Application Fil Rouge P.O.E.C. JAVA

Table des matières

I – But 3
II – Contenu métier 3
1 – Enoncé métier et classes candidates 3
2 - Cas d'utilisation des interactions système 4
3 - Diagramme de classes résultant de la modélisation de l'énoncé métier 4
III – Etapes de mises en œuvre 6
1 – Construction d'une application console non persistante 8
2 – Ajout d'un fragment de couche de persistance codée à la main 10

I – But

Explorer à travers un exemple simple, les diverses technologies exposées au fur et à mesure de la progression de la formation.

II – Contenu métier

1 – Enoncé métier et classes candidates

On veut modéliser très simplement un **petit site de vente en ligne** dont on peut consulter les **produits** en effectuant un **parcourt du catalogue**. On peut effectuer des **achats** d'un ou plusieurs des produits examinés, et se constituer ainsi un **panier**.

Enfin, la **validation de son panier** permet de **passer une commande**. On devra fournir, à ce moment-là, son code client, (si on est déjà inscrit) ou bien ses informations personnelles, si on est un nouveau Client.

Le **Catalogue** de ces produits, est une simple collection indicée, une liste.

Un **Achat** concerne un produit, et le nombre d'exemplaires souhaité (la quantité), ainsi que le sous-total (le prix du produit qu'il faudra ajouter au coût du panier).

Un **Produit** a un code, un nom, un prix et une image associée. Un produit a également un court descriptif (une phrase ou deux).

Option : un produit a également une quantité en stock.

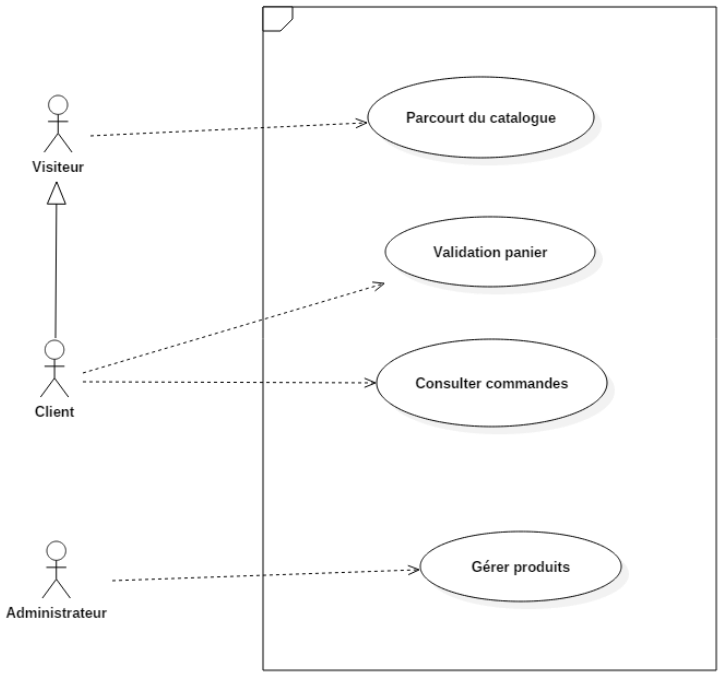
Un **Client** a un code, nom, un prénom, et une adresse.

Enfin une **Commande** a un numéro, une date de commande, un total, concerne un Client et possède une série de lignes (ou 'détails') de commande (des achats).

Remarques :

- Les éléments métier persistants sont les produits, les achats (lorsqu'ils sont validés par une commande), les commandes et les clients. Le panier n'est pas persisté au-delà de la session courante du shopping sur le site (en option on pourra le persister).
- Le modèle métier décrit ici, est susceptible d'évoluer, durant la fabrication des diverses étapes de la construction du projet. Ce document sera mis à jour en conséquence et **constitue un document de travail, consensuel**, partagé par les participants et le formateur.

2 - Cas d'utilisation des interactions système



Détails du cas d'utilisation « Parcourt du catalogue »

Précondition : le visiteur est arrivé sur le site web de vente en ligne et a choisi le lien "parcourt du catalogue" sur la page d'accueil.

Le visiteur parcourt le catalogue en sélectionnant les boutons (ou liens) "produit suivant" et "produit précédant". Le serveur envoie la page détaillant le produit correspondant, et affiche son image. Le visiteur peut, s'il le souhaite ajouter le produit à son panier, après en avoir saisie la quantité désirée.

Sur chacune des pages du parcourt du catalogue, il lui est également possible de *consulter son panier*. Les articles qui y sont présents y sont affichés, ainsi que leur quantité. Il peut en supprimer ou en modifier la quantité. Il lui est également possible, à partir de cette page, de parcourir le cas d'utilisation "**Validation panier**".

Détails du cas d'utilisation « Validation panier »

Si le visiteur le souhaite, il peut depuis la visualisation de son panier, le valider et créer une commande. Si c'est un nouveau client il devra renseigner un formulaire détaillant ses noms, prénom et adresse. Sinon, après avoir saisi son code et son nom, il pourra directement passer sa commande.

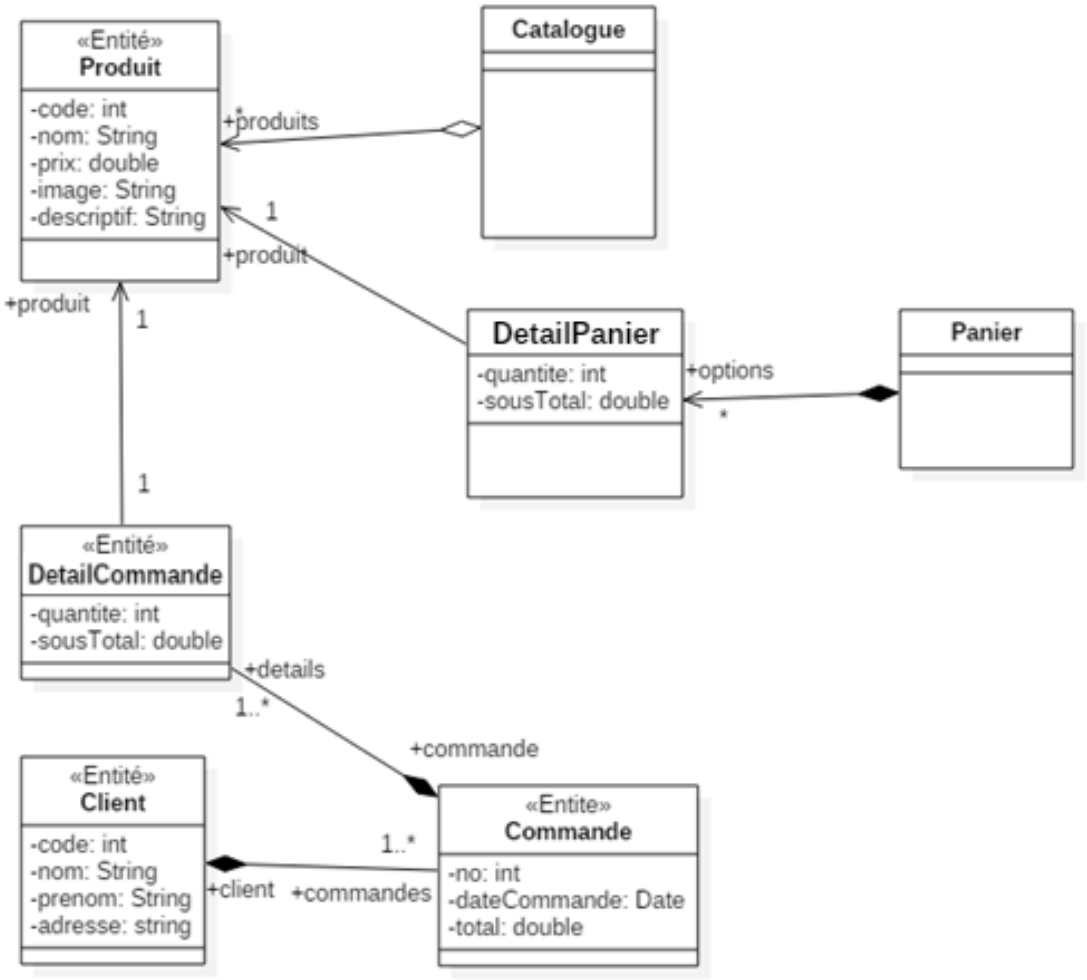
Il est ensuite redirigé sur la page d'accueil.

Détails du cas d'utilisation « Consulter commande »

Après s'être identifié, le visiteur peut, depuis la page d'accueil, consulter les commandes qu'il a déjà passé. Leur date, montant, total lui est présenté. Il peut, en cliquant sur l'un des liens des commandes en connaitre le détail.

Le cas d'utilisation « Gérer produits » ne sera pas, à priori, implémenté dans l'application.

3 - Diagramme de classes résultant de la modélisation de l'énoncé métier


«Entité»
Produit
-code: int
-nom: String
-prix: double
-image: String
-descriptif: String
Catalogue
+produits
*
1
+produit
+produit
1
DetailPanier
-quantite: int
-sousTotal: double
+options
*
Panier
1
«Entité»
DetailCommande
-quantite: int
-sousTotal: double
+details
1..*
+commande
«Entité»
Client
-code: int
-nom: String
-prenom: String
-adresse: string
«Entite»
Commande
-no: int
-dateCommande: Date
-total: double
1..*
+client
+commandes

III – Etapes de mises en œuvre

Les étapes de l'écriture de cette application, destinée à mettre en œuvre les technologies étudiées, ont semblé être :

1. Réflexion sur le contenu métier et modélisation objet en UML.
2. Implémentation du modèle UML dans une application console, avec une utilisation minimale.
3. Ajout d'une couche simple de persistance (une seule classe persistée, pour ne pas perdre de temps), codée à la main, avec JDBC. On utilisera le pattern DAO.
4. Fabrication d'une application Web utilisant les technologies Servlet/JSP/JSTL et implémentant la première version de la boutique en ligne, sans persistance en base de données.
5. Ajout de quelques fonctionnalités AJAX, en utilisant jQuery.
6. Ajout d'une couche de persistance JPA/Hibernate, à l'application Web Tomcat, avec une gestion manuelle, explicite, des transactions.
7. Utilisation de transactions automatiques grâce à un serveur d'applications possédant un container d'EJB (note : cette étape ne sera pas mise en œuvre par les stagiaires, mais on consultera et réfléchira simplement sur un projet solution).
8. Utilisation de Spring, pour effectuer sans container d'EJB, les transactions automatiques et les injections de beans.
9. Restructuration de l'application, par l'utilisation de JSF.
10. Consultation de ses commandes, via une page construite avec Angular.

Chacune de ces étapes (sauf la première…) va **correspondre à un projet Eclipse** contenant la **correction de l'étape correspondante**.

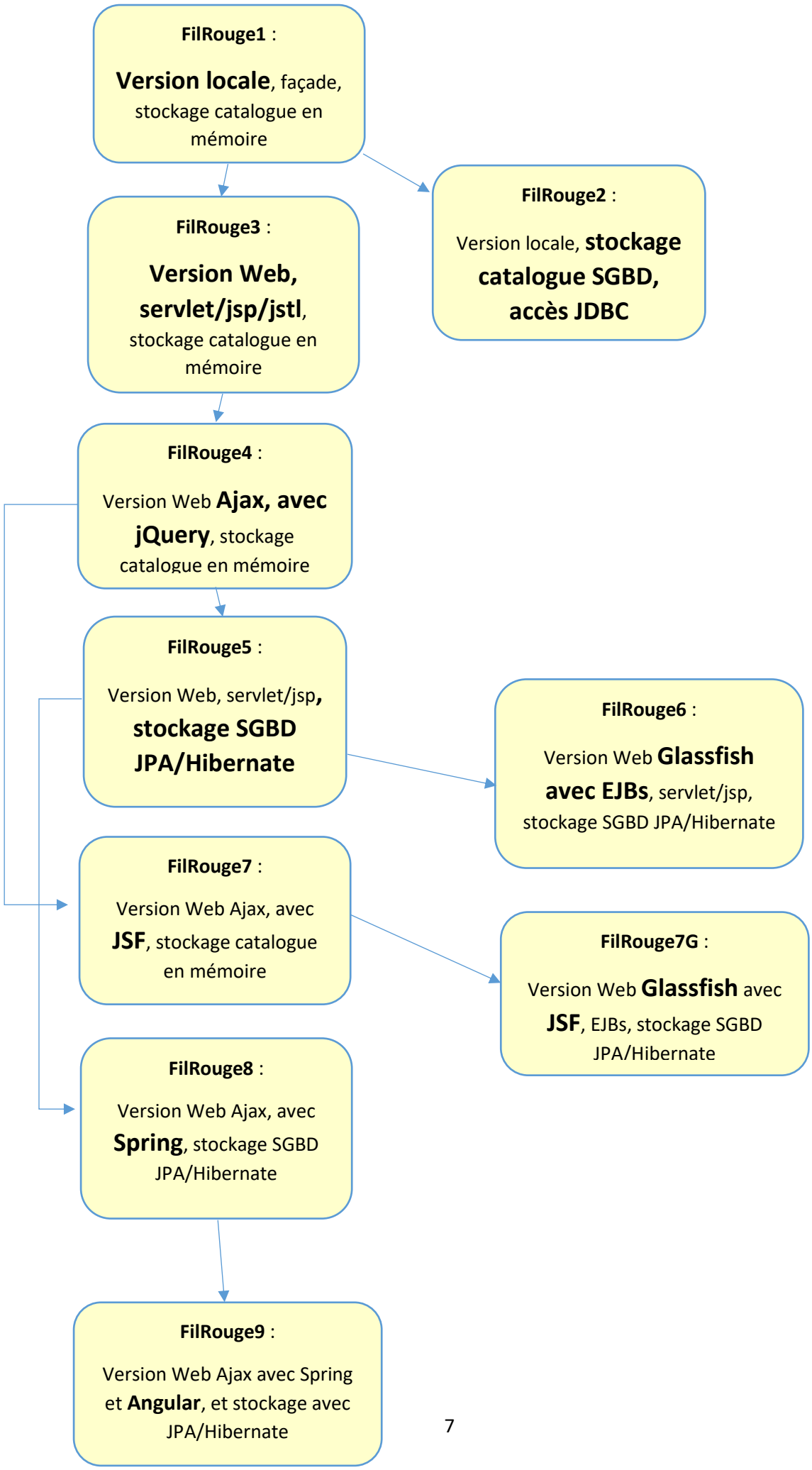
Ces projets seront développés durant les **diverses périodes de travaux pratiques**, en essayant de **respecter le calendrier suivant** :

- Travaux pratiques du 15 et 16 juin : étapes 1, 2 et 3.
- Travaux pratiques du 9 et 10 juillet : étape 4.
- Travaux pratiques du 21 et 22 juillet : étapes 5, 6 (et accessoirement 7).
- Travaux pratiques du 3 aout : étape 8.
- Travaux pratiques du 20 aout : étape 9.
- Travaux pratiques du 28 aout : étape 10.

Le diagramme suivant montre les projets correspondants à toutes ces étapes.

***Note** : pour chaque version, les nouveautés apparaissent en gras et les flèches indiquent de quelle étape du projet est issue une nouvelle version.*

***Exemple** : FilRouge8 est une évolution de FiRouge5 (il ne contient pas de technologie JSF, introduite dans FilRouge7) et il ajoute la technologie Spring à FilRouge5 (point n° 9 ci-dessus).*


FilRouge1 :
Version locale, façade, stockage catalogue en mémoire
FilRouge2 :
Version locale, stockage catalogue SGBD, accès JDBC
FilRouge3 :
Version Web, servlet/jsp/jstl, stockage catalogue en mémoire
FilRouge4 :
Version Web Ajax, avec jQuery, stockage catalogue en mémoire
FilRouge5 :
Version Web, servlet/jsp, stockage SGBD JPA/Hibernate
FilRouge6 :
Version Web Glassfish avec EJBs, servlet/jsp, stockage SGBD JPA/Hibernate
FilRouge7 :
Version Web Ajax, avec JSF, stockage catalogue en mémoire
FilRouge7G :
Version Web Glassfish avec JSF, EJBs, stockage SGBD JPA/Hibernate
FilRouge8 :
Version Web Ajax, avec Spring, stockage SGBD JPA/Hibernate
FilRouge9 :
Version Web Ajax avec Spring et Angular, et stockage avec JPA/Hibernate

1 – Construction d'une application console non persistante

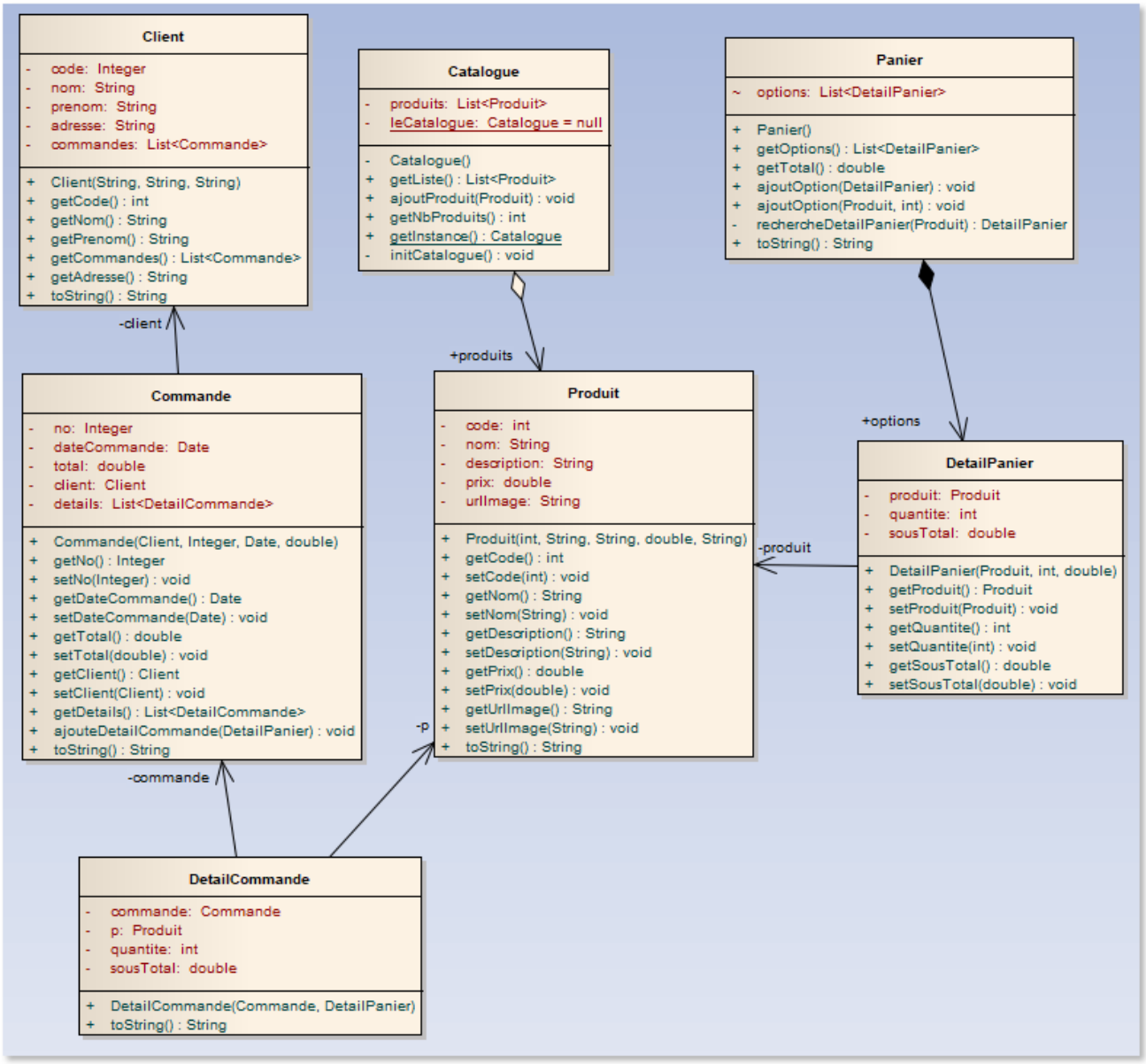
Projet corrigé : FilRouge1 ***TP : 15 et 16 juin***

Dans cet exemple, on s'est attaché à :

- Fabriquer une couche métier conforme à la modélisation métier dégagée par la première étape. Les associations, particulièrement, ont conduit à la création d'attributs de support, typiquement des collections.
- Utiliser une couche de service simple, appliquant le pattern façade, et simulant une session d'interaction avec la couche de présentation.
- Plus généralement, immédiatement structurer ce projet en utilisant une architecture 4 couches, afin de faciliter son portage ultérieur vers une architecture web.

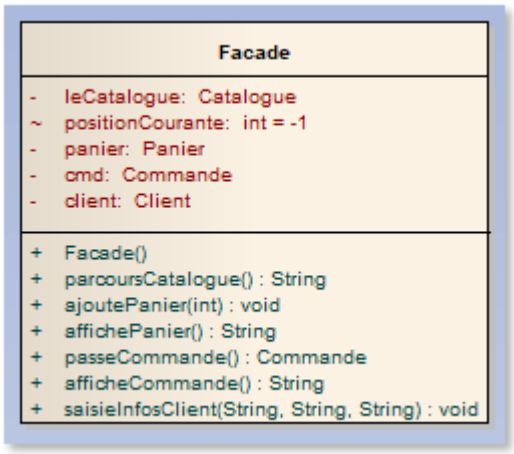
Le "reverse engineering" du projet Eclipse codé, fait avec l'outil Enterprise Architect nous donne :

- **Package metier :**



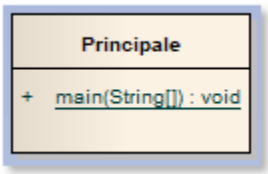
Remarque *: la classe optionAchat a été renommée en DetailPanier.*

- **Package service :**



- **Package persistance** (non implémenté) :

- **Package presentation** (contient une simple fonction main(), simulant un parcourt du catalogue, quelques choix de produits et un passage de commande) :

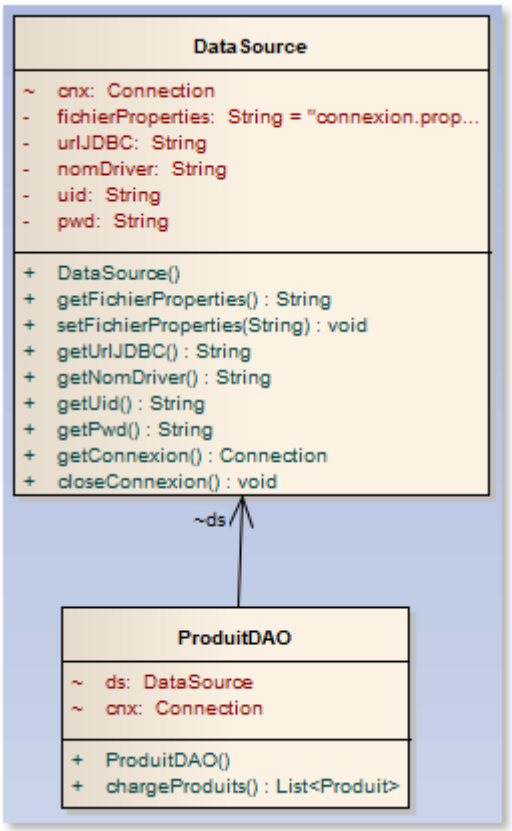


2 – Ajout d'un fragment de couche de persistance codée à la main

Projet corrigé : FilRouge2 ***TP : 15 et 16 juin***

Dans cette version, on a implémenté une couche de persistance minimale, seulement pour la classe Produit en :

- Simulant une `DataSource` serveur, qui retrouve les informations de connexion depuis un fichiers properties appelé `connexion.properties`.



connexion.properties

```
#contient toutes les informations pour se connecter à la base de données
url-jdbc=jdbc:mysql://localhost:3306/test
driver=com.mysql.jdbc.Driver
uid=root
pwd=arati
```

- Implémentant, une seule méthode d'accès à la base de données, chargeProduits(), retrouvant tous les produits issus de la table PRODUITS et les plaçant dans le catalogue.