+30% pour le magasinier, le msg vers Iklinéa et le msg vers commande d'achat pour préciser que la commande est retirée

+10% si paramètre et nom des instances (j'ai été fénéant)

Question 2.5 : Réalisez un test de validation couvrant la séquence que vous avez réalisée en question 2.4 et visant à révéler des fautes dans l'application qui sera réalisée. Pour chaque test, vous préciserez le cas d'utilisation correspondant.

La faute facile concerne le test des 30j (pour retourner) ou celui des 3j (pour transformer une demande en commande), tout test essayant de mettre à défaut ce délai est bon.

Exemple :

Contexte : une demande d'achat a été faite par un vendeur et date d'il y a 4 jours

Entrée : la référénce de la demande d'achat

Scénario : le caissier saisie la référence de la demande d'achat et tente de la transformer en commande d'achat.

Résultat attendu : Iklinéa doit afficher un message précisant que le délais est dépassé et doit interdire l'enregistrement de la commande.

Barême :

100% si test cohérent (scénario, contexte initial et donnée d'entrée) et essayant de faire planter l'application

50% si manque de cohérence mais l'objectif est correct

Moins de 50% si la faute ou le scénario n'est vraiment pas recherchée (mauvaise saisie dans les paramètres d'entrée)

0% sinon (si le test n'essaye pas de révéler une faute).

M1 : Ingénierie du Logiciel

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Examen de décembre 2007 (2 heures avec documents)

1. Questions de cours [5 Pts]

Q1.1 : A quoi sert un processus de développement ? Quels sont les critères permettant de comparer deux processus de développement ?

Organiser les différentes taches à réaliser permettant de passer d'un problème à une solution.

Il n'y a pas de critères bien définis permettant la comparaison des processus de développement. En cours, nous avons identifié la **visibilité du client** sur l'état d'avancement du travail, la **vérification des travaux réalisés** et la prise en compte de l'**évolution**.

Barème :

+50% pour l'explication sur les différentes taches et le pb/sol

+50% si identification moins 2 critères.

0% sinon

Q1.2 : Pourquoi préconiser l'utilisation d'un langage de modélisation pour suivre un processus de développement ? Pourquoi préconiser l'utilisation d'UML plutôt que celle d'un autre langage de modélisation ?

L'utilisation d'un langage de modélisation permet de préciser les relations de **cohérence** entre les différents produits réalisés dans chacune des taches. C'est le méga-intérêt des langages de modélisation.

UML est **standard**. C'est là son grand intérêt.

Barème :

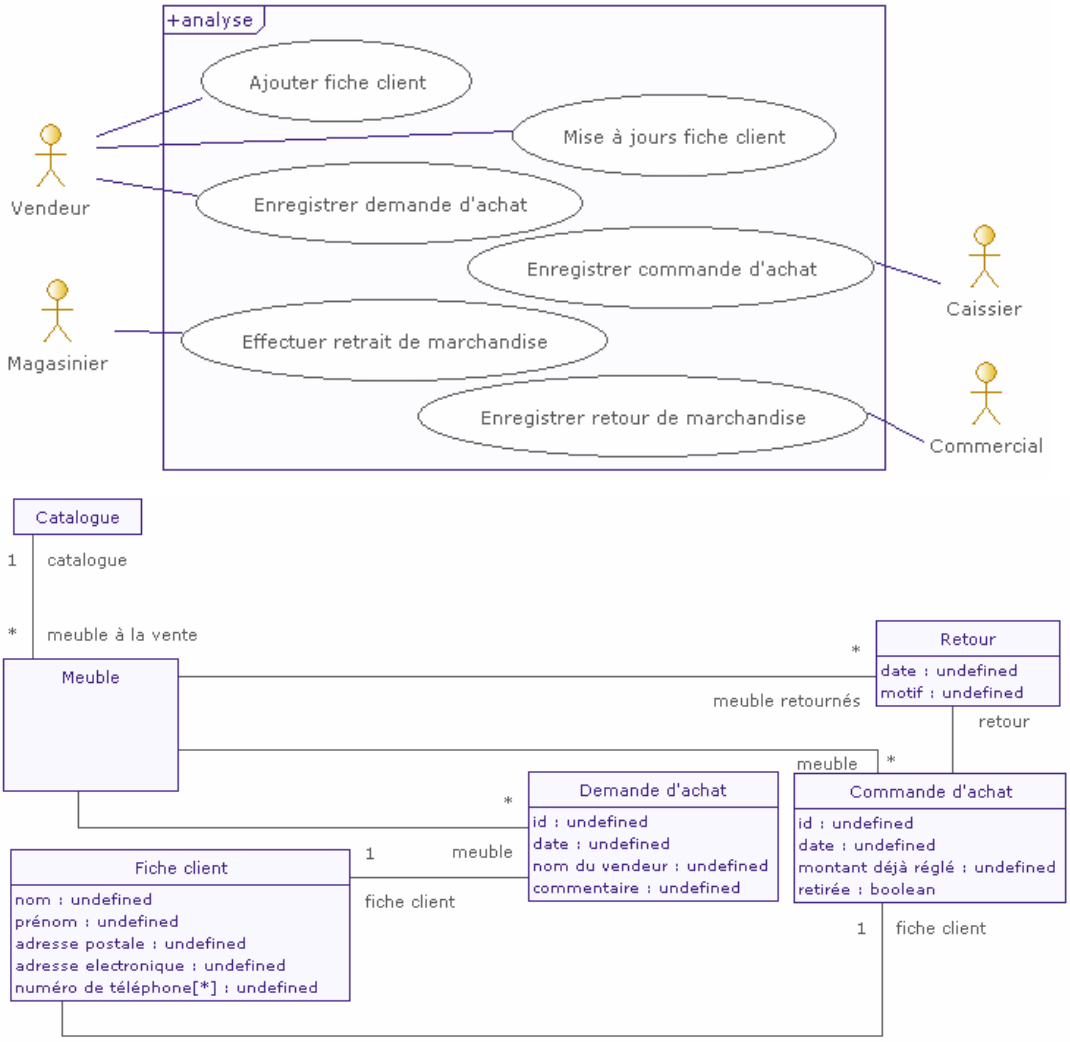
50% par réponse (cohérence + standard)

0% sinon

2. Problème: Conception de iklinéa [15 Pts]

Une équipe d'étudiants de Paris VI a réalisé la phase d'analyse du système de la société « iklinéa » permettant de suivre les achats, livraisons et retours de ses clients.

Une compréhension détaillée de ce système n'est pas nécessaire pour cet examen. Il suffit juste de savoir que les clients font d'abord des demandes d'achat auprès des vendeurs, puis transforment ces demandes en commandes auprès des caissiers, puis retirent ces commandes auprès des magasiniers et peuvent effectuer des retours auprès des commerciaux. Les figures suivantes représentent le diagramme de cas d'utilisation ainsi que le diagramme de classes d'analyse (sauf la classe Iklinéa) réalisés par ces étudiants.

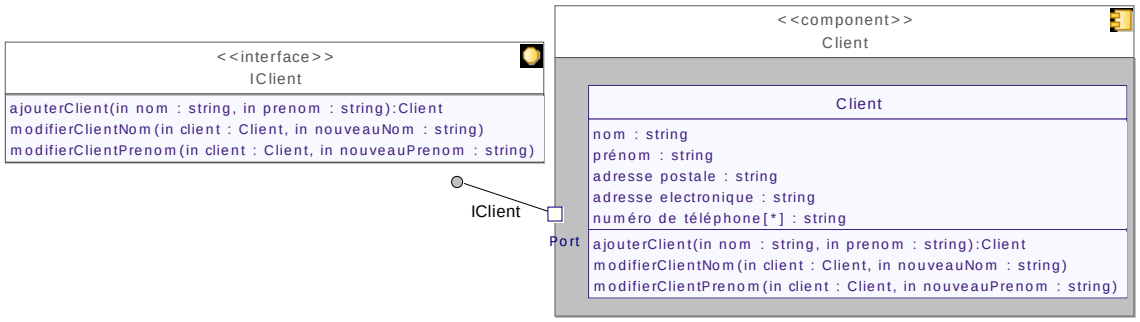


Pour le reste de l'examen, on considèrera que cette analyse est complète et on prêtera attention aux règles métier suivantes :

- Une fiche client ne peut être mise à jour qu'après avoir été créée
- Une demande d'achat doit référencer un client existant et un meuble existant (sinon sa création ne peut se faire).
- Une commande d'achat ne peut être faite que si une demande existait au préalable et qu'au moins 10% du prix a été réglé.
- Un retour ne peut être fait que pendant une période de 30j après que la commande ait été créée.

Les mêmes étudiants ont réalisé la phase de conception de cette application. Ils proposent de découper l'application en quatre composants. Un composant « **Client** » qui gère l'ensemble des fiches des clients, un composant « **Achat** » qui gère l'ajout d'une demande et sa transformation en commande, un composant « **Livraison** » qui gère les meubles, la livraison ainsi que le retour, et un composant « **IHM** » qui représente l'interface graphique de l'application.

Question 2.1 : La figure suivante présente l'interface offerte et la classe interne du composant « Client ». Corrigez cette conception en modifiant ce que vous jugerez nécessaire (justifiez vos modifications).



Voici ma correction :

(1) Il ne faut pas que la classe Client apparaisse comme type d'un paramètre de des opérations. Il est possible d'utiliser des ids.

(2) il faut une nouvelle classe qui contient les clients (association 0-*). C'est cette classe qui contient les opérations.

(3) Idéalement il faudrait ajouter des opérations pour rechercher des clients.

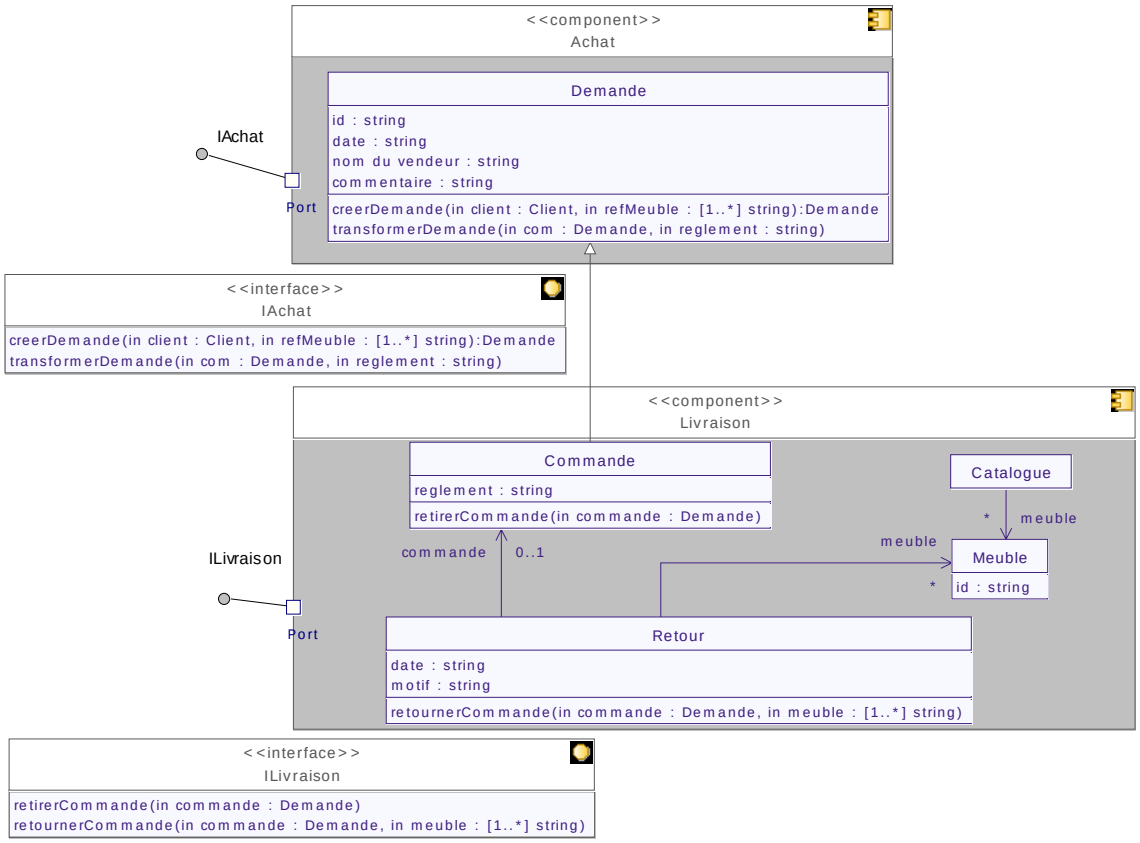
Barême :

(1) +50%

(2) +50 %

(3) +20% (si les 100% ne sont pas atteint)

Question 2.2 : La figure suivante présente les interfaces offertes et les classes internes des composant « Achat » et « Livraison ». Corrigez cette conception en modifiant ce que vous jugerez nécessaire sans toutefois déplacer les classes.



Voici ma correction :

Achat :

(A1) ne pas rendre Demande mais passer par id dans les opérations

(A2) ajouter une nouvelle classe (liste des demandes) et mettre l'opération dedans

(A3) il faut un lien fonctionnel (interface requise) vers Livraison pour que la transformation puisse se faire.

(A4) faire un lien avec client

Livraison

(L1) ne pas mettre Demande dans les types des paramètres (d'ailleurs c'est commande)

(L2) liste de commande

(L3) les opérations sont mal placées dans les classes

(L4) l'héritage est foireux

(L5) faire le lien avec les clients

(L6) il faut une opération dans l'interface offerte pour construire une commande (cette opération sera utilisée par Achat.

Barême :

10% par question

Question 2.3 : Réalisez le diagramme de structuration interne du composant « Achat ».

Voici ma correction :

Il faut voir une part instance de la classe représentant la liste des demandes.

Il doit y avoir des connecteurs de délégation entre cette part et les ports (le port offrant l'interface IAchat et le port qui requiert l'interface ILivraiso

Barême :

50% pour la part

25% par port

Question 2.4 : Réalisez un diagramme de séquence présentant un test d'intégration relatif au composant « Livraison ».

Voici ma correction :

Ce diagramme de séquence doit faire apparaître les différents appels sur le composant Livraison.

Idéalement il faudrait voir apparaître l'appel de création de la commande, l'appel du retrait de la commande (et optionnellement l'appel du retour).

Barême :

30% pour la tenue du diagramme (un instance représentant le composant Livraison, le testeur, et RIEN D'AUTRE).

70% si les messages correspondent à des appels d'opérations offertes et ressemblent à un diagramme de test d'intégration.

Question 2.5 : Lors de la phase de réalisation, chaque composant donne lieu à la création d'un package. Réalisez un diagramme de classes présentant uniquement les packages correspondant aux composants ; faisant apparaître les interfaces offertes et requises ; précisant les relations de dépendance entre ces packages ; et précisant, si possible, une classe bouchon.

Voici ma correction :

Un package par composant.

Placer les interfaces dans les packages.

Barême :

100% si pas de faute dans la cohérence (les interfaces sont positionnées et les lien de dépendance sont corrects).

0% sinon

Question 2.6 : Quelles que soient les modifications que vous avez effectué dans les questions précédentes, on considère qu'il existe une seule classe **Achat** représentant à la fois une demande et une commande et permettant de savoir si la commande a été retournée. Réalisez un diagramme présentant la machine à états de cette classe ainsi qu'un diagramme de classes représentant uniquement cette classe avec toutes ses opérations. Proposez un test unitaire pour cette classe.

Voici ma correction :

Barême :

20% pour la classe

60% pour la machine à états avec des opérations sur les transitions

20% pour le test qui devrait correspondre à un chemin dans la machine à états.

M1 : Ingénierie du Logiciel

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Examen de décembre 2007 (2 heures avec documents)

1. Questions de cours [5 Pts]

Q1.1 : A quoi sert un processus de développement ? Quels sont les critères permettant de comparer deux processus de développement ?

Organiser les différentes taches à réaliser permettant de passer d'un problème à une solution.

Il n'y a pas de critères bien définis permettant la comparaison des processus de développement. En cours, nous avons identifié la **visibilité du client** sur l'état d'avancement du travail, la **vérification des travaux réalisés** et la prise en compte de l'**évolution**.

Barème :

+50% pour l'explication sur les différentes taches et le pb/sol

+50% si identification moins 2 critères.

0% sinon

Q1.2 : Pourquoi préconiser l'utilisation d'un langage de modélisation pour suivre un processus de développement ? Pourquoi préconiser l'utilisation d'UML plutôt que celle d'un autre langage de modélisation ?

L'utilisation d'un langage de modélisation permet de préciser les relations de **cohérence** entre les différents produits réalisés dans chacune des taches. C'est le méga-intérêt des langages de modélisation.

UML est **standard**. C'est là son grand intérêt.

Barème :

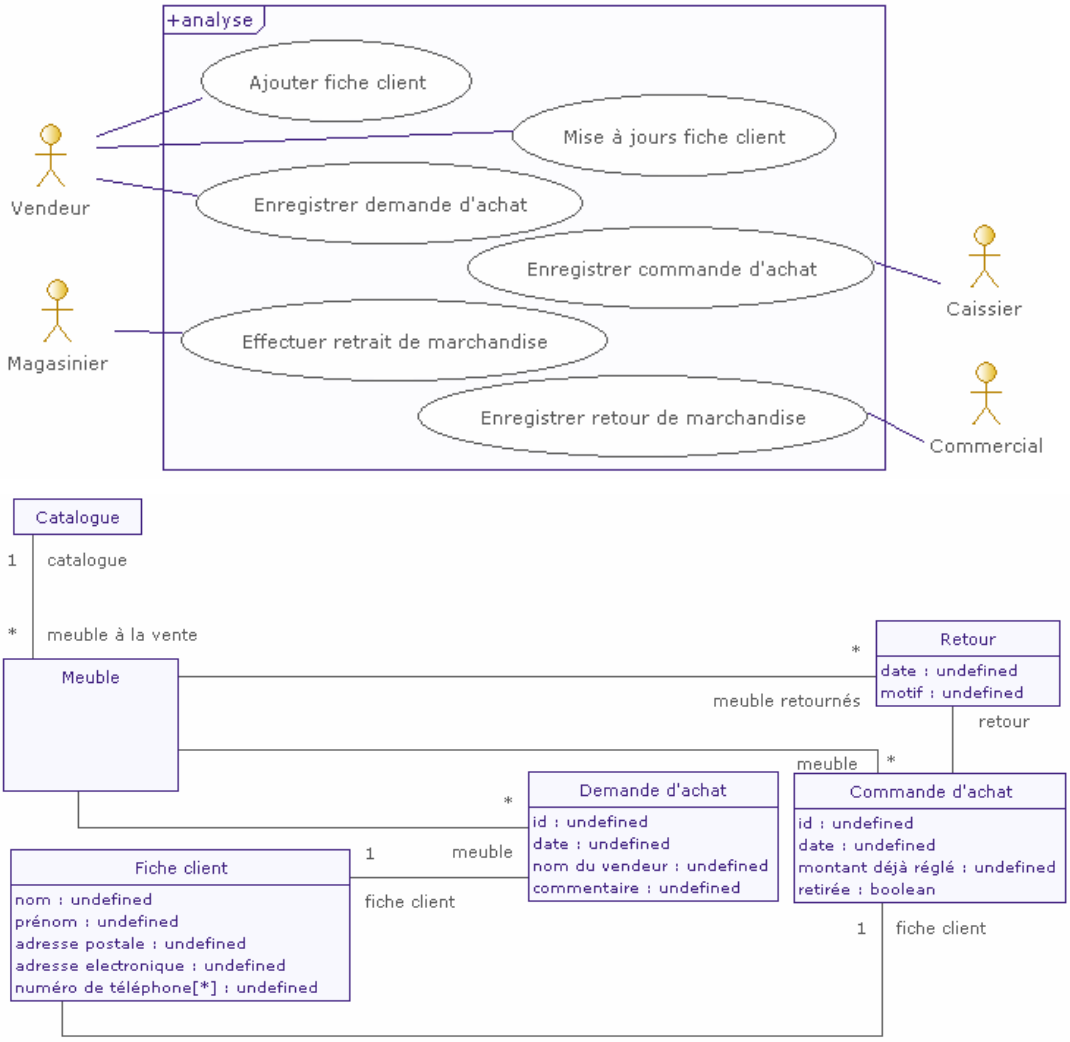
50% par réponse (cohérence + standard)

0% sinon

2. Problème: Conception de iklinéa [15 Pts]

Une équipe d'étudiants de Paris VI a réalisé la phase d'analyse du système de la société « iklinéa » permettant de suivre les achats, livraisons et retours de ses clients.

Une compréhension détaillée de ce système n'est pas nécessaire pour cet examen. Il suffit juste de savoir que les clients font d'abord des demandes d'achat auprès des vendeurs, puis transforment ces demandes en commandes auprès des caissiers, puis retirent ces commandes auprès des magasiniers et peuvent effectuer des retours auprès des commerciaux. Les figures suivantes représentent le diagramme de cas d'utilisation ainsi que le diagramme de classes d'analyse (sauf la classe Iklinéa) réalisés par ces étudiants.

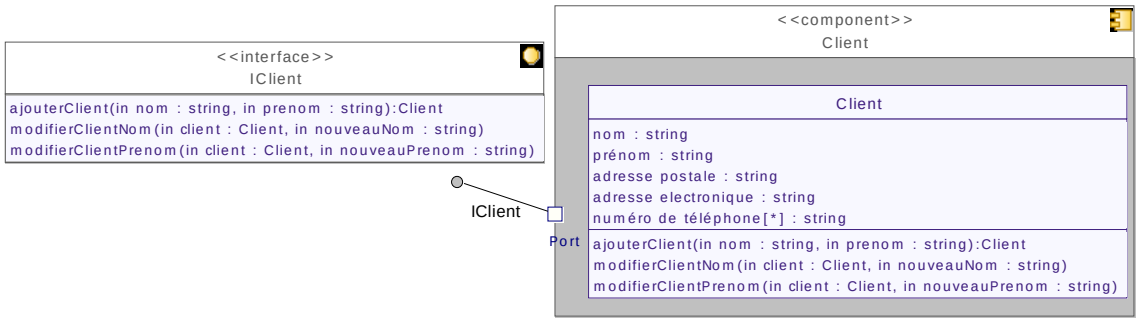


Pour le reste de l'examen, on considèrera que cette analyse est complète et on prêtera attention aux règles métier suivantes :

- Une fiche client ne peut être mise à jour qu'après avoir été créée
- Une demande d'achat doit référencer un client existant et un meuble existant (sinon sa création ne peut se faire).
- Une commande d'achat ne peut être faite que si une demande existait au préalable et qu'au moins 10% du prix a été réglé.
- Un retour ne peut être fait que pendant une période de 30j après que la commande ait été créée.

Les mêmes étudiants ont réalisé la phase de conception de cette application. Ils proposent de découper l'application en quatre composants. Un composant « **Client** » qui gère l'ensemble des fiches des clients, un composant « **Achat** » qui gère l'ajout d'une demande et sa transformation en commande, un composant « **Livraison** » qui gère les meubles, la livraison ainsi que le retour, et un composant « **IHM** » qui représente l'interface graphique de l'application.

Question 2.1 : La figure suivante présente l'interface offerte et la classe interne du composant « Client ». Corrigez cette conception en modifiant ce que vous jugerez nécessaire (justifiez vos modifications).

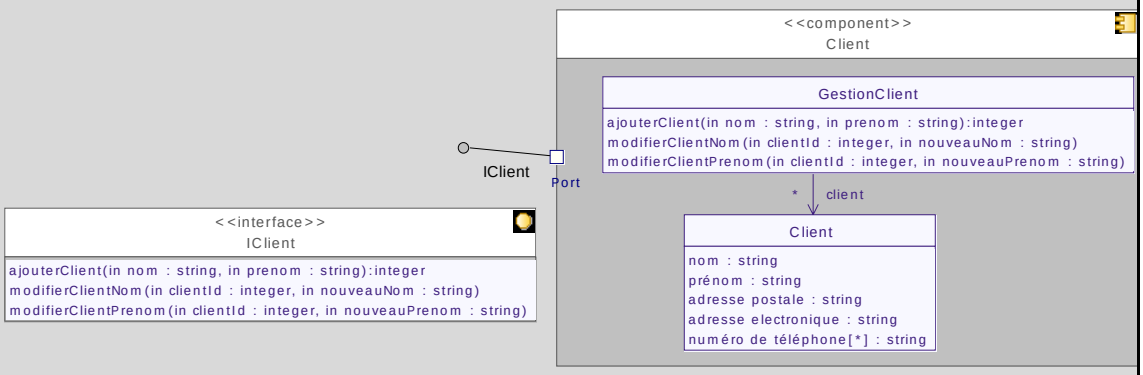


Voici ma correction :

(1) Il ne faut pas que la classe Client apparaisse comme type d'un paramètre de des opérations. Il est possible d'utiliser des ids.

(2) il faut une nouvelle classe qui contient les clients (association 0-*). C'est cette classe qui contient les opérations.

(3) Idéalement il faudrait ajouter des opérations pour rechercher des clients.



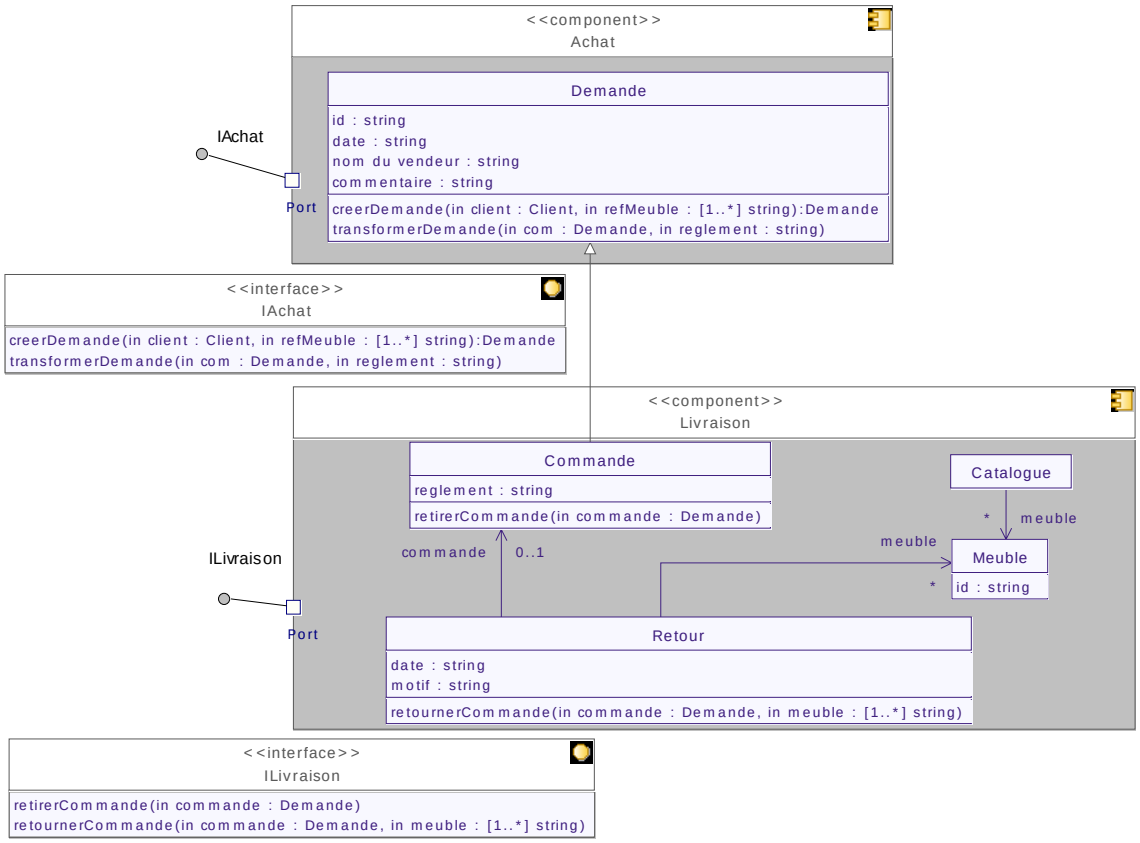
Barême :

(1) +50%

(2) +50 %

(3) +20% (si les 100% ne sont pas atteint)

Question 2.2 : La figure suivante présente les interfaces offertes et les classes internes des composant « Achat » et « Livraison ». Corrigez cette conception en modifiant ce que vous jugerez nécessaire sans toutefois déplacer les classes.



Voici ma correction :

Achat :

(A1) ne pas rendre Demande mais passer par id dans les opérations

(A2) ajouter une nouvelle classe (liste des demandes) et mettre l'opération dedans

(A3) il faut un lien fonctionnel (interface requise) vers Livraison pour que la transformation puisse se faire.

(A4) faire un lien avec client

Livraison

(L1) ne pas mettre Demande dans les types des paramètres (d'ailleurs c'est commande)

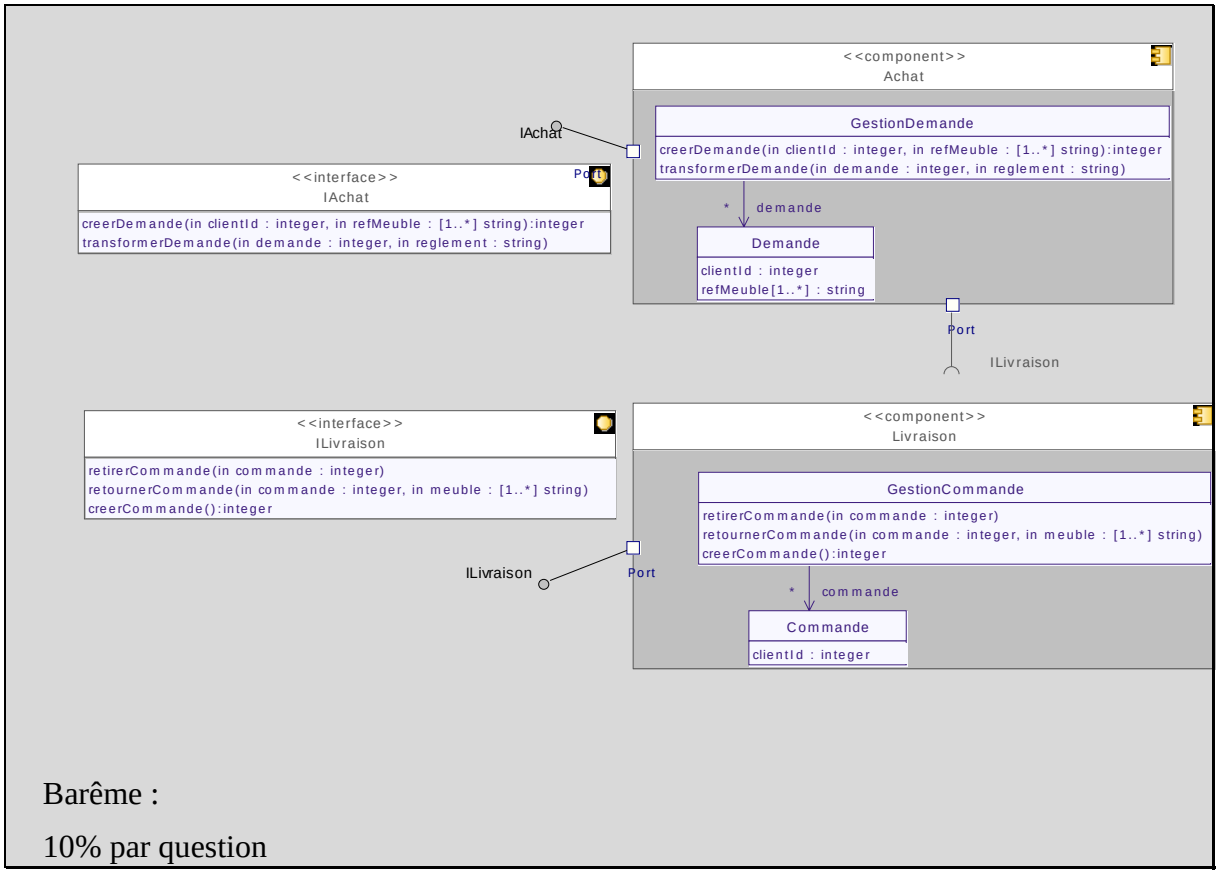
(L2) liste de commande

(L3) les opérations sont mal placées dans les classes

(L4) l'héritage est foireux

(L5) faire le lien avec les clients

(L6) il faut une opération dans l'interface offerte pour construire une commande (cette opération sera utilisée par Achat.



Barême :

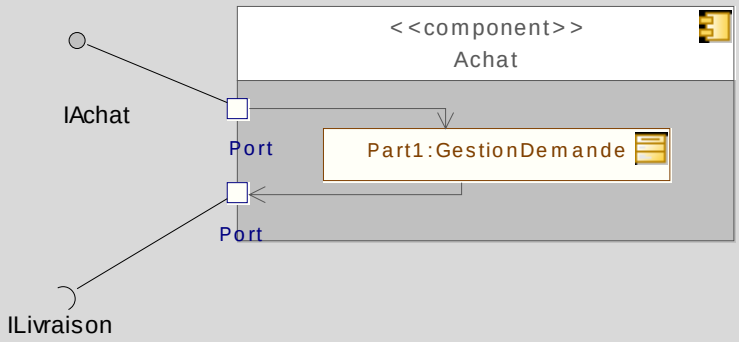
10% par question

Question 2.3 : Réalisez le diagramme de structuration interne du composant « Achat ».

Voici ma correction :

Il faut voir une part instance de la classe représentant la liste des demandes.

Il doit y avoir des connecteurs de délégation entre cette part et les ports (le port offrant l'interface IAchat et le port qui requiert l'interface ILivraiso



Barême :

50% pour la part

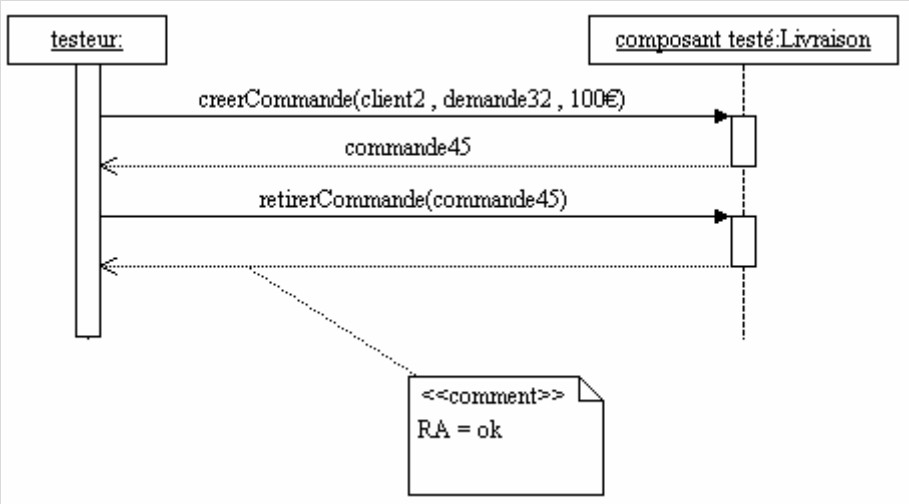
25% par port

Question 2.4 : Réalisez un diagramme de séquence présentant un test d'intégration relatif au composant « Livraison ».

Voici ma correction :

Ce diagramme de séquence doit faire apparaître les différents appels sur le composant Livraison.

Idéalement il faudrait voir apparaître l'appel de création de la commande, l'appel du retrait de la commande (et optionnellement l'appel du retour).



Barême :

30% pour la tenue du diagramme (un instance représentant le composant Livraison, le testeur, et RIEN D'AUTRE).

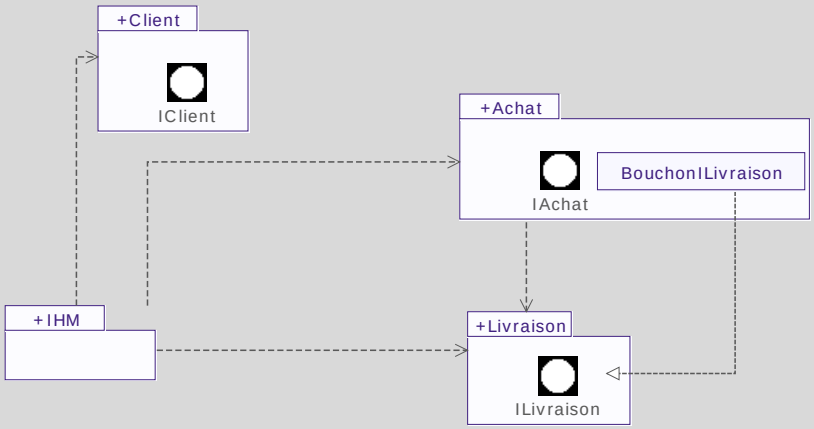
70% si les messages correspondent à des appels d'opérations offertes et ressemblent à un diagramme de test d'intégration.

Question 2.5 : Lors de la phase de réalisation, chaque composant donne lieu à la création d'un package. Réalisez un diagramme de classes présentant uniquement les packages correspondant aux composants ; faisant apparaître les interfaces offertes et requises ; précisant les relations de dépendance entre ces packages ; et précisant, si possible, une classe bouchon.

Voici ma correction :

Un package par composant.

Placer les interfaces dans les packages.



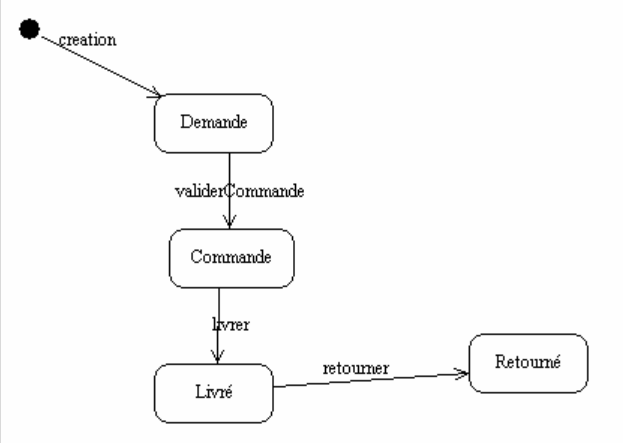
Barême :

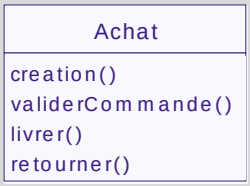
100% si pas de faute dans la cohérence (les interfaces sont positionnées et les lien de dépendance sont corrects).

0% sinon

Question 2.6 : Quelles que soient les modifications que vous avez effectué dans les questions précédentes, on considère qu'il existe une seule classe **Achat** représentant à la fois une demande et une commande et permettant de savoir si la commande a été retournée. Réalisez un diagramme présentant la machine à états de cette classe ainsi qu'un diagramme de classes représentant uniquement cette classe avec toutes ses opérations. Proposez un test unitaire pour cette classe.

Voici ma correction :





Test :

Achat a = new Achat() ;

a.livrer() ;

RA = « livraison interdite car demande non validée »

Barême :

20% pour la classe

60% pour la machine à états avec des opérations sur les transitions

20% pour le test qui devrait correspondre à un chemin dans la machine à états.

M1 : Ingénierie du Logiciel

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Examen de Janvier 2008 (2 heures avec documents)

1. Questions de cours [4 Pts]

Q1.1 : Quelle est la critique principale faite au cycle en V ? Quels sont les avantages et les inconvénients des cycles itératifs ?

La critique principale du cycle en V est que le client n'a que peu de visibilité sur l'état d'avancement ou que l'on a pas de gestion des évolutions.

Les cycles itératifs ont pour avantage d'améliorer grandement la visibilité du client quant à l'état d'avancement mais aussi de pouvoir intégrer de nouveaux besoins. L'inconvénient est qu'il est possible que les coûts de re-conception (à chaque cycle) finissent par prendre de l'ampleur. Il est aussi très complexe d'ordonner les itérations (par quoi commencer).

Barème :

+40% pour un des avantages du cycle en V

+35% pour un des avantages du cycle itératif

+25% pour un inconvénient du cycle itératif

Q1.2 : Rappelez la contrainte de conception fixée pour la construction des composants ? Quelle était la justification de cette contrainte ? Faudra-t-il la respecter dans tous vos futurs développements ?

La contrainte posée est celle d'interdire les dépendences structurelles entre composants (pas d'association, pas d'héritage, etc.).

La justification était que les composants sont autonomes dans leur structure. Ils communiquent avec d'autre composants.

Cette contrainte est pédago. Elle peut être levée.

Barème :

+40% pour dépendance structurelle

+40% pour autonomie de structure et composant via interface

+20% pour dire que la contrainte est optionnelle.

0% sinon

Q1.3 : Quels types de diagrammes UML vous paraissent les mieux adaptés à la définition d'un comportement ?

Etat, séquence, activité

Barème :

100% si état et séquence (activité n'a pas été réellement vue)

75% si état seulement

50% si séquence seulement

0% sinon

Q1.4 : Pourquoi n'est-il pas possible de connecter des composants entre eux (à l'aide de connecteurs) alors qu'il est possible de connecter des « part » dont les types sont des composants ?

Les composant sont des classifiers. Il est simplement possible de définir leurs interfaces offertes et requises (c'est donc au niveau des types). La connection se fait entre parts dans un composant (au niveau des spécifications d'instance).

Barème :

+100% dès qu'il y a une explication que les interactions entre composants sont spécifiées à l'aide des interfaces offertes et requises

0% sinon (si explication très compliquée à comprendre).

2. Problème: Conception de FreeBike [16 Pts]

Une équipe d'étudiants de Paris VI est en train d'élaborer le système de la société « FreeBike » permettant à ses abonnés d'emprunter facilement des vélos en libre service.

Une compréhension détaillée de ce système n'est pas nécessaire pour cet examen. Il suffit de savoir que les utilisateurs de « FreeBike » doivent d'abord s'abonner. Le système « FreeBike » gère plusieurs stations locales. Chaque station locale contient une borne et plusieurs bornettes où sont garés les vélos. Un utilisateur abonné peut emprunter un vélo à une station locale et le retourner à n'importe quelle autre station. Un utilisateur abonné ne peut emprunter qu'un seul vélo à un moment donné. Les administrateurs peuvent gérer les stations et les vélos. Ils peuvent ainsi savoir si un vélo est actuellement emprunté ou s'il est garé dans une bornette. Ils peuvent aussi savoir quels sont les états des bornes (pleine ou vide) et des bornettes (en panne, libre ou occupée).

Le diagramme de cas d'utilisation et le diagramme de classes des figures suivantes présentent le résultat de la phase d'analyse réalisée par les étudiants. Pour le reste de l'examen, on considèrera que cette analyse est complète.

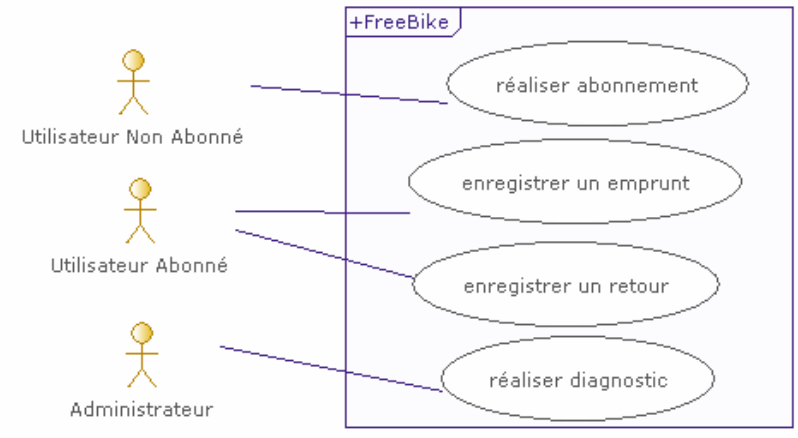


Diagramme de cas d'utilisation de la phase d'analyse

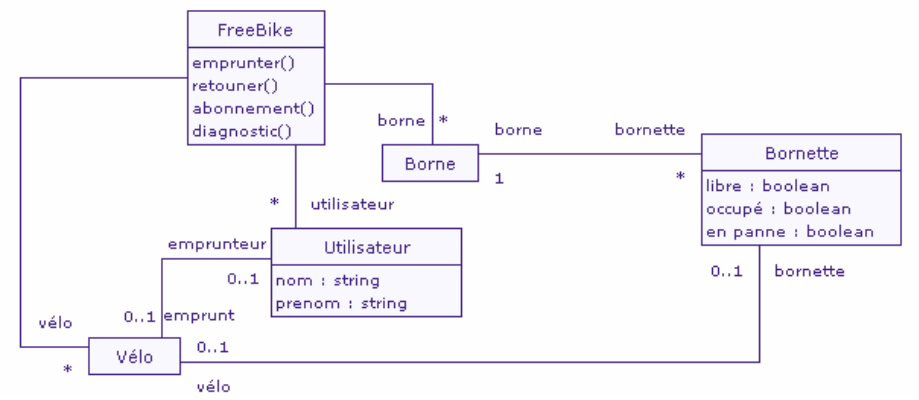


Diagramme de classes de la phase d'analyse

Les étudiants ont décidé de réaliser ce système en proposant une découpe en deux composants principaux (un composant représentant le serveur « ServeurCentral », et un composant représentant une station « StationLocale ») et en deux composants d'interface graphique (un composant pour l'interface des administrateurs et un composant pour l'interface des utilisateurs).

Question 2.1 : La figure suivante présente une partie des composants ServeurCentral et StationLocale. L'utilisateur, via son interface graphique, s'adresse à une station locale pour s'abonner puis pour emprunter et retourner un vélo. La station locale s'adresse au serveur central afin d'enregistrer les informations concernant l'emprunt du vélo. Représentez, à l'aide d'un diagramme de séquence, un scénario montrant les interactions entre les composants illustrant l'emprunt et le retour d'un vélo par un utilisateur abonné.

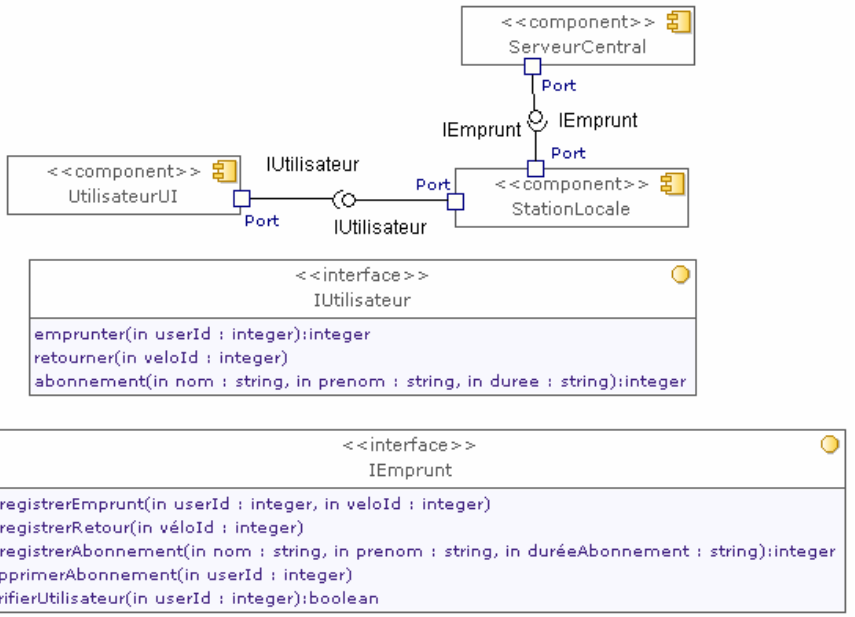
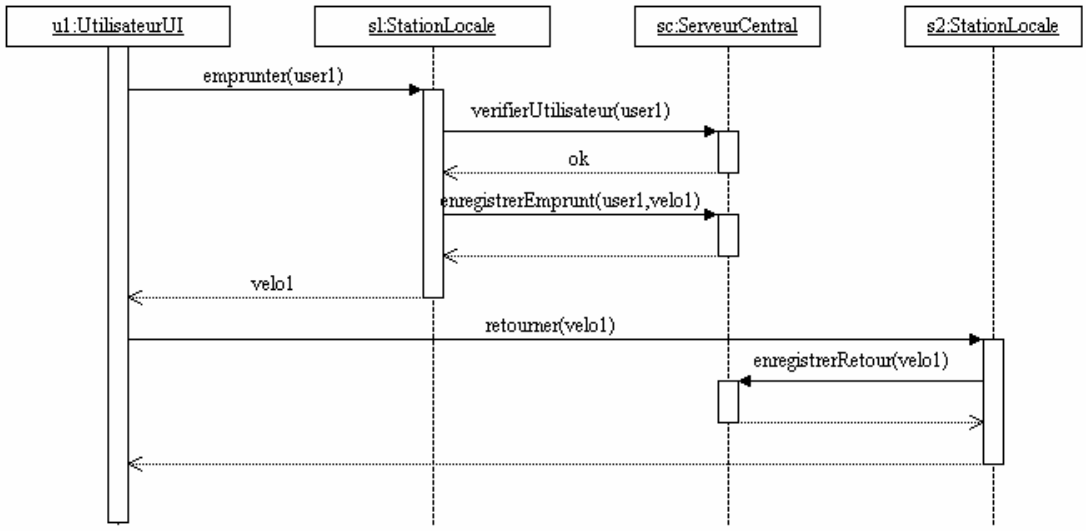


Diagramme de composant de la phase de conception présentant l'aspect utilisateur

Voici ma correction :



Barême :

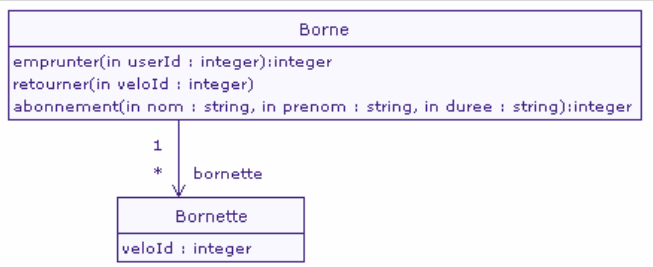
+50% pour l'emprunt (message emprunter,vérifier, enregistrerEmprunt)

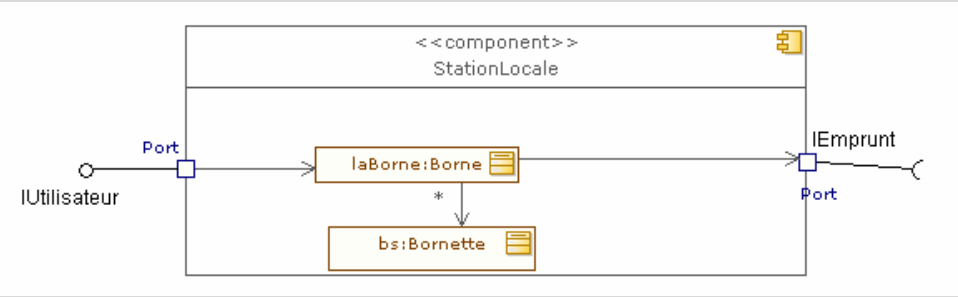
+25% pour retour (message retourner, enregistrerRetour)

+25% pour la bonne tenue du diagramme

Question 2.2 : Les classes Borne et Bornette appartiennent au composant StationLocale. Présentez la structure interne du composant StationLocale tout en proposant une conception des classes Borne et Bornette (opérations et attributs). Vous présenterez les nouvelles classes et/ou associations dont vous aurez besoin.

Voici ma correction :





Barême :

+30% pour mettre les opérations de l'interface dans Borne (à la limite retourner peut être positionnée dans Bornette)

+10% pour l'id vélo dans Bornette (le vélo garé)

+40% pour la part Borne connectée au 2 ports

+10% pour la part Bornette

+10% pour la bonne tenue du diagramme

Question 2.3 : La figure suivante présente l'autre partie des composants ServeurCentral et StationLocale. Cette partie concerne les aspects d'administration de « FreeBike ». L'administrateur, via son interface graphique, peut s'adresser directement au serveur central ou même à certaines stations locales. Présentez la structure interne du composant ServeurCentral tout en proposant une conception des classes Utilisateur et Vélo appartenant à ce composant. Vous présenterez les nouvelles classes ou associations dont vous aurez besoin.

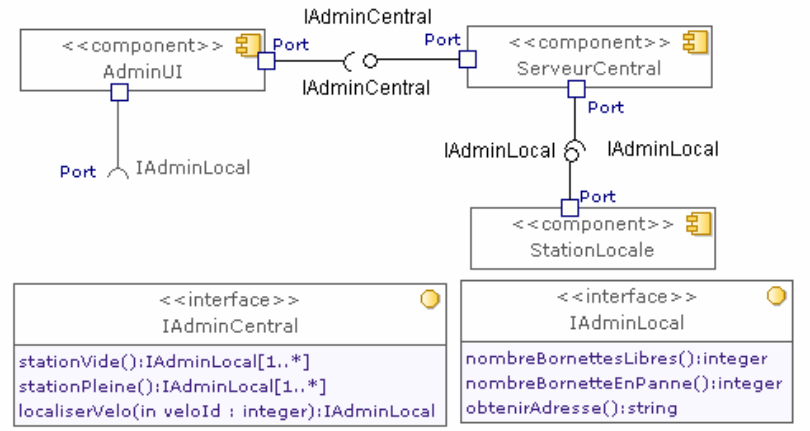
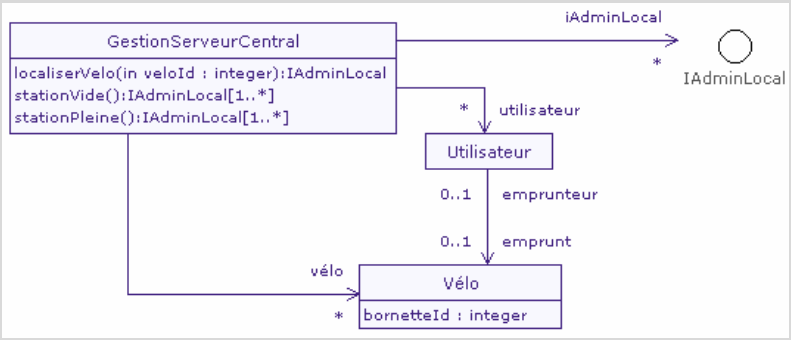
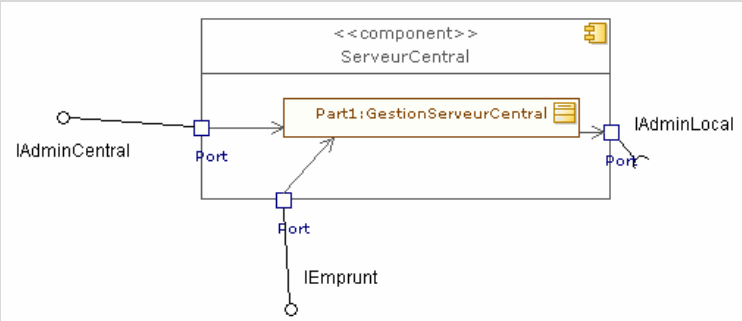


Diagramme de composants de la phase de conception présentant l'aspect administration

Voici ma correction :





Barême :

+30% pour le gestionnaire (ou les si plusieurs classes ont été proposées)

+20% pour l'association vers l'interface

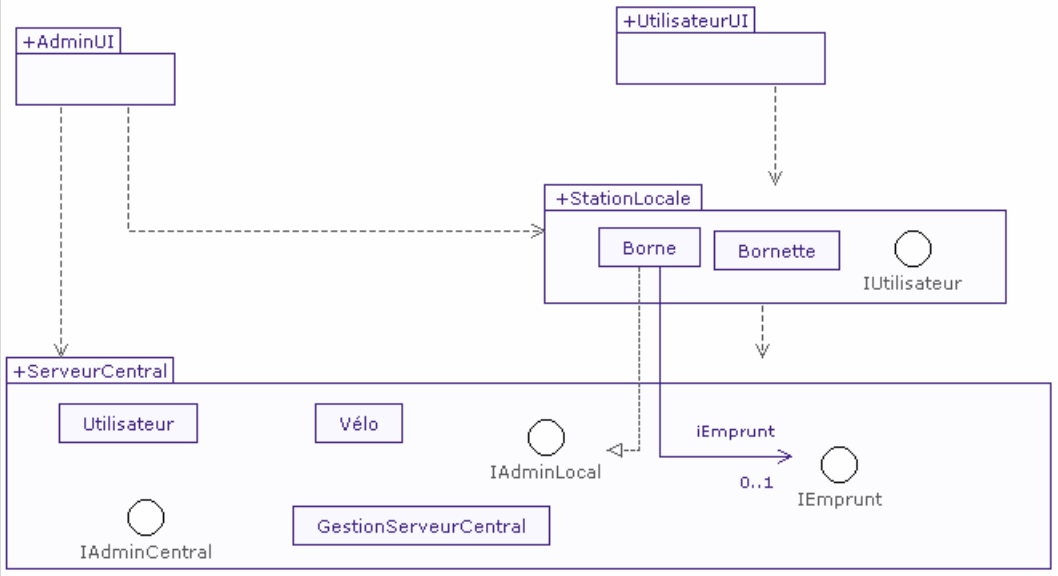
+10% pour l'id vers bornette

+40% pour la part avec les connecteurs

Question 2.4 : Présentez à l'aide d'un ou de plusieurs diagrammes de classes la réalisation des composants StationLocale et ServeurCentral. Rappelons que les cycles de dépendance entre packages sont interdits.

Voici ma correction :

Il faut simplement bien loger les interfaces et les classes afin de ne pas faire de cycle de dépendance. D'autres solutions existent.



Barême :

+40% si on voit toutes les interfaces et toutes les classes.

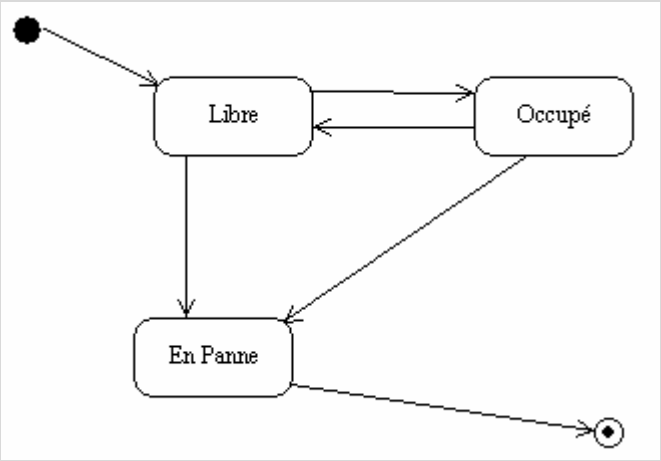
+50% pour si pas de cycle

+10% si respect de la syntaxe

Question 2.5 : Réalisez la machine à état de la classe Bornette et présentez un test unitaire.

Voici ma correction :

En gros ca doit donner un truc comme ca (trois états)



Barême :

+50% pour la machine à état

+50% pour le test (sous forme de pseudo code). Il faut que le test concerne la bornette (genre libérer alors que la borne est en panne !!!)

M1 : Ingénierie du Logiciel

UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Partiel de novembre 2007 (2 heures avec documents)

1. Questions de cours [5 Pts]

Répondez de façon précise et concise aux questions.

Q1.1 : Quelles sont les principaux concepts caractéristiques du modèle de composants d'UML ?

Mise en avant des notions architecturales suivantes :

- Composant : une brique logicielle réutilisable
- Interfaces (requise & offerte) : connecteurs entre composants
- Topologie de connection : spécifiée **séparément** à l'aide des diagrammes de structure interne

Barème :

30% composant

20% + 20% interfaces requise et offertes.

30% topologie

Q1.2 : A quelles conditions est-il possible de substituer un composant par un autre, sans modifier le comportement de l'application, ni les autres composants ?

1. Respect des interfaces requises et offertes (test syntaxique).
2. Respect du « contrat comportemental », i.e. séquences d'action légales sur le composant (cf. classes bouchon, exemple du Fichier open/(read/write)*/close)

Barème :

50% interfaces

50% contrat

0% sinon

Q1.3 : Quand peut-on dire que la phase d'analyse est terminée ?

Quand la mission du système a été intégralement spécifiée (complète et cohérente vis-à-vis du cahier des charges), et que le client a paraphé les tests de validation couvrant les fonctionnalités à développer.

Barème :

50% specification complète et cohérente

50% tests de validation vus par le client

0% sinon

Q1.4 (2 Pts): Quels sont les objectifs, et en quoi consiste une méthodologie de développement de logiciel ?

Toute méthodologie a pour objectif de produire un logiciel répondant au mieux au cahier des charges(XX), tout en améliorant la productivité(XX).

Elle définit :

- (XX) des étapes (e.g. Analyse, Tests Validation, Conception…), le plus souvent aussi une découpe en itérations plus ou moins longues.
- (XX) des moyens (e.g. diagrammes UML, tests, fiche détaillée, génération de code…),
- (XX) des produits intermédiaires du développement (c.àd. les délivrables : e.g. document d'analyse, tests de validation…).

(XX) Elle peut aussi définir une structure générale pour la gestion de projet (petites ou grosses équipes, organisation du travail e.g. Approche XP),

Barème :

+20% par point (XX) cité sur l'objectif

+15% par point (XX) cité sur le contenu

0% sinon.

2. Problème: Analyse de MyChess [15 Pts]

Une société proposant un site web dédié au jeu d'échecs (articles, tutoriaux, compte rendus des compétitions…) désire mettre en ligne une application de jeu d'échecs.

L'objectif est de permettre d'une part aux visiteurs du site de participer sans inscription à une partie contre d'autres joueurs en mode « invité » et, d'autre part, aux utilisateurs référencés du site (payant un abonnement) de participer à des parties ainsi que d'évaluer leur classement ELO (standard de notation du niveau aux échecs). Le classement ELO est calculé à partir des défaites et victoires obtenues dans les parties les opposant à d'autres joueurs référencés du site.

Le système devra être développé comme une application client riche, ne nécessitant pas d'installation particulière.

Le jeu d'échecs est un jeu à deux joueurs, l'un jouant les pièces blanches et l'autre les pièces noires. Le jeu se déroule sur un plateau de 64 cases, chacune pouvant être occupée par une pièce. Les pièces sont le Roi, la Dame, les Fous, les Cavaliers, les Tours et les Pions. La position initiale des pièces est fixée. Chaque pièce est soumise à des règles de déplacement particulières, qui dépassent le cadre de ce sujet de partiel.

Les joueurs jouent à tour de rôle, un coup consiste à déplacer une pièce de sa couleur. Le joueur Blanc joue le premier. La partie se termine quand un des joueurs mate son adversaire (victoire), quand une situation de pat est obtenue (match nul), ou que l'un des deux joueurs abandonne (victoire de l'adversaire). Le nul peut aussi être proposé à l'adversaire, qui peut l'accepter ou continuer à jouer.

Le jeu d'échecs, a fortiori en ligne, se caractérise par l'utilisation d'une pendule pour limiter la durée de la partie. Chaque joueur dispose du même temps de réflexion pendant lequel il doit jouer tous ses coups. En d'autres termes, la pendule du joueur Blanc est décrémentée quand c'est son tour de jouer, et celle du joueur Noir quand c'est aux noirs de jouer. Les parties sont souvent rapides, d'une durée de 1, 2, 3, 5 ou 10 minutes par pendule en général. Plus rarement, on peut manuellement spécifier une durée arbitraire pour la partie. Si la pendule d'un joueur atteint 0, il perd la partie. La partie peut également se terminer avant à travers les règles citées ci-dessus (mat, pat, abandon ou nul).

De plus, il est possible de spécifier un temps par coup, permettant de favoriser les parties ou beaucoup de coups sont joués. On spécifie alors un intervalle de 1, 2 ou 5 secondes supplémentaires par coup. Les règles sont les mêmes que précédemment, mais à chaque coup joué, on ré-incrémente la pendule du joueur de l'intervalle spécifié.

Les joueurs, qu'ils soient ou non « invité », devront pouvoir défier d'autres joueurs. Un défi consiste à spécifier les paramètres d'une partie (pendules, temps additionnel) et à lancer le défi à un ou plusieurs autres joueurs. L'utilisateur pourra sélectionner les joueurs visés par le défi parmi les joueurs connectés au site et ne participant pas actuellement à une partie. On proposera l'option de défier tous les joueurs connectés répondant à ces critères. Les défis reçus par un joueur sont affichés et mis à jour en temps réel, tant qu'il ne participe pas à une partie. Il suffit de sélectionner un défi et d'accepter pour entamer la partie.

Au cours de la partie, une zone de discussion permettra de dialoguer avec son adversaire. A la fin d'une partie, on pourra proposer à l'autre joueur de rejouer avec les mêmes paramètres, il pourra accepter ou décliner. Tout joueur pourra également observer les parties en ligne des autres joueurs, mais il n'est alors pas possible de dialoguer.

Question 2.1 : 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.

Acteurs : Joueur, Joueur référencé (extends Joueur)

Use case principaux (Joueur) : Défier, Jouer partie, Observer partie.

Use case principaux (Joueur Ref) : évaluer ELO.

On peut aussi détailler un peu plus :

Use case annexes (extend sur Jouer partie) : dialoguer, abandonner, gérer proposition nul, rejouer

On évitera de représenter par des use case : Définir pendule, définir temps additionnel, jouer un coup, décrémenter pendule, etc… Car ces actions sont de grain trop fin et à inclure dans jouer partie ou défier. L'<<include>> n'est pas non plus très indiqué, sauf si on réutilise la fonctionnalité en deux endroits.

On pourrait par contre éventuellement décomposer les défis en « Envoyer défi » et « Recevoir défi »

Barême :

+10% acteur Joueur

+10% use case observer partie

+30% acteur joueur référencé + use case ELO + héritage/ou autre spécification correcte des use case offerts

+20% use case défier,

+20% jouer partie,

+10% use case annexes détaillés de certaines parties, avec <<extend>> corrects (!direction de la flèche)

-30% par acteur supplémentaire

-30% si niveau de détail trop fin (jouer un coup…)

-20% par héritage, include ou extend injustifiable ou autre incohérence/mésusage d'UML.

-10% si on ne précise pas qui fait l'action dans le scenario

-10% aucun commentaire/aucun texte pour accompagner le diagramme.

Question 2.2 : Précisez la feuille détaillée (acteurs concernés, hypothèses/pré-conditions, post-conditions, scénario nominal, alternatives, exceptions) du (ou des) cas d'utilisation(s) correspondant à la phase de défi qui précède une partie, jusqu'à l'instant ou le joueur Blanc est en situation de jouer le premier coup.

Hypothèse : Les joueurs invité « Bob » et « Joe » sont connectés à MyChess.

Pré : néant (ou on peut mettre l'hypothèse en Pré)

Post : « Bob » et « Joe » sont en train de *Jouer une partie (*cf. cas d'utilisation CU3*)*

Scénario :

1. Bob choisit de lancer un défi.
2. Le système demande de paramétrer la pendule (cocher 1,2,3,5,10 min. ou remplir un champ de saisie) et le temps additionel (1,2, 5 secondes).
3. Le système affiche la liste des utilisateurs connectés.
4. Bob choisit Joe dans la liste.
5. Le système affiche le défi sur l'interface de Joe.
6. Joe relève le défi.

Alternative A1 : Défi à tous les joueurs.

A1.4. Bob choisit de défier tous les concurrents.

A1.5. Tous les joueurs choisis reçoivent le défi

A1.6. Le premier des joueurs à relever le défi engage une partie contre Bob.

Exception E1 : Défi refusé (c'est une exception car cela viole la Post-condition)

E1.6 Le défi n'est pas relevé par Joe, aucune partie n'est engagée.

Barême :

Cette question est très délicate à corriger. Il faut donc vérifier les points suivants.

+40% cohérence globale du texte, utilisation correcte des champs Pré/Post/Scenario etc...

+30% séquence principale défi 1vs1

+20% séquence tous les joueurs

+20% séquence défi refusé.

-10% sur chaque séquence mal expliquée/peu détaillée

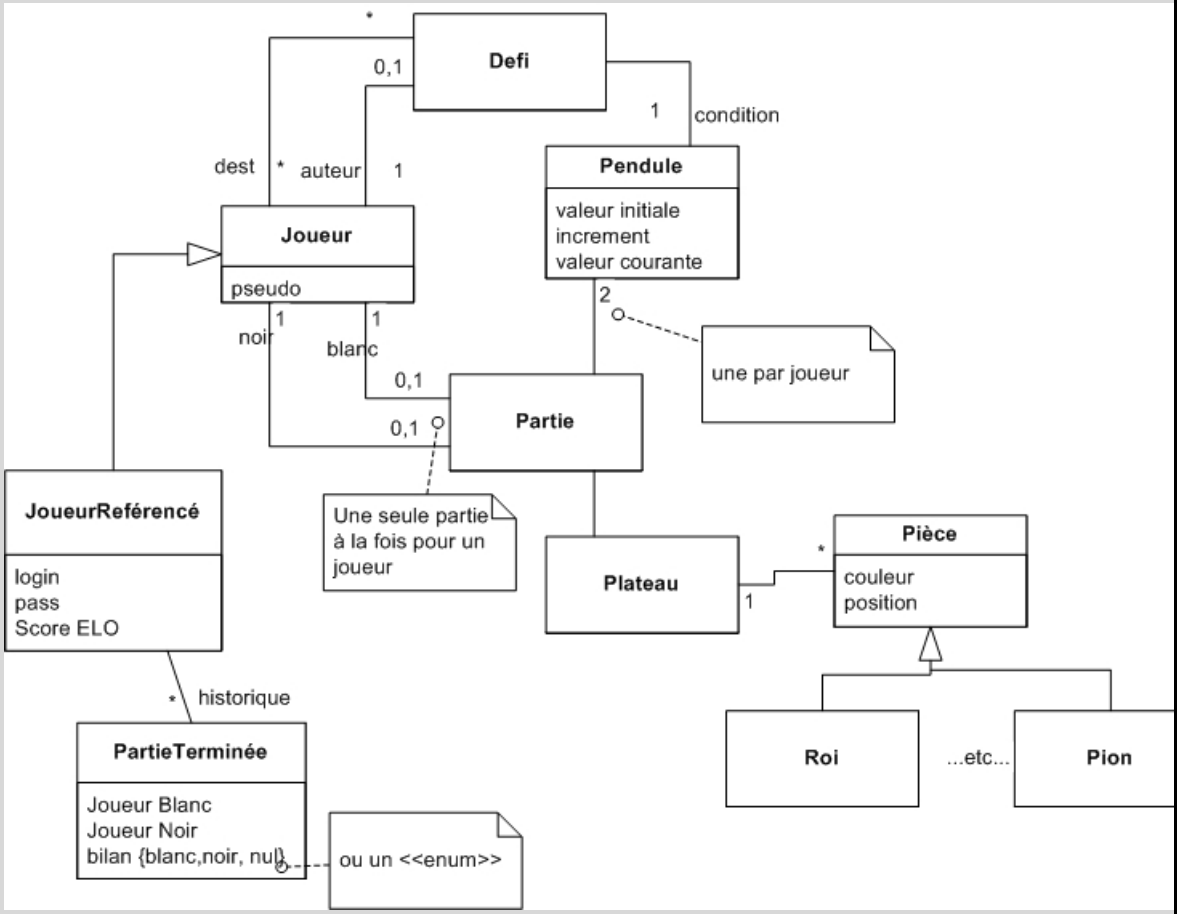
-50% si on a découpé en plusieurs use case en question 1, mais que leur description détaillée n'est pas cohérente avec le diagramme.

-50% si les pré et post condition ne sont pas cohérente avec le scénario (ex : une précondition ne se teste pas dans le scénario)

-50% si le scénario fait apparaître des interactions entre des entités autres que les acteurs et le système

Question 2.3 : 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 », MyChess, introduite dans l'approche en V du module.

Voici ma correction :



Barême :

+20% pour Joueur

+20% pour Joueur référencé + classement ELO (historique optionel)

+15% pour les pièces et l'héritage (plateau est optionel)

+15% pour la pendule et ses attributs (valeur courante et/ou initiale, et l'incrément)

+15% pour la partie et ses relations avec joueur et pendule

+15% pour le défi et ses relations (source et destination)

-15% si associations orientées, compositions etc…

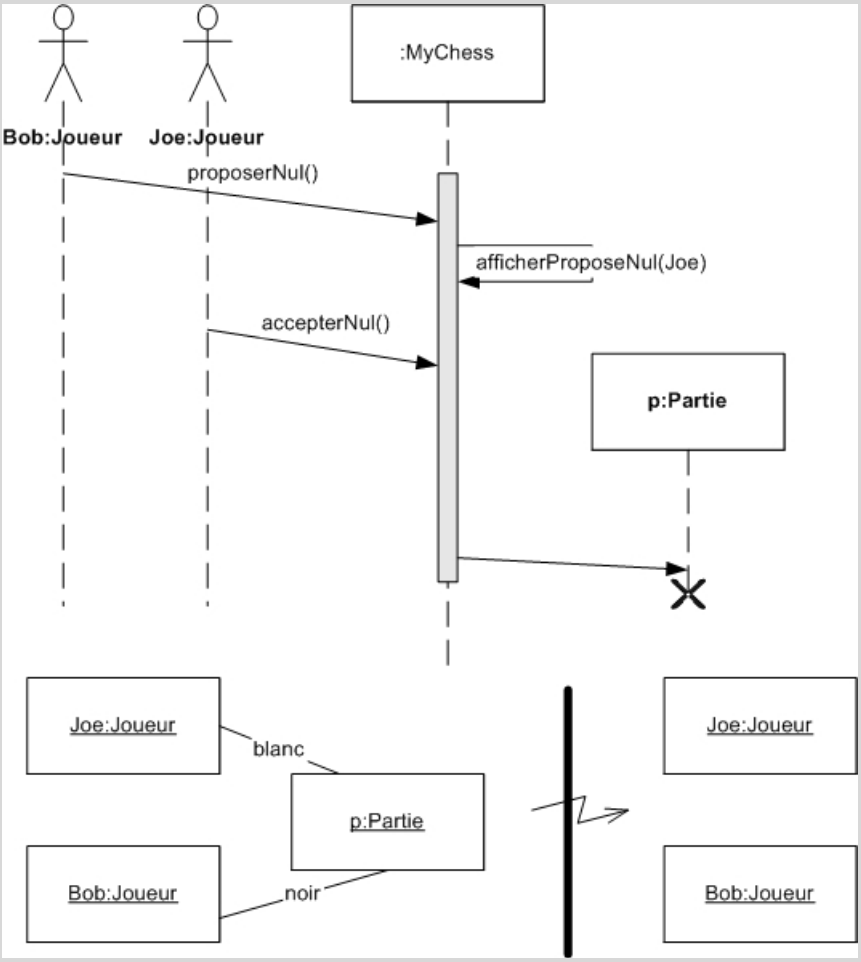
-15% si opérations sur les classes

-10% à 20% pour toute autre faute ou aberration

-10% aucun commentaires, aucune note.

Question 2.4 : Réalisez un diagramme de séquence présentant le déroulement (scénario nominal) de la fin d'une partie s'achevant sur un nul (sur proposition d'un joueur, pas un pat). On considèrera qu'il s'agit de deux joueurs invités. Représentez par un diagramme d'objets l'état des objets pertinents du système, avant et après ce scénario.

Voici ma correction



Barême (sur 110%):

+10%*3 par ligne de vie identifiée (joueur j1, j2, système)

+30% invocation du J1 puis du J2

+10% si la proposition de rejouer apparait

+40% diagrammes d'objets correct

-20% si appel du système à une opération « affiche proposition » de l'acteur Joe. L'envoi asynchrone d'un message, ou une note expliquant qu'on considère que Joe représente l'acteur et son IHM => -10%. Cela reste incorrect. On cherche les responsabilités du système, pas des acteurs (donc externes au système).

Question 2.5 : Réalisez un test de validation permettant de valider la fin de partie sur expiration d'une pendule.

Bon essentiellement, on démarre une partie avec temps = 1 minute, on attends une minute, on contrôle que c'est bien Noir qui gagne.

Exemple :

Test TV2 :

Contexte : Une partie à été démarrée avec une durée de pendule de 1 minute, suite à un défi, cf test de validation TV1, à exécuter avant celui-ci.

Entrée : aucune

Scénario : L'utilisateur attends une minute, sans toucher l'interface.

Résultat attendu : MyChess doit annoncer une victoire des Noirs.

Moyen de Vérification : utiliser un chronomètre externe à l'application.

Barême :

100% si test cohérent (scénario, contexte initial et donnée d'entrée) et essayant de faire planter l'application

50% si manque de cohérence mais l'objectif est correct

0% sinon