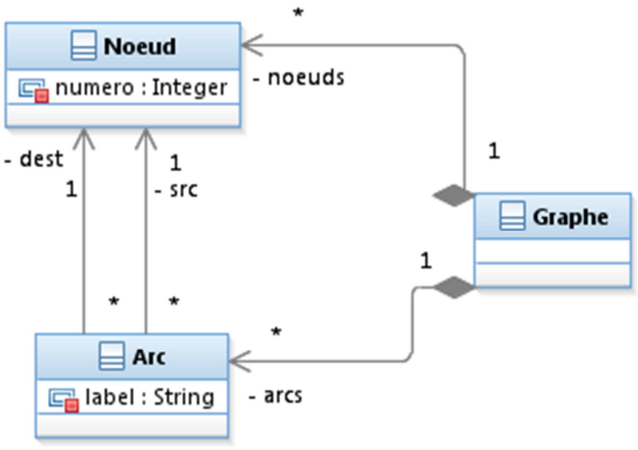
M1 : Ingénierie du Logiciel - UNIVERSITE PIERRE & MARIE CURIE (PARIS VI)

Examen Réparti 2eme partie

9 Janvier 2015 (2 heures avec documents : tous SAUF ANNALES CORRIGEES).
Barème indicatif sur 21 points (donne le poids relatif des questions) (max 20/20).

Questions de cours [5 Pts]

Répondez de façon précise et concise aux questions.
Soit le méta-modèle suivant permettant de représenter des instances de graphes composés de nœuds portant un numéro et d'arcs orientés d'un nœud source vers un nœud destination et portant une étiquette qui est un string.



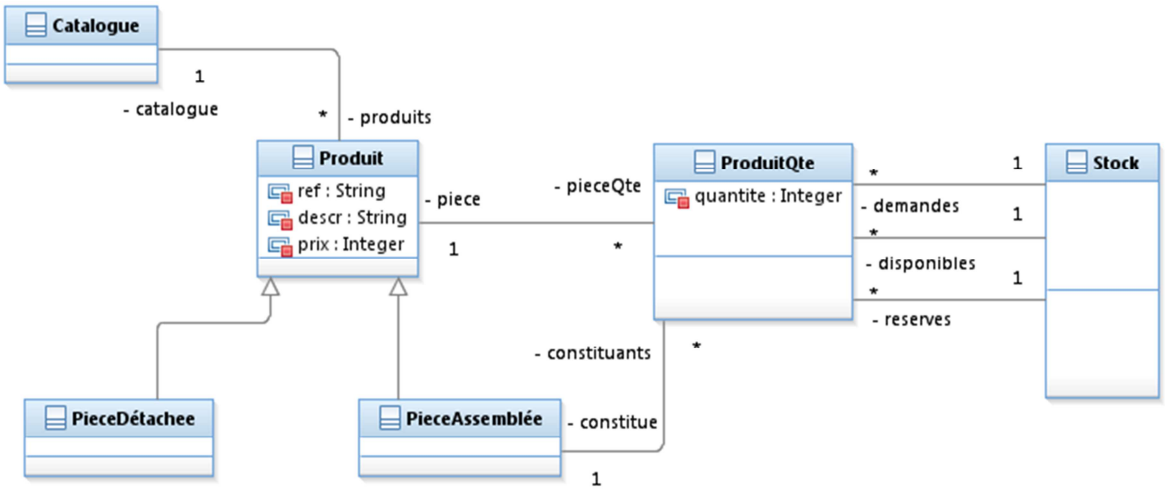
Question Cours (QC):

a) Représentez sous une forme textuelle arborescente (proche de la structure d'un fichier XMI) les objets d'un modèle instance représentant un graphe constitué de deux nœuds (indicés 0 et 1) et de deux arcs de 0 à 1 (étiquette « a ») et de 1 à 0 (étiquette « b »).
b) Avec ce méta-modèle peut-on construire un modèle instance où plusieurs arcs portant la même étiquette « a » lient le nœud 0 au nœud 1 ? Justifiez pourquoi.
c) Proposez une modification de ce méta-modèle permettant de représenter qu'un nœud du graphe est le nœud initial.
d) Proposez une modification de ce méta-modèle permettant de représenter qu'un modèle peut contenir un ensemble de graphes et pas un seul.
e) Les arcs pondérés sont des arcs qui portent en plus de leur étiquette un poids représenté par un entier. Modifiez le méta-modèle pour permettre de gérer ce cas.

2. Problème: Gestion de production [Barème sur 16 Pts]

Rappels d'analyse : Nous considérons une application permettant la gestion de stocks et de production d'une usine de cycles. Un produit est muni d'une référence unique, d'une description et d'un prix. Certains produits sont des pièces détachées, achetées par l'usine auprès de fournisseurs. D'autres produits sont des pièces assemblées dans l'usine, en utilisant d'autres produits comme constituants.

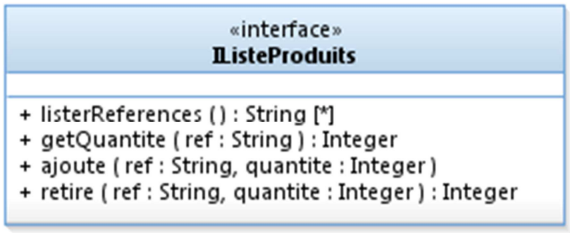
On rappelle ces éléments (simplifiés) du diagramme de classe d'analyse.



On s'intéresse ici à la gestion du stock, qui va s'appuyer sur trois composants, gérant respectivement un ensemble de produits associés à des quantités (un composant utilitaire simple, largement réutilisé), un catalogue (avec les descriptions détaillées des produits et de leurs constituants éventuels), et un composant gérant la mise à jour de l'état des stocks suite aux achats de pièces détachées auprès des fournisseurs, aux opérations d'assemblage réalisées dans l'usine et à l'évolution des commandes.

I. Liste de produits (5 points)

Soit l'interface suivante permettant d'associer des références de produits à des quantités. Par exemple, une liste peut contenir 50 occurrences du produit de référence « rayon », et 1 occurrence de « jante ».



- `listerReferences` : rend la liste des références associées à une quantité non nulle
- `getQuantite` : rend la quantité associée à une référence, 0 si la référence est inconnue
- `ajoute` : ajoute *quantité* occurrences de la référence à la liste
- `retire` : essaie de retirer *quantité* occurrences de la référence de la liste. Rend le nombre de références effectivement obtenues (plus petit ou égal à *quantité*), la quantité dans la liste ne peut pas descendre en négatif.

Question I.1 : (2 pts)

a) Représentez sur un diagramme de composant un composant CListeProduit qui réalise cette interface.
b) Proposez à l'aide d'un diagramme de classes une conception détaillée possible de ce composant.

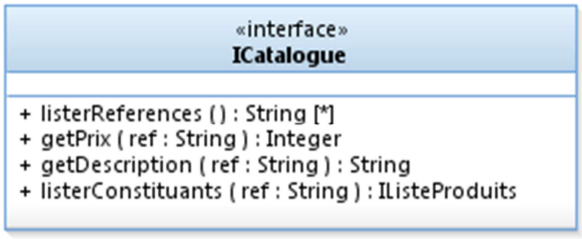
Question I.2 : (3 pts)

On souhaite mettre en place des tests d'intégration de ce composant validant (entre autres) le scénario suivant : partant d'une liste vide, on ajoute 5 vélos, on en retire 3, la quantité de vélos dans la liste doit être de 2.

a) Représentez un ou plusieurs composants (testeur, bouchon(s)) permettant de mettre en place ces tests.
b) Sur un diagramme de structure interne, représentez une configuration des composants pour le test.
c) Rédigez (avec une syntaxe exécutable en pseudocode style JUnit) un test d'intégration couvrant le scénario précité.

II. Catalogue (6 points)

Le composant **CCatalogue** stocke la description des produits. Il réalise l'interface suivante :



- `listerReferences` : donne la liste des références de produits du catalogue
- `getPrix` : rend le prix d'un produit particulier du catalogue (0 si la référence est inconnue)
- `getDescription` : rend la description associée à un produit du catalogue (**null** si la référence est inconnue)
- `listerConstituants` : rend les constituants d'un produit du catalogue (**null** si la référence est inconnue ou si la référence correspond à une pièce détachée)

Question II : (6 pts)

a) Représentez **CCatalogue** sur un diagramme de composant.
b) Proposez à l'aide d'un diagramme de classes une conception détaillée possible de ce composant.
c) Un composant **CCatalogueIHM** permet d'afficher l'information contenue dans le catalogue. Représentez-le sur un diagramme de composant.
d) (2 pts) Faites un diagramme de séquence de niveau intégration (où les lignes de vies sont des instances de composants) qui permette à une instance de CCatalogueIHM d'afficher (en cohérence avec l'exemple donné en III):

> Une « roue » est constituée de 50 « rayon » et de 1 « jante ». « jante » est une pièce détachée. « rayon » est une pièce détachée.

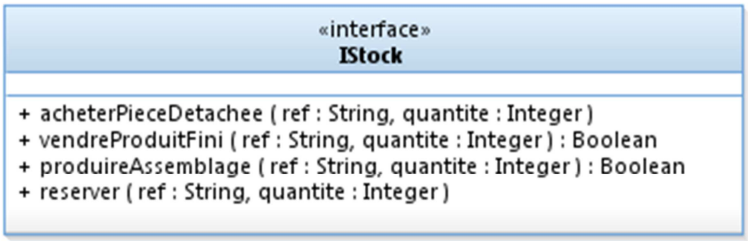
On ne demande pas que cette IHM affiche les descriptions ou les prix des produits. Cependant l'IHM doit tenter de récursivement afficher les constituants. On prendra soin de spécifier arguments et valeurs de retours sur les messages.

e) Représentez sur un diagramme de structure interne les instances de composants qui participent à l'interaction décrite en d).

III. Stock (5 points)

Le composant **CStock** gère et maintient à jour trois listes de produits :

- Les produits **disponibles** sont immédiatement en stock et n'ont pas été réservés.
- Les produits **réservés** sont présents dans le stock, mais font l'objet d'une réservation pour honorer une commande
- Les produits **demandés** ne sont pas présents dans le stock, il faudra les acheter auprès d'un fournisseur (produit de type pièce détachée) ou les produire dans l'usine (produit de type pièce assemblée).



Il offre l'interface suivante :

- `acheterPieceDetachee` : permet d'ajouter *quantité* occurrences de la pièce détachée portant la référence *ref* au stock. Cette opération est invoquée quand le fournisseur livre des pièces à l'usine. Si le stock contient 70 **demandes** pour *ref*, et si on en achète 100 unités, le stock contiendra 70 produits **réservés** et 30 produits **disponibles**.
- `vendreProduitFini` : permet d'enregistrer la sortie d'usine (et donc du stock) de *quantité* occurrences d'un produit assemblé fini, pour honorer les commandes. Les produits à vendre doivent impérativement être **réservés** dans le stock, ou l'opération échoue et rend « false ».
- `produireAssemblage` : permet d'enregistrer la fin d'une opération de production, c'est-à-dire l'assemblage de *quantité* occurrences d'une pièce assemblée *ref* dans l'usine. Les pièces nécessaires à l'assemblage sont retirées du stock, en prenant d'abord dans les pièces **réservées**, puis dans les pièces **disponibles**. L'opération échoue et rend « false » si les constituants ne sont pas présents dans le stock. L'ajout des produits nouvellement assemblés suit la même règle que l'achat de pièces détachées : on bascule de **demandé** à **réservé**, l'excédent éventuel devient **disponible**.
- `reserver` : permet de préparer la satisfaction des commandes, tous les produits figurant sur une commande sont réservés quand la commande est enregistrée. Les éléments passent de **disponible** à **réservés**, s'il n'y a pas assez de produits en stock, le reste est ajouté sous la forme de demandes. Les nouvelles demandes de produits assemblés nécessitent récursivement de réserver les constituants de l'assemblage.

Exemple :

Catalogue : Une « roue » est constituée de 50 « rayon » et 1 « jante ». « rayon » et « jante » sont des pièces détachées.

1. Initialement le stock contient :
   - **Disponible** : 10 jantes, 3 roues
   - **Réservé** et **Demandé** sont vides

2. Une nouvelle commande contenant 4 roues est saisie. On réserve donc 4 « roue » ; le stock devient :
   - **Disponible** : 9 jantes
   - **Réservé** : 1 jante, 3 roues
   - **Demandé** : 1 roue, 50 rayon

3. On achète 1000 rayon au fournisseur ; le stock devient
   - **Disponible** : 950 rayon, 9 jantes
   - **Réservé** : 50 rayon, 1 jante, 3 roues
   - **Demandé** : 1 roue

4. On produit l'assemblage d'une roue ; le stock devient :
   - **Disponible** : 950 rayons, 9 jantes
   - **Réservé** : 4 roue
   - **Demandé** : vide

5. On vend les 4 roues ; le stock devient :
   - **Disponible** : 950 rayons, 9 jantes
   - **Réservé** et **Demandé** sont vides

Question III. : (5 pts)

a) (1 pt) Modélisez **CStock** sur un diagramme de composants.
b) (1,5 pts) Proposez une conception détaillée de **CStock** à l'aide d'un diagramme de classe. Les listes de produits disponibles, réservés et demandés seront représentées par des occurrences de **CListeProduit.**
c) (2,5 pts) Modélisez sur un diagramme de séquence de niveau interaction l'étape 2 du scénario exemple donné ci-dessus (on suppose un composant **CIHM** qui initie l'interaction). Faites figurer le stock et les occurrences de listes de produits qu'il gère. On fera attention à bien préciser les arguments et les valeurs de retours sur les messages. Comme ce scénario nécessite certaines interactions déjà modélisées, on pourra placer un simple commentaire expliquant ce qu'on réutilise, ne recopiez pas ici la réponse à II.d).