CONSERVATOIRE NATIONAL DES ARTS ET METIERS

Cnam

CENTRE REGIONAL MIDI - PYRENEES

NFA 031 – Travaux pratiques

IPST-
CNAM
Licence

PROJET

Années 2016-2017
TOULOUSE
J. LAFORGUE

1. LE BESOIN

On se propose de réaliser un programme qui permet de gérer un stock de produits et les *bons de commandes* d'un *site* de vente en ligne.
Le stock contient les produits identifiés par une référence.
Chaque bon de commande contient, les *références* des *produits* commandés.

A terme le programme devra être capable de :

- [FUNC 1] lire deux fichiers texte (fournis) contenant les produits du stock et les bons de commande.
- [FUNC 2] afficher tous les produits du stock du site
- [FUNC 3] afficher tous les bons de commande du site
- [FUNC 4] afficher un bon de commande en particulier
- [FUNC 5] réaliser la livraison de tous les bons de commande non livrées : le stock des produits est mis à jour en décrémentant la quantité des produits, les commandes correctes sont marquées comme étant livrées, les commandes incorrectes sont marquées comme non livrées et un texte associé à la commande précise les produits non disponibles et leurs quantités non disponibles.
- [FUNC 6] modifier les quantités des références de produit d'un bon de commande non livré.
- [FUNC 7] affiche la somme en euros totale de vente des bons de commande livrés.

Bien sur, Il n'est pas envisagé de tout développer en une seule fois. Nous allons réaliser ce programme en 3 étapes.

2. ETAPE 1

Dans cette étape on se propose de réaliser les fonctionnalités :
[FUNC 1] [FUNC 2] [FUNC 3] et [FUNC 4].

Les classes que vous devez créer et/ou modifier sont :

- **Site** Cette classe est la description du site qui contient:
  - la liste des bons de commande (ArrayList<Commande>)
  - la liste des produits en stock (ArrayList<Produit>)

- **Produit** Cette classe est la description d'un produit. Un produit est caractérisé par :
  - la référence du produit (String) (unique) (utilisé dans les bons de commandes)
  - le nom du produit (String)
  - le prix du produit (double)
  - la quantité du produit en stock (int)

- **Commande** Cette classe est la description d'un bon de commande. Un bon de commande est caractérisé par :
  - le numéro du bon de commande (int) (unique)
  - la date de création (String) au format JJ/MM/AAAA
  - le nom du client (String)
  - la liste des références des produits commandés (ArrayList<String>)

Pour démarrer votre programme vous partez du code fourni (projet.zip) dans lequel vous trouvez :
une classe **Projet** avec une méthode main qui crée le **Site** et l' **IHMSite**.
L'IHMSite prend en entrée le Site.

Le constructeur de la classe **Site** doit lire les deux fichiers textes :

- le fichier data/Produits.txt
  Chaque ligne de ce fichier est un produit du stock, de la forme :
  ref_produit;nom;prix;quantité

  Exemple :

```
LIVRE-1;Les cavernes d'acier de Issac Asimov ;8.50;6
LIVRE-2;Dune de Franck Hernert;8.5;4
VETEMENT-1;Pull bleu taille 44 homme;12.0;5
VETEMENT-2;Pantalon jean femme 48;23.0;4
VETEMENT-3;Pantalon jean femme 52;30.0;6
LIVRE-3;Assassin Royal de Robbin Hoob Tome 1;11.0;5
LIVRE-4;Assassin Royal de Robbin Hoob Tome 2;11.0;5
LIVRE-5;Assassin Royal de Robbin Hoob Tome 3;11.0;3
BD-1;Blueberry tome 1;11.5;2
BD-2;Blueberry tome 2;11.5;2
BD-3;Blueberry tome 3;11.5;0
```

- le fichier data/Commandes.txt
  Chaque ligne de ce fichier est un produit d'une commande (pas nécessairement dans l'ordre), de la forme :

num_commande;date;client;ref_produit_commandé

avec • **ref_produit_commandé** est la référence et la quantité du produit commandé qui est au format : **reference=quantite**
où reference est la référence du stock et quantite la quantité du produit commandé.

Exemple :

```
1;21/12/2016;Lafont Pierre;LIVRE-1=1
1;21/12/2016;Lafont Pierre;LIVRE-2=2
1;21/12/2016;Lafont Pierre;VETEMENT-1=3
2;22/12/2016;Dupont Paul;VETEMENT-2=3
3;23/12/2016;ABBE Simon;LIVRE-1=8
4;23/12/2016;DURAND Jean;LIVRE-1=9
2;22/12/2016;Dupont Paul;VETEMENT-3=7
2;22/12/2016;Dupont Paul;LIVRE-3=7
```

(Vous utilisez la méthode String[] Terminal.lireFichierTexte(String nomFichier); pour lire les fichier)
(La méthode String[] split(String ligne,4) permet de décoder une ligne du fichier contenant 4 champs).

Pour traiter le fichier des produits du stock :

Pour chaque ligne lue, on crée un produit et on l'ajoute à la liste des produits du stock

Pour traiter le fichier des bons de commande :

Pour chaque ligne lue, on crée le bon de commande que l'on ajoute à la liste des commandes s'il n'existe pas déjà, puis on ajoute la référence du produit (reference=quantite) au bon de commande.

La classe **IHMSite** crée un formulaire dans lequel :

- deux boutons permettent d'afficher tous les produits et tous les bons de commande (chaque produit ou commande est affiché sous la forme d'une chaîne de caractère)
- un texte de saisi et un bouton permettant d'afficher la commande d'une commande donné. Vous devez tester si le numéro de commande saisi existe ou pas.

Pour réaliser ces trois actions, codez dans la classe Site les trois méthodes suivantes :

public String listerTousProduits()

Cette méthode retourne sous la forme d'une chaine de caractère, tous les produits (une boucle qui appelle la méthode toString de chaque produit)

public String listerToutesCommandes()

Cette méthode retourne sous la forme d'une chaine de caractère, toutes les commandes (une boucle qui appelle la méthode toString de chaque commande)

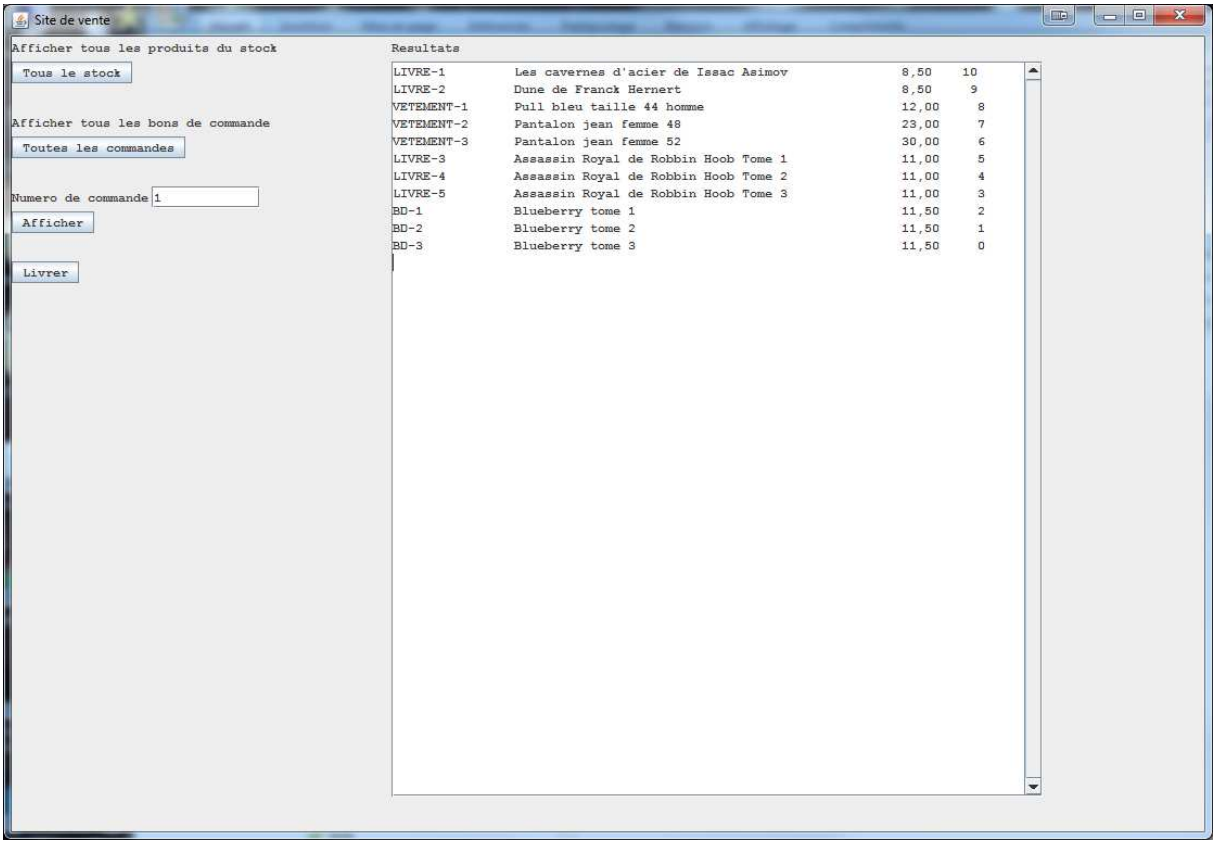
public String listerCommande(int numero)

Cette méthode retourne sous la forme d'une chaine de caractère la commande de numero.

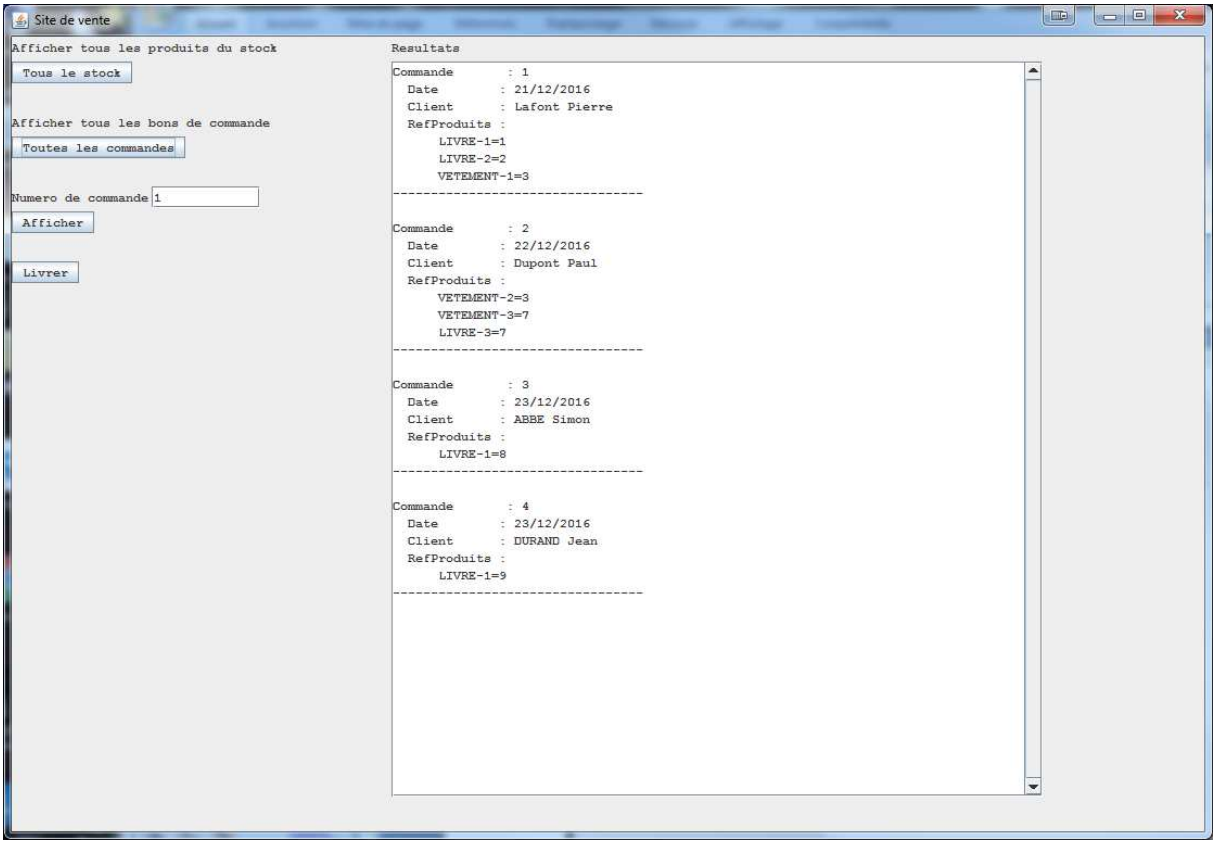
Ces trois méthodes sont appelées dans la méthode **submit** de la classe IHMSite.

Nous supposons que les fichiers texte ne contiennent pas d'erreur de syntaxe: pas de "point-virgule" en moins, toutes les informations sont correctement écrites.

La méthode listerTousProduits affiche, par exemple, le résultat ainsi :



La méthode listerToutesCommandes affiche, par exemple, le résultat ainsi :


Site de vente
Afficher tous les produits du stock
Tous le stock
Afficher tous les bons de commande
Toutes les commandes
Numero de commande 1
Afficher
Livrer
Resultats
Commande : 1
Date : 21/12/2016
Client : Lafont Pierre
RefProduits :
LIVRE-1=1
LIVRE-2=2
VETEMENT-1=3
Commande : 2
Date : 22/12/2016
Client : Dupont Paul
RefProduits :
VETEMENT-2=3
VETEMENT-3=7
LIVRE-3=7
Commande : 3
Date : 23/12/2016
Client : ABBE Simon
RefProduits :
LIVRE-1=8
Commande : 4
Date : 23/12/2016
Client : DURAND Jean
RefProduits :
LIVRE-1=9

3. ETAPE 2

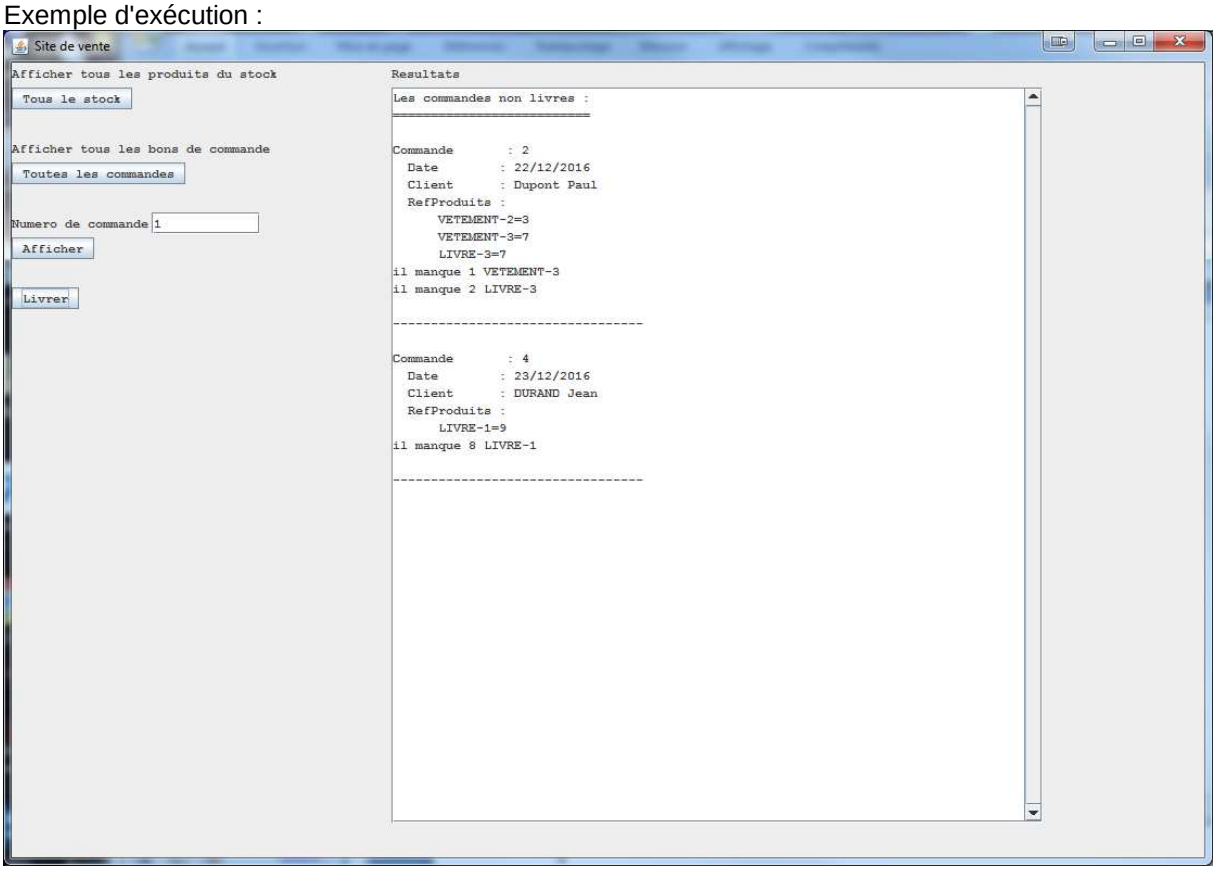
Dans cette étape on se propose de réaliser la fonctionnalité [FUNC 5].

Vous devez ajouter dans la classe Commande, les deux attributs qui permettent de :

- savoir si le bon de commande a été livré ou pas (boolean). Par défaut, il est à faux.
- un texte libre (String) qui précise les raisons pour lesquelles le bon de commande n'a pas pu être livré.

Vous créez dans la classe Site la méthode qui réalise le traitement demandé, et vous ajoutez dans l'IHM un bouton permettant d'exécuter la méthode. Cette méthode n'a pas de paramètre en entrée. Elle retourne la liste de tous les bons de commandes non livrés.

Exemple d'exécution :



4. ETAPE 3

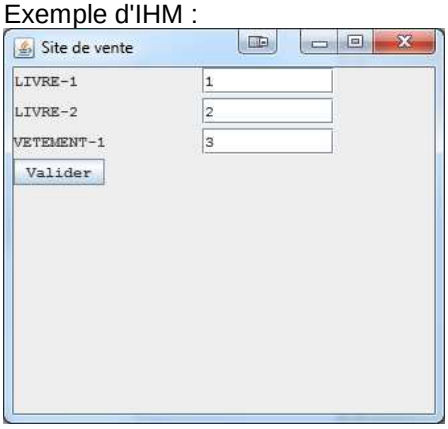
Dans cette étape on se propose de réaliser les fonctionnalités [FUNC 6] et [FUNC 7].

Pour la [FUNC 6], il faut créer une nouvelle IHM dédié à la modification d'un bon de commande non livrée.

On saisi un numéro de bon de commande (déjà fait dans l'étape 1) et on clique sur un nouveau bouton de l'IHM principale (exemple "Modifier"). Le rôle de ce bouton est de créer une nouvelle IHM (à l'image de IHMSite) qui est une classe qui s'appelle par exemple IHMModifierCommande.

Cette IHM affiche des zones de saisies permettant de modifier les quantités des références de produit du bon de commande. Un bouton "Valider" permet de valider les modifications et fermer la fenêtre.

Exemple d'IHM :



La fenêtre se ferme une fois la validation réalisée.

Si le bon de commande est livré on ne peut pas la modifier.

Pour la [FUNC 7], vous ajoutez un bouton dans l'IHM qui permet de faire le traitement et le résultat est affiché dans la zone de résultat.

5. QUELQUES PRECISIONS

Les règles de codage que vous devez suivre sont :

- sauf pour les constantes, les attributs de toutes les classes doivent être privés
- toutes les classes commencent par une majuscule
- tous les attributs, méthodes, paramètres et variables commencent par une minuscule
- les noms de tous les répertoires de package sont en minuscule
- les packages créés commencent tous par "fr.cnam"
- une indentation stricte est respectée dans tout le code
- un minimum de commentaire est nécessaire afin de comprendre ce que fait le code. Surtout quand ce que fait le code n'est pas décrit précisément dans le sujet.

Conseils :

- Ecrivez peu de code à chaque fois afin d'avoir toujours un code qui se compile correctement
- Après chaque étape ou sous-étape, le programme s'exécutant correctement, faites une sauvegarde des sources. Cela permet de pouvoir revenir en arrière si besoin et permet aussi de pouvoir donner à l'enseignant une version du programme qui s'exécute correctement et qui remplit correctement une partie des fonctionnalités demandées dans le sujet.
- Dans un premier temps ne faites pas plus que le sujet ne le demande. Une fois que le programme respecte le sujet, vous faites une sauvegarde et vous pouvez apporter des améliorations qui sont toujours appréciées par le correcteur, à condition qu'elles soient bien commentées.

Dans le fichier Projet.java, vous renseigner les noms et prénoms du groupe.

6. ARCHITECTURE DES REPERTOIRES

L'architecture suivante est une proposition de création des répertoires et des fichiers pour réaliser votre projet. Cette architecture doit être respectée.

projetNFA031
- **data**
  - Produits.txt
  - Commandes.txt
- **src**
  - **fr**
    - **cnam**
      - **projet**
        - Projet.java (*main*)
        - Site.java
        - Produit.java
        - Commande.java
        - IHMSite.java
        - IHMModifierCommande.java
      - **ihm**
        - Formulaire.java
        - FormulaireException.java
        - FormulaireInt.java
        - ....
      - **util**
        - DateString.java
        - Terminal.java
        - TerminalException.java
        - ...
- **bin**
  - <les .class>

Les classes en rouge sont des classes qui vous sont données.

Le répertoire **data** contient les fichiers de données.

7. PLANNING

Vous avez 3 séances de TP pour réaliser ce projet

Le projet sera ramassé 1/4 heure avant la fin de la dernière séance de TP (le temps de les ramasser tous).
L'enseignant vous demandera de déposer vos projets dans le répertoire "commun" spécifique (**lecteur réseau Q:**) qu'il récupèrera sur une clef USB.

Il est donc **impératif** que, au moins un membre du groupe de TP, soit présent lors de la dernière séance afin de donner son projet.