UPMC-LIP6

TD

4I502-IL : Ingénierie du Logiciel

Yann Thierry-Mieg

Master Informatique – 2016/2017

Supports en ligne : https://pages.lip6.fr/Yann.Thierry-Mieg/IL/

TD1

Considérons une application de commerce électronique permettant à des utilisateurs d'effectuer des achats de livres en ligne.
Cette application est composée de trois modules logiciels :

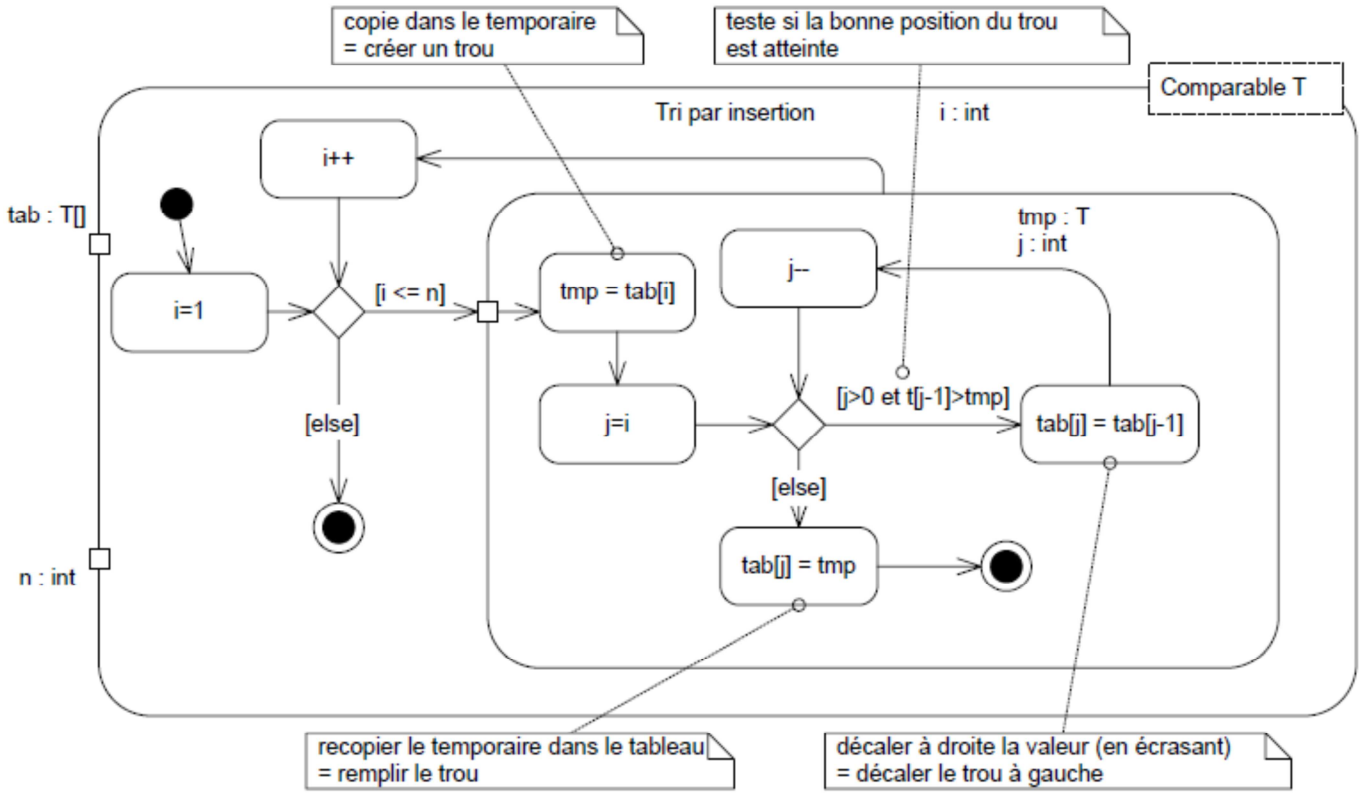
1. une base de données contenant toutes les informations sur les différents livres,
2. une interface graphique permettant aux utilisateurs d'effectuer leurs achats
3. et un gestionnaire permettant d'effectuer les virements bancaires.

Q1 - Décrivez cette architecture à l'aide d'un schéma. Vous expliquerez la notation graphique que vous utilisez (légende du schéma).
Q2 – A quoi peut servir ce schéma et quelles sont les qualités qu'il doit posséder ?

Cette application a été déployée par une société qui a décidé de mettre la base de données et le gestionnaire sur une même machine. Les interfaces graphiques s'exécutent dans les navigateurs web des clients. Les communications entre les machines s'effectuent via internet.

Q3 – Décrivez ce déploiement à l'aide d'un schéma. Vous expliquerez la notation graphique que vous utilisez (légende du schéma). A quoi peut servir ce schéma ?

Q4 – Le schéma d'architecture et le schéma de déploiement représentent deux points de vue d'une même application. Schématisez la relation qui lie ces deux schémas : on fera en sorte que les dépendances entre composants soient présentées ainsi que les connexions réseau entre les différentes entités supports au déploiement (machine). A quoi peut servir ce nouveau schéma ? A-t-on besoin des trois schémas (architecture, déploiement, lien) ou peut-on se contenter d'un seul ?
On souhaite modéliser un algorithme de tri.
Comparez le diagramme d'activité UML suivant et le code java correspondant.




```
public static <T extends Comparable<T>> void triSelect(T[] tab) {
      int n = tab.length;
      for (int i = 1; i <= n; i++) {
            T tmp = tab[i];
            int j;
            for (j = i; j > 0 && tab[j - 1].compareTo(tmp) > 0; j--) {
                  tab[j] = tab[j - 1];
            }
            tab[j] = tmp;
      }
}
```

Q5 – Quels sont les qualités et inconvénients de ces deux façons de présenter les choses ? Code et diagramme contiennent-ils la même information ? Quelle présentation préférer pour communiquer avec un collègue développeur Java ? Avec un informaticien qui ne parle pas Java ? Avec un mathématicien ? Avec un ordinateur ? Laquelle de ces deux représentations est la plus facile à construire ?

Q6 – Cet algorithme est lié au gestionnaire qui est déjà présent dans le schéma d'architecture. Schématisez cette relation entre l'algorithme et le gestionnaire. A quoi peut servir ce nouveau schéma ?

Q7 – Donnez une liste d'autres schéma qu'il serait intéressant de réaliser (expliquez leur intérêt).

Q8 – Expliquez les différences entre modèle et schéma. Un modèle a-t-il besoin d'une légende ? A quoi sert un modèle ? Un modèle doit-il être abstrait ou concret ? Qu'elles sont les qualités que doit posséder un modèle ?

Q9 – Expliquez la différence entre un modèle UML et un diagramme UML.

Q10 – Le code peut-il remplacer tous les schémas que vous venez de réaliser ? Expliquez les différences entre schéma et code. Expliquez les différences entre modèle et code.

Q11 – A quoi sert une méthode de conception logicielle ? Quelles doivent être les qualités d'une bonne méthode ?

TD2

Analyse

Notions abordées : Lire un cahier des charges, diagrammes de cas d'utilisation, diagrammes de classe métier.

Q1 : Rappelez l'objectif de la phase d'analyse.

Q2 : Identifiez les acteurs du SIGB.

Q3 : Un acteur est-il obligatoirement un utilisateur des fonctionnalités offertes par le système ?
Q4 : Identifiez les cas d'utilisation du SIGB, et élaborez le diagramme de cas d'utilisation du SIGB. En particulier, justifiez tout lien d'héritage entre acteur et toutes les relations (*include* et *extend*) entre cas d'utilisation.
Q7 : Identifiez les classes métier du SIGB.
Q8 : Elaborez le diagramme de classes métier du SIGB. Justifiez en particulier toutes les associations entre les classes.
Q9 : Echangez votre travail avec un autre groupe d'étudiants. Vérifiez que les réponses des autres étudiants ne contiennent pas d'erreurs. En particulier, prêtez attention à ce que, par rapport au cahier des charges, aucune information ne manque mais aussi à ce qu'aucune information ne soit en trop.

TD3

Spécifications détaillées

Notions abordées : Fiches détaillées, diagramme de séquence d'analyse, classe « système ».

Q1 : Réalisez la fiche détaillée des cas d'utilisation « enregistrer un retour » et « enregistrer un emprunt ».

Q2 : Pour chaque les cas d'utilisation « enregistrer retour » et « enregistrer emprunt » construisez les diagrammes de séquence d'analyse illustrant un fonctionnement nominal.

Q3 : Listez les évolutions nécessaires que vous avez faites aux classes de ces objets.
Q4 : Utilisez le temps restant pour appliquer la même démarche aux autres cas d'utilisation.

TD4

Notions abordées : Tests de validation, rupture Analyse/Conception du cycle de vie.

Q1. Rappelez l'objectif des tests de validation.

Q2 : A partir des diagrammes de séquence et des fiches détaillées élaborées, réalisez très précisément au moins deux tests de validation. Pour ce faire, vous remplirez le tableau suivant :

Titre :	**Id**
Contexte :	
Entrée :	
Scénario :	
Résultat attendu :	
Moyens de vérification :	

Q3 : Définissez un ordre dans lequel les tests de validation doivent être réalisés. Justifiez l'ordre que vous avez choisi. Pensez-vous que votre ensemble de tests est complet ?

Q4 : Echangez votre travail avec un autre groupe d'étudiants. Vérifiez que les réponses des autres étudiants ne contiennent pas d'erreur. En particulier, prêtez attention à ce que toute fonctionnalité ait des tests de validation lui correspondant.

Q5 : Expliquez les différences entre le point de vue conception et le point de vue de l'analyse. Quand peut-on parler de conception architecturale ?

L'objectif principal de cette première étape de conception est de proposer une découpe cohérente avec une séparation en composants.
On va commencer par réfléchir sur une découpe structurelle des données.

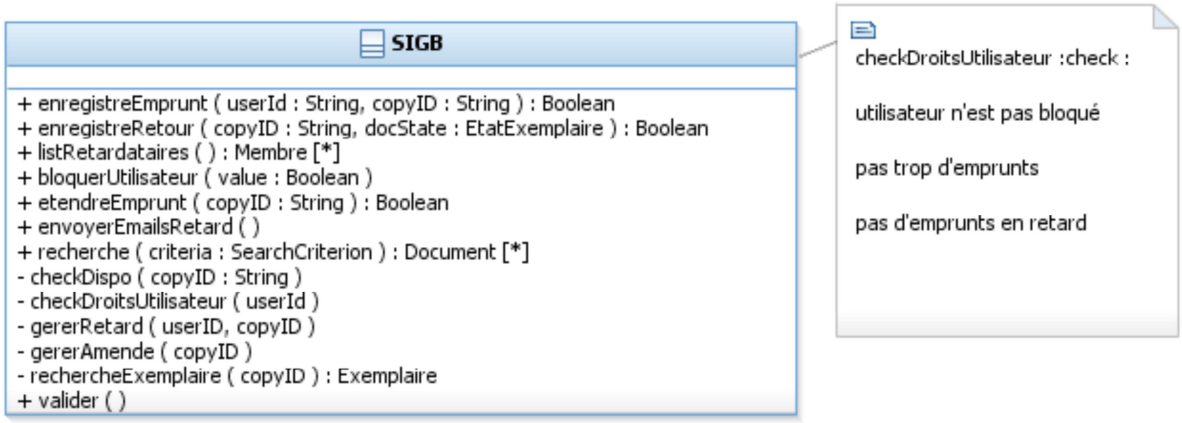
Q6. **Découpage orienté structure.**
En partant du diagramme de classes métier réalisé en analyse, identifiez une découpe possible sur des composants. Il faudra pour cela éliminer les dépendances structurelles, à l'aide d'identifiants. On peut être amené à séparer des classes et à ajouter des classes ou à modifier la répartition des données sur les classes. On proposera 3 ou 4 composants.

TD5

Notions abordées : Découpe fonctionnelle, interfaces, composants, configuration.

Découpage orienté fonctionnalités.

L'analyse a permis de dégager la liste (sans doute pas exhaustive) des responsabilités suivantes. Utilisez ce diagramme comme base, ou le travail réalisé aux séances précédentes. Les valeurs de retour Boolean sont utilisées pour noter que divers types d'erreurs peuvent se produire au cours de cette opération, empêchant de les mener à bien. En pratique, on utilisera des exceptions pour traiter ces cas en réalisation.



Q1. Proposez une découpe de ces responsabilités selon des critères logiques, et partitionnez-les sur des interfaces couvrant ces fonctionnalités. On pourra aussi essayer de mettre en valeur le découpage en fonctionnalités selon les acteurs.

Imaginons que chaque interface ainsi définie soit offerte par un composant dédié. On va travailler la cohérence entre les aspects structurel (TD 4) et fonctionnel, les interfaces définies devant correspondre avec les données manipulées. On affine donc les diagrammes élaborés pour

1. casser les dépendances structurelles entre composants en introduisant des identifiants.
2. S'assurer que chaque composant dispose bien des données nécessaires pour réaliser les traitements dont il est responsable
3. S'assurer que les types manipulés dans les opérations d'interface sont soit des types de base (String, int…) soit des interfaces. Introduire des interfaces au-dessus des classes concrètes au besoin.

Q2. A l'aide de diagrammes de séquence de niveau composant et en raisonnant sur les données nécessaires pour réaliser un traitement, identifier des dépendances entre composants. Essayez dans la mesure du possible d'orienter les dépendances identifiées pour éviter les cycles. Enrichissez vos interfaces existantes ou créez-en de nouvelles pour loger les opérations qui traduisent ces dépendances.
On cherche à se rapprocher de traitements réalisables par les composants.
Q3. Afin de faire apparaître l'IHM dans le modèle UML, nous choisissons une approche de conception qui consiste à construire un composant nommé « IHM ». Ce composant n'a pas d'interface offerte mais utilise les autres composants. Définissez ce composant.

Q4. Représentez à l'aide d'un diagramme de composants les composants proposés. On représentera aussi les interfaces explicitement avec leurs signatures. Quelles sont les qualités et

défauts principaux de l'architecture retenue ?

Q5 : Décrivez brièvement cette découpe en précisant le rôle des composants et leurs relations. Critiquez votre travail (forces et faiblesses).
Q6. Modélisez sur un diagramme de structure interne la configuration nominale des composants proposés.
Q7 : Expliquez en quoi le modèle de composants établi en conception générale est relativement indépendant des plateformes et langages de réalisation.

TD6

Notions abordées : Conception détaillée, réalisation de « composants » en Java (SE).

Q1. En partant de la conception architecturale réalisée, ajoutez à chaque composant une classe jouant le rôle de Façade et qui implémente l'interface offerte du composant.

Q2 : A l'aide des diagrammes de séquence de niveau composant, vérifiez que chaque composant a la capacité de traiter les opérations qu'il offre. Au besoin, enrichissez les interfaces offertes des composants.
Q3 : Les classes de conception du composant IHM représentent les écrans et les boutons de l'IHM. Proposez une approche pour les modéliser en UML

Q4 : Ajoutez une classe jouant le rôle de Factory pour les composants. On utilisera une forme simple portant des opérations *static*.

Q5 : Expliquez comment obtenir l'assemblage de composants décrit en Q6 du TD5 à l'aide de cette factory.

Q6 : Que manque-t-il aux modèles UML pour être pleinement exécutables ? Pourquoi n'est-ce pas souhaitable de chercher à décrire ce niveau de détail directement en UML ?

Q7 : Quels sont les avantages et inconvénients d'un modèle décrit au niveau d'abstraction « code » ?

Q8 : Pensez-vous avoir fini la phase de conception ?

TD7

Conception détaillée, Tests d'intégration, Tests Unitaires

Q1. On va se pencher à présent sur la conception détaillée du moteur de recherche de documents.
On supposera que l'on part d'une chaine de texte décrivant la requête utilisateur, par exemple : author= « E. Gamma » and year=1995
On commencera par proposer une représentation de l'arbre de syntaxe abstraite de la requête, à l'aide du DP Composite.
On décrira ensuite une façon de réaliser cette recherche, en supposant

a) Une implémentation OO
b) Une implémentation sur un moteur de base de données relationnel.

Tests d'intégration

Q2. Rappelez la nature et l'objectif des tests d'intégration. Qu'est-ce qu'un composant de test et un composant bouchon ?

Q3 : Rappelez en quoi les diagrammes de séquence inter-composant (cf. TD 5) peuvent être utilisés lors de la rédaction des tests d'intégration.

Q4. Définissez les composants bouchon et testeur nécessaires pour tester le composant gérant les abonnés, puis le composant gérant les emprunts.

Q5 : A l'aide de diagrammes de structure interne définissez des configurations des composants pour ces tests.
Expliquez comment créer ces assemblages de composants à l'aide de la Factory.
Q5 : Définissez le jeu de données qui sera utilisé lors de l'exécution des tests d'intégration.
Rappelez en quoi votre jeu de données ne peut être complet et ne peut être que satisfaisant.

Q6 : Définissez les tests d'intégration de l'application SIGB.

Tests Unitaires

Q7 : Rappelez l'objectif des tests unitaires et expliquez la différence entre test unitaire et test d'intégration.

Q8 : Quelles métriques utiliser pour mesurer la qualité d'un jeu de tests unitaires.
Q9 : Proposez une extension à UML afin de pouvoir spécifier des tests.

TD8

Adapter, Façade

Réutilisation de l'existant

L'analyse d'un système de réservations de salles a produit le diagramme de cas d'utilisation suivant :

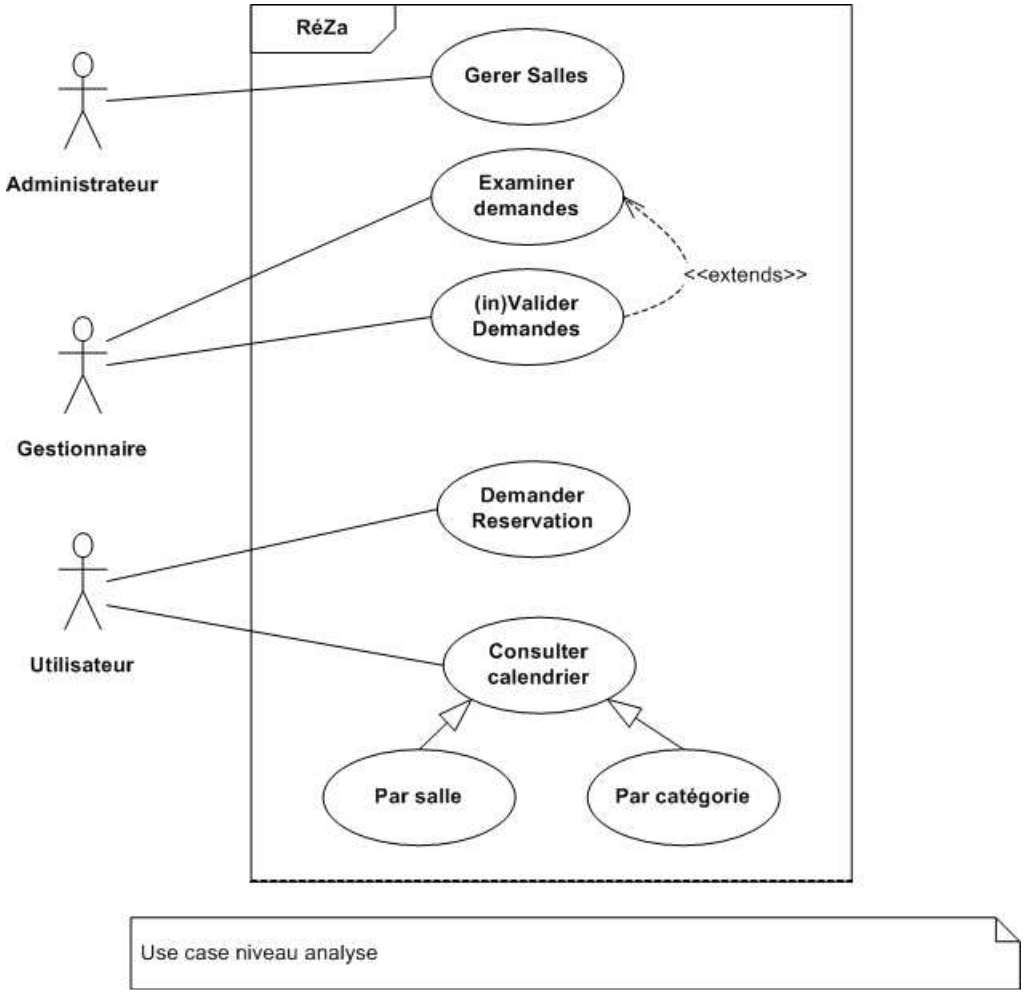


Fig. 1 : Use cases

Ces cas d'utilisation sont détaillés ici :

Acteur : Administrateur

- L'administrateur gère la liste des salles disponibles. On ne le détaillera pas dans la suite.

Acteur : Utilisateur

- Consulte les emplois du temps et fait des demandes de réservation

Demander Réservation :

Il existe deux types de réservations : les réservations ponctuelles et les réservations périodiques. Pout toute réservation le demandeur doit préciser son nom (par exemple « Y.Thierry-Mieg »), la catégorie à laquelle la réservation est liée (par exemple « ue4I502 »), un nom pour cette réservation (par exemple « Examen rattrapage »), la salle demandée (par exemple « Amphi B1 »), et la date et plage horaire désirée (par exemple « 22 Janvier, 15h30-17h30 »).

Si la réservation est périodique, l'utilisateur le précisera sur le même formulaire, et fixera la périodicité (quotidienne, hebdomadaire ou mensuelle) et le nombre d'occurrences (par exemple

pour « Cours 4I502 », 10 occurrences de périodicité hebdomadaire).
Spécifications complémentaires :

- Une demande périodique de n occurrences sera traitée comme si l'utilisateur avait fait n demandes ponctuelles du point de vue de la validation (chaque occurrence pouvant être individuellement validée ou non).

Consulter Calendrier

La consultation des emplois du temps permet aux utilisateurs d'obtenir un calendrier qui correspond à leurs critères de recherche. Il existe deux types de calendrier :

- Par catégorie : chaque réservation contient un champ « catégorie », qui permet de préciser à quel évènement cette réservation se rapporte. Par exemple « ue 4I502 ». La recherche par catégorie permet d'obtenir un emploi du temps couvrant toutes les réservations qui se rapportent à la catégorie.
- Par salle: il est possible d'obtenir l'emploi du temps d'une salle particulière, sur une période donnée. L'utilisateur n'a qu'à préciser la salle et les dates de début et de fin du calendrier à produire.

Acteur : Gestionnaire

- Chargé de valider les demandes de salles

Examiner demandes et (In)Valider demande

Le gestionnaire peut consulter toutes les demandes de réservation non validées. Il les inspecte alors une par une et décide de les valider ou les invalider.
Spécifications complémentaires :

- Dans les deux cas, le demandeur de la réservation sera informé par mail de la mise à jour de l'état de sa réservation.

Q1. L'équipe en conception a défini le composant Reservations de la façon suivante (fig.2). Quel est l'intérêt d'un tel composant ? Comment le réaliser en pratique ? Quel design pattern est utilisé pour accéder à l'unique instance de Reservations ?

Q2. L'équipe a décidé de découper les responsabilités de la classe Système d'analyse sur des interfaces distinctes pour chaque use case principal (fig. 3). On suppose que ces interfaces sont offertes par un seul composant qui joue le rôle de contrôleur, représentez cette situation sur un diagramme de composant.

Q3. Interactions pour la gestion des réservations.
Modélisez à l'aide d'un ou plusieurs diagrammes de séquence de niveau intégration, c'est-à-dire où les lignes de vie représentent des composants, les interactions nécessaires pour demander une réservation ponctuelle de la salle B1 le 22 janvier de 15h à 17h, puis que le gestionnaire valide cette demande.

Q4. Le contrôleur s'appuie sur le composant Calendrier, trouvé sur étagère, permettant la génération d'images représentant des calendriers. L'API de ce composant est la suivante (fig.4). Quel pattern de création reconnaissez-vous ? Mettez à jour le diagramme de composants de la question 2 pour représenter que le contrôleur s'appuie sur ce composant.

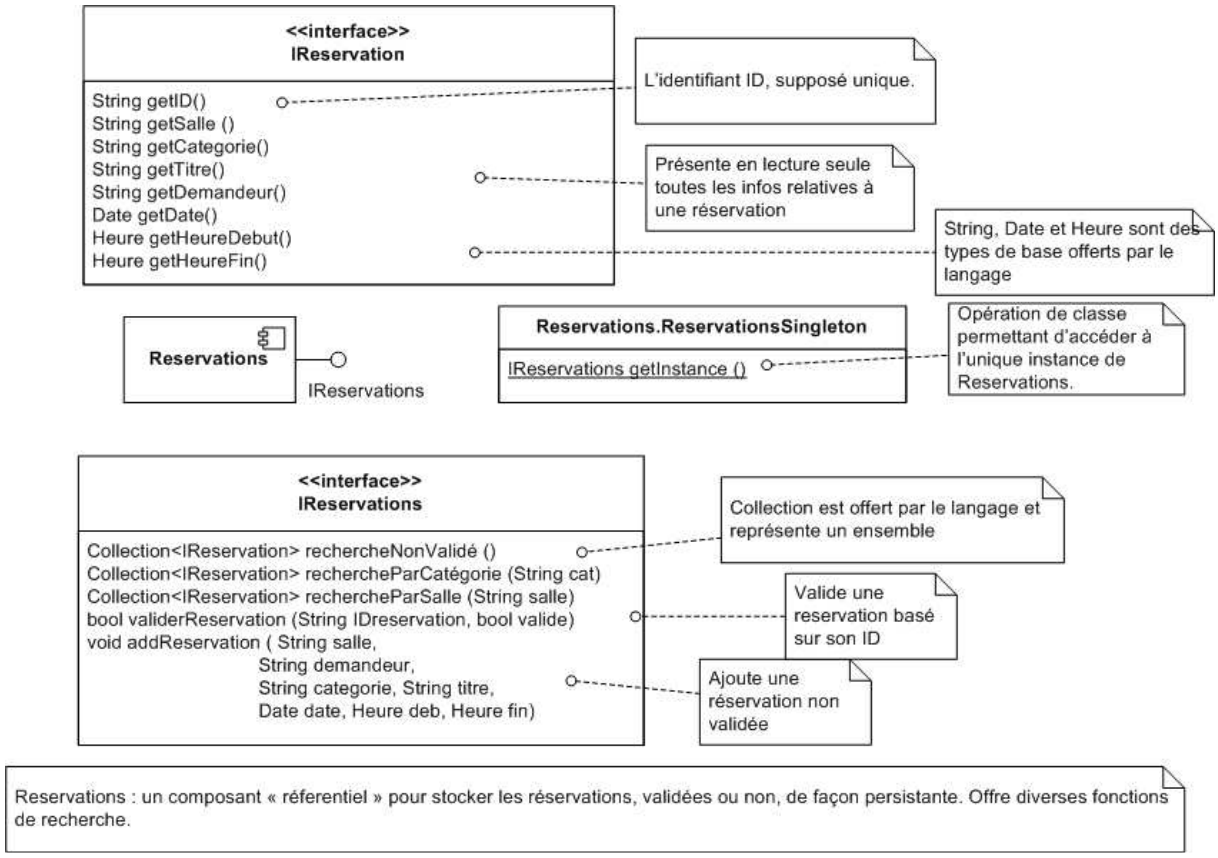


Fig.2 : Composant Reservation

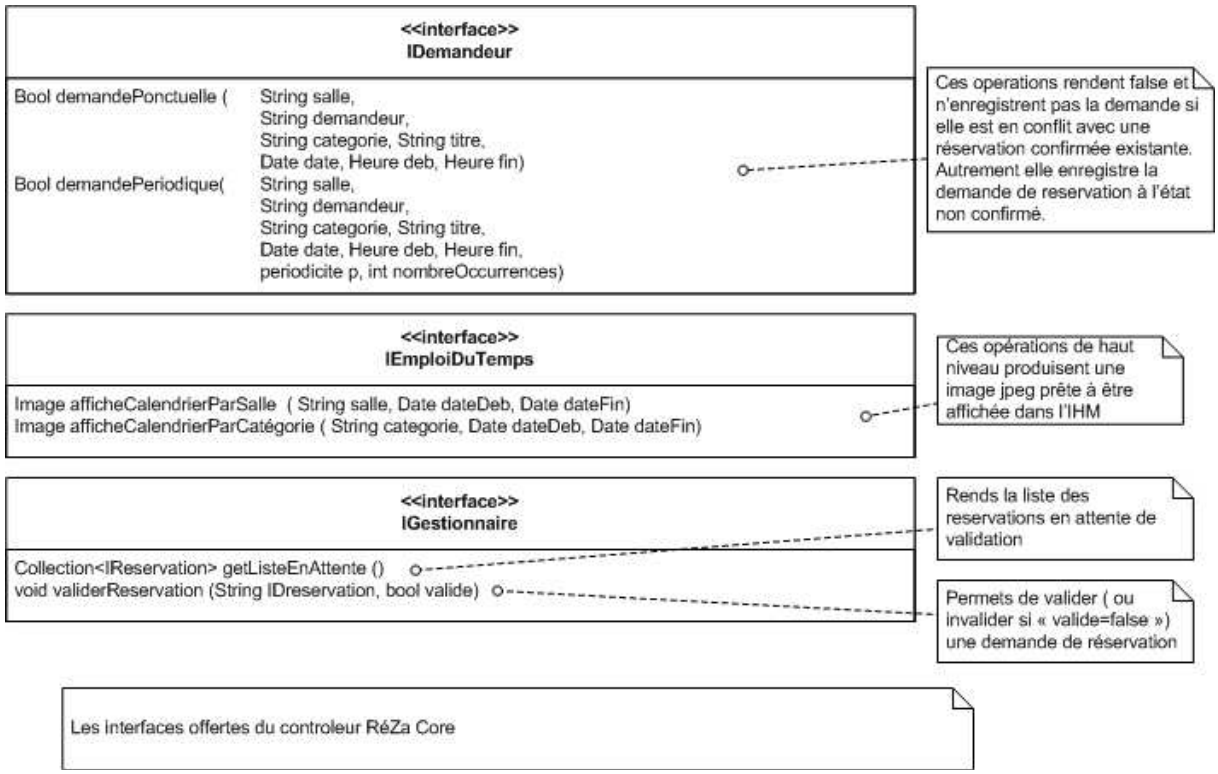
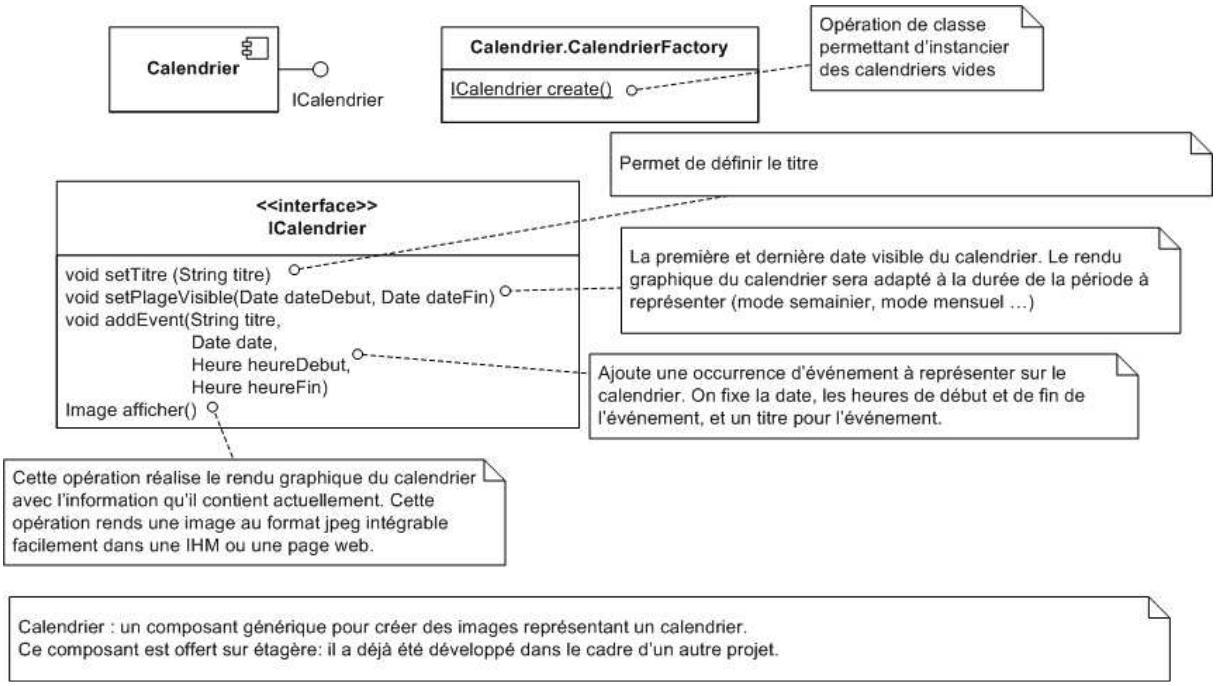


Fig.3 : Les interfaces



Q5. Modélisez à l'aide d'un ou plusieurs diagrammes de séquence de niveau intégration, c'est-à-dire où les lignes de vie représentent des composants, les interactions nécessaires pour qu'un utilisateur consulte un calendrier portant sur la salle B22 du 1er au 15 janvier.
Q6. Comment réaliser le composant contrôleur en pratique ? Quel(s) design pattern(s) appliquer ?

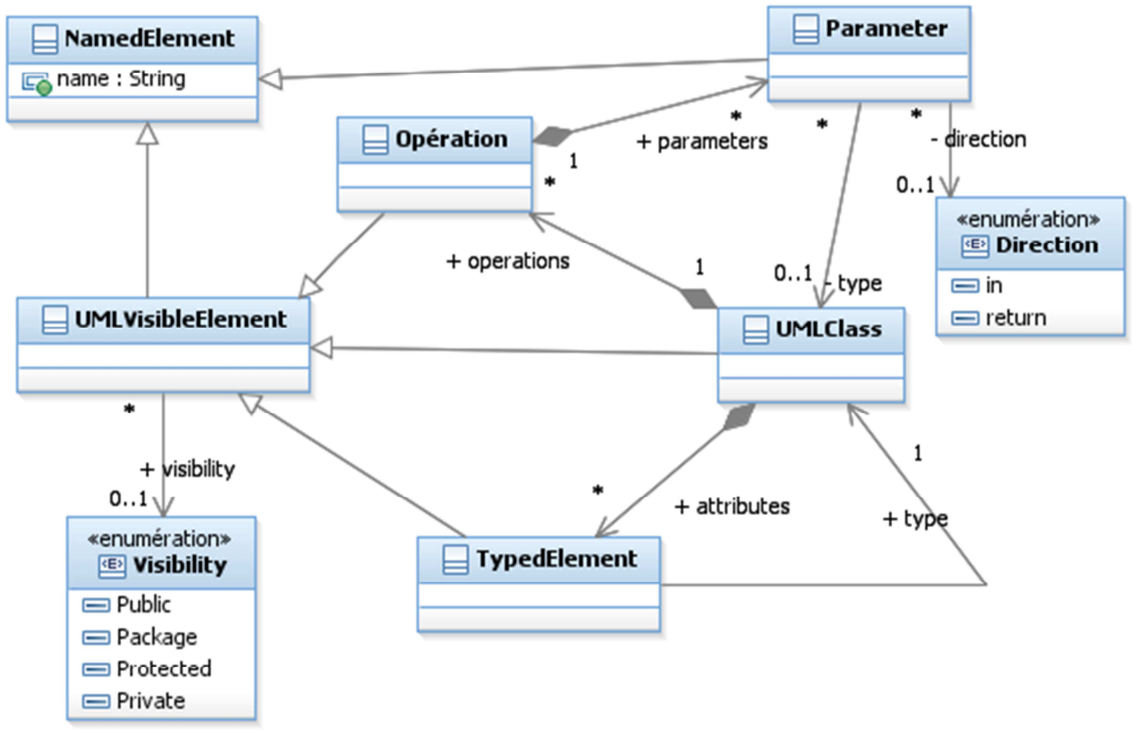
TD9

Cycles de développement
Opérations de production, Méta-modélisation

Q1 : Rappelez, pour chaque étape du cycle en V, les diagrammes UML utilisés. Expliquez le rôle de chacun de ces diagrammes et les différentes opérations de production qu'il est possible de réaliser (génération de code ou autre).
Q2 : Définissez les étapes du processus "Extreme Programming".
Q3 : Proposez une exploitation d'UML pour l'extreme programming XP

Q4 : Rappelez les différents types d'opérations de production sur les modèles et précisez l'intérêt de ces opérations. Donnez pour chaque type un exemple concret.

Q5 :
Soit l'extrait simplifié du méta-modèle des diagrammes de classe d'UML :



Décrivez les objets montrant comment ce méta-modèle s'instancie pour modéliser une classe Chainon, munie d'un attribut « next » de type Chainon, et d'une opération addNext(in c : Chainon) . On utilisera une représentation arborescente, proche de la structure du fichier XMI.
Q6. Nous allons travailler sur la génération de code qui est une opération de production "Modèle vers Texte". Reprécisez l'algorithme de génération de code pour les classes avec leurs attributs (sans prendre en compte l'héritage, la réalisation d'interface et les opérations). On s'intéressera particulièrement à la conversion des types des attributs.
Q7. Cette transformation est-elle facilement réversible ? Quelles opérations faut-il mettre en place pour le faire ?

M1 : Ingénierie du Logiciel UNIVERSITE PIERRE & MARIE CURIE

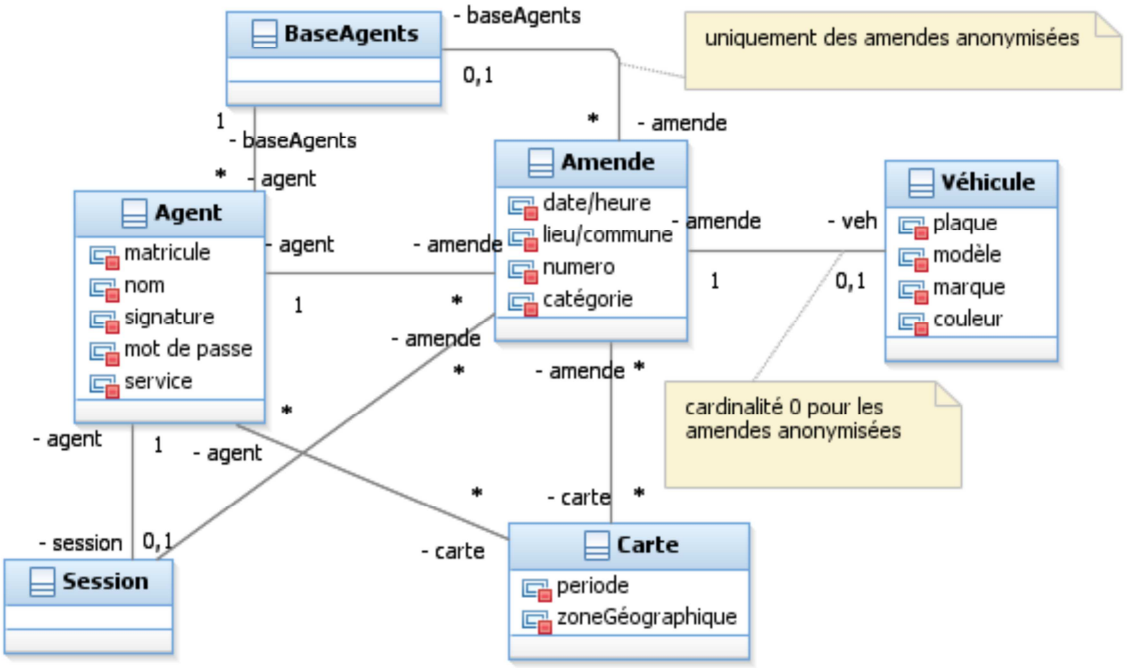
Examen Réparti 2eme partie 16 Décembre 2010

2 heures avec documents -- Barème indicatif sur 20 points.

Problème: Conception de @mende [20 Pts]

Extrait du cahier des charges (rappel): De retour au poste de police, l'agent va clore sa session et noter la fin de sa tournée. Il branche le smart-phone sur un terminal, qui récupère les données de la session et les envoie au fichier central des amendes. Ce transfert n'utilisera pas une connexion sans fil en raison des problèmes de sécurité que cela poserait, au vu de la sensibilité forte des données. La fin du téléchargement des données ferme la session de l'agent. Les données des amendes sont transférées au système national de traitement des amendes **(SNTA)**. Si le serveur national est momentanément indisponible, les données seront stockées sur le terminal par l'application, qui retentera un transfert plus tard à intervalles réguliers. En plus, une copie anonymisée (les matricules d'agent sont conservés mais les plaques d'immatriculation des véhicules sont retirées) est stockée dans une base de données locale.

L'analyse de @mende a permis de construire ce diagramme de classes métier :



Partie I (4 points)

On propose dans un premier temps de construire un composant bout de chaîne pour représenter un ensemble d'amendes (anonymes ou non). On se limitera à la gestion des caractéristiques suivantes de l'amende, gérées à l'aide de types simples (String, Integer, Date…) : numéro de l'amende, catégorie, lieu, date, marque du véhicule, identifiant de l'agent, service de l'agent, et pour les amendes qui ne sont pas anonymes la plaque minéralogique du véhicule.

Question 1 (4 points): Proposez une conception de ce composant permettant la manipulation d'un ensemble d'amendes. On se limitera aux opérations de création et de lecture.

a) On commencera par représenter sur un diagramme de composant son/ses interfaces offertes et requises (avec opérations et signature dans les interfaces).

b) On réalisera ensuite un diagramme de classe expliquant une conception détaillée d'une réalisation possible du composant.

NB : Dans la suite on appellera IListAmendes l'interface offerte de ce composant permettant la manipulation d'un ensemble d'amendes. Sur les diagrammes de composant qui suivent, il n'est pas demandé de représenter la dépendance sur IListAmendes des divers composants.

Partie II : **(7 points)**

On s'intéresse à présent à la gestion de la fin de session de l'agent. L'analyse détaillée du cas d'utilisation « Clore Session Agent » a identifié les scenarii de comportement suivant :

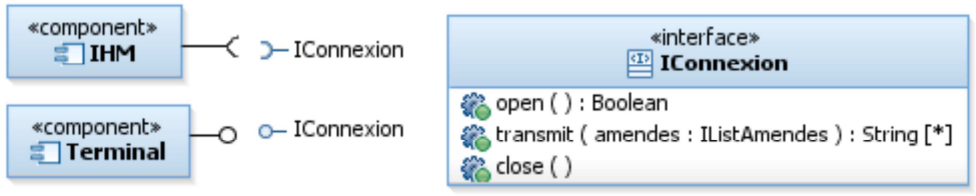
1. L'agent sélectionne l'action « Clore Session » **(offert par IHM)**
2. Le système **(IHM)** contrôle la connexion au Terminal.
3. Le système **(TERMINAL)** transmet les amendes de la session en cours au SNTA.
4. Le système **(TERMINAL)** stocke les amendes anonymisées dans une base locale.
5. Le système **(IHM)** affiche un message de confirmation.

Exception E1 :
En étape 2, si la connexion est indisponible, le système **(IHM)** affiche un message invitant l'agent à vérifier sa connexion physique (filaire) au terminal. Le cas d'utilisation est interrompu.
Alternative A1 :
En étape 5, si l'envoi de certaines données au SNTA a échoué à l'étape 3, le système **(IHM)** affiche un message indiquant la nature de l'erreur. Ces Amendes seront stockées **(TERMINAL)** sur le terminal qui retentera un transfert **(TERMINAL)** plus tard à intervalles réguliers. La session de l'agent est tout de même fermée **(IHM)**.

En débutant la conception, on a ajouté à cette description un raffinement pour distinguer les composants de l'application, en particulier le terminal et l'IHM de l'agent, notés en **MAJUSCULE**.

Question 2 (3 points) : (NB : les Questions 2 et 3 sont liées, il est recommandé de lire les deux avant de répondre)

On propose de s'appuyer sur l'interface suivante pour représenter la connexion de l'IHM au terminal. Interface IConnexion :



open() : boolean : démarre la connexion au terminal, on gère l'existence d'un problème avec un booléen pour éviter de modéliser des exceptions.

transmit (amendes : IListAmendes) : String[*] : transmet les amendes de la liste fournie, et rend un rapport d'erreurs potentielles de transmission, sous la forme d'une collection de String, une par amende n'ayant pas pu être transmise. Si cette liste est vide, c'est qu'il ne s'est pas produit d'erreurs.

close() : void : ferme la connexion.

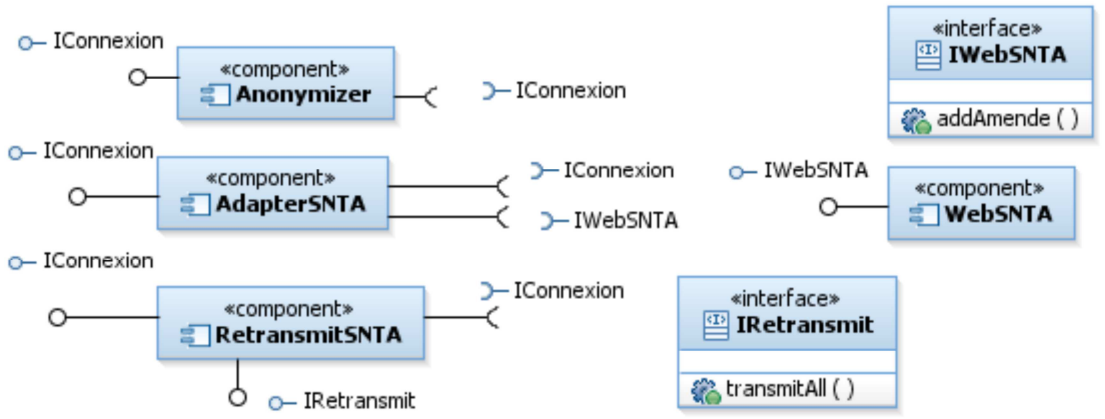
a) Quel est l'intérêt, pour un composant donné, de réaliser une classe bouchon qui implémente les différentes interfaces qu'il requiert ?

b) Comment faut-il spécifier cette classe dans le cas présent, pour permettre de substituer le bouchon au terminal réel ? Faites un diagramme de classe pour décrire la classe bouchon.

c) Modélisez l'assemblage de l'IHM et de ce composant bouchon dans une configuration de test à l'aide d'un diagramme de structure interne.

Question 3 (4 points)
Représentez par un ou plusieurs diagrammes de séquence les interactions entre l'IHM de l'agent et le Terminal. On couvrira tous les cas de figure identifiés par l'analyse.
On se place au niveau intégration, en considérant les composants Terminal et IHM uniquement. On se focalise dans cette question sur les responsabilités de l'IHM dans ces scenarii, on ne représentera donc pas les actions du terminal qui seront traitées dans les questions suivantes. Par contre on représentera les affichages réalisés par l'IHM. On considérera que l'IHM a déjà préalablement construit une IListAmendes représentant les données de la session.

Partie III (9 points)
On s'intéresse à présent à la conception du terminal. Les responsabilités du terminal sont identifiées dans la description du cas d'utilisation de la question 2.
On propose le découpage suivant, détaillé dans la suite (**NB : il est fortement recommandé de lire les questions 4 à 6 avant de commencer à répondre)**:



Un composant **Anonymizer** réalise le stockage dans la base locale des amendes anonymes, un composant **AdapterSNTA** réalise la transmission au SNTA ou en cas de souci de connexion au composant **RetransmitSNTA** de retransmission périodique. Ces trois composants vont être chaînés ; ils offrent et requièrent chacun l'interface IConnexion. Le composant **Anonymizer** sera connecté à **l'IHM** dans le déploiement final ; il constitue le point d'entrée du Terminal.
Question 4 (2 points) : Anonymizer
Ce composant n'est pas complètement décrit dans le schéma ci-dessus.
Il doit fonctionner de façon transparente, il retransmet directement les amendes reçues sur son interface offerte sur son interface de sortie.
Cependant au passage il stocke une version anonyme des amendes dans la base de données locale, gérée par un composant LocalBD muni d'une interface qui lui est propre.

a) Complétez la description du composant Anonymizer proposée ci-dessus en intégrant cette contrainte (sur un diagramme de composant).

b) Quelles opérations (+signatures) devrait offrir LocalBD ?

Question 5 (4 points) : AdapterSNTA
Le système SNTA (acteur secondaire du système) offre une interface de manipulation via un webservice présenté sur une couche de connexion sécurisée (https). Pour nos besoins, l'équipe de développement web fournit le composant WebSNTA qui gère la connexion au webservice et l'envoi des données.

On donne la signature de l'unique opération de IWebSNTA :
addAmende (numero : int, lieu : string, date : string, matricule agent : string, service : string, plaque : string, marque : string, categoriePV : int) throws ConnexionException.
Le composant AdapterSNTA réalise IConnexion. Quand on (dans le système final, ce sera une occurrence du composant Anonymizer) invoque son opération ***transmit(liste),*** **AdapterSNTA** doit essayer d'envoyer les amendes contenues dans ***list*** au serveur web. S'il n'y parvient pas, il doit transmettre les amendes *qu'il n'a pas réussi à envoyer* à sa connexion sortante (IConnexion requise). Cette interface requise sera connectée au composant de retransmission dans l'application finale. De plus pour chaque amende qu'il n'a pas pu transmettre au SNTA, une String décrivant l'erreur (identifiant de l'amende et le message associé à la ConnexionException reçue) sera ajoutée dans la valeur de retour de ***transmit***.
Modélisez ce comportement à l'aide d'un diagramme de séquence représentant le comportement d'une instance de **AdapterSNTA** quand on invoque « transmit » de IConnexion et que certains envois échouent. On pourra user de syntaxe libre (notes de commentaire, pseudo-code) pour expliquer les points algorithmiques plus difficiles à modéliser (boucles, conditionnelles, exceptions…).

Question 6 (3 points) : RetransmitSNTA

Ce composant crypte les amendes qui lui sont passées et assure leur persistance. Quand on invoque « transmitAll » sur son interface IRetransmit, il recharge les amendes cryptées et les transmets sur sa connexion sortante. « transmitAll » est invoqué à intervalles réguliers (toutes les heures) par une tache de fond (de type cron).
Ce composant reçoit et transmet ses amendes à une même instance de AdapterSNTA dans l'architecture finale.
Représentez sur un diagramme de structure interne l'assemblage final du système. On relira attentivement les descriptions des divers composants dans les questions précédentes.
On devra trouver une instance de chacun des composants IHM, Anonymizer, LocalDB, AdapterSNTA, WebSNTA, RetransmitSNTA, et une instance d'un composant Cron modélisant la tache périodique. On fera attention à l'orientation des liens entre instances.

Système Intégré de Gestion de Bibliothèque

Le SIGB répertorie l'ensemble des documents constituant le fonds de la bibliothèque (plus de 50 000 documents). Ce fonds est constitué actuellement de livres et de périodiques. Il est prévu que le fonds contienne dans l'avenir d'autres catégories de documents.
Chaque document a un titre, une année de publication, un éditeur, un court texte de description et une unique référence. Un livre a un (des) auteur (s), et un code ISBN (International Standard Book Number). Un périodique a un volume, un numéro ainsi qu'un code ISSN (International Standard Serial Number).

Chaque exemplaire d'un document a un identificateur unique constitué d'un numéro d'ordre ainsi que de la date d'achat, et une référence donnant sa position (étagère) dans la bibliothèque. Le nombre d'exemplaires de chaque document est stocké dans le catalogue. Un exemplaire d'un document présent dans la bibliothèque est dit « en rayon ». Seuls les exemplaires « en rayon » peuvent être prêtés. Quand un utilisateur emprunte un exemplaire d'un document, l'exemplaire est dit « en prêt ». Un exemplaire qui n'a pas été rendu dans les délais par un utilisateur est dit « en retard ». Un exemplaire d'un document présent dans la bibliothèque est dit « en réserve » lorsqu'il ne peut pas être prêté. Chaque document a au moins un exemplaire « en réserve ». Un exemplaire d'un document qui est temporairement hors de la bibliothèque pour des travaux de restauration est dit « en travaux » ; dès que les travaux sont terminés, l'exemplaire est remis dans la bibliothèque.
Chaque exemplaire d'un document dispose d'un état. Les valeurs possibles sont au nombre de 5: *neuf, très bon état, bon état, usagé, endommagé*. Si, au retour, il s'avère qu'un document baisse de plus de 3 niveaux, une amende forfaitaire sera exigée. On dispose d'une grille qui permet de calculer l'indemnité à régler en fonction de la valeur du document. Les documents dans l'état endommagé sont notifiés pour être envoyés en travaux ou être détruits.
Le système enregistre toutes informations relatives aux utilisateurs de la bibliothèque dans une base de données. Il distingue trois catégories de clients : les *utilisateurs occasionnels* qui ont le droit d'emprunter un seul document à la fois pour une durée de 15 jours, les *abonnés* qui ont le droit d'emprunter en même temps 4 documents au plus pendant un mois, les *abonnés privilégiés* qui ont le droit d'emprunter en même temps 8 exemplaires de documents au plus pendant un mois.

Tout emprunt est enregistré dans le système. Chaque emprunt a une durée limitée définie par une date de début et une date de fin. Si un utilisateur ne rend pas dans les délais un exemplaire de document emprunté, un email lui rappelant de ramener l'exemplaire est généré. Deux autres mails sont générés à une semaine d'intervalle si le document n'est toujours pas rendu. Un utilisateur ne peut plus emprunter de documents tant qu'il n'a pas rendu les exemplaires conservés au-delà des délais de prêt. Le bibliothécaire peut visualiser la liste des utilisateurs en retard de plus de trois semaines afin de prendre les mesures adéquates (courrier postal, convocation…).

Les bibliothécaires forment plusieurs catégories en fonction de leur position hiérarchique. Les *stagiaires* ne sont présents que pour une faible durée. Ils viennent aider les *bibliothécaires principaux* pendant les périodes d'affluence (avant et au retour des vacances scolaires) et ne peuvent qu'enregistrer des emprunts ou des retours. Les *bibliothécaires principaux* peuvent en plus prolonger un emprunt, interdire temporairement à un utilisateur d'emprunter des documents et consulter la liste des utilisateurs en retard.
Le système permet également à la fois aux usagers de la bibliothèque (via un poste en libre accès) et aux bibliothécaires de rechercher un exemplaire (selon des critères multiples: titre, auteur…) et de connaitre sa position et sa disponibilité.