maj 2016

Programmation avancée en Java

Travaux dirigés - Énoncés

par Dr. Samia GAMOURA-CHEHBI

Contexte

- Fil conducteur : La création d'une site de commerce "CAVE A VINS"
  - L'utilisateur s'identifie, puis choisit un type de vin à commander à partir d'une base de données de référence
  - Ensuite, il saisit un nombre de bouteilles à commander, ce qui a pour effet de diminuer le stock disponible
  - Enfin, la commande est sauvegardée dans la base de données et un fichier au format CSV est créé avec toutes les commandes de la base

Contexte (2)

- Packages et classes disponibles :
  - package *commande* :
    - Commande
    - LigneCommande
  - package *production.caveVins* :
    - StockVin
    - Vin
  - package *utils* :
    - AccesBD
    - Authentification
    - Tools

TD 1 : Interface
TD 2 : Classes abstraites
TD 3 : Exceptions
TD 4 : Entrées-Sorties
TD 5a : JUnit
TD 5b : API JDBC pratique
TD 6 : Servlets
TD 7 : Java Server Pages
TD 8 : Finalisation du site

TD 1 - Interface

Définition d'une interface

TD 1 - Objectif

- Définir une interface *Comparable* dans un package *production*

```
package production;

public interface Comparable {
    abstract public boolean estPlusCher(Object o);
}
```

TD 1 - Marche à suivre

- Modifier la classe *Vin* pour qu'elle réalise l'interface *Comparable*

- Implémenter la méthode *plusCherProduit()* (i.e. bouteille la plus chère) dans la classe *StockVin* de façon à pouvoir manipuler plus tard autre chose que des vins

- Modifier la méthode *Main* de la classe StockVin afin de vérifier le produit le plus cher de votre stock.

TD 1 : Interface
TD 2 : Classes abstraites
TD 3 : Exceptions
TD 4 : Entrées-Sorties
TD 5a : JUnit
TD 5b : API JDBC pratique
TD 6 : Servlets
TD 7 : Java Server Pages
TD 8 : Finalisation du site

TD 2 – Classes abstraites

Définition de classes abstraites

TD 2 - Objectif

- Création et extension de classes abstraites

TD 2 - Marche à suivre

- Créer la classe abstraite *Stock* dans le package *production* :

```
public abstract class Stock {
        public Stock() {}
        public abstract int verifierStockRef(Integer ref);
        public abstract void distribuerRef(int quantite,
        Integer ref);
        public abstract Object plusCherProduit();
}
```

- Créer la classe abstraite *Produit* dans le package *production*. La classe Produit possède et gère la variable _ref (la référence est définie pour tous les produits et pas seulement pour une bouteille de vin).
- Modifier les classes *StockVin* et *Vin* afin qu'elles héritent des classes abstraites définies ci-dessus.

TD 1 : Interface
TD 2 : Classes abstraites
TD 3 : Exceptions
TD 4 : Entrées-Sorties
TD 5a : JUnit
TD 5b : API JDBC pratique
TD 6 : Servlets
TD 7 : Java Server Pages
TD 8 : Finalisation du site

TD 3 - Exceptions

Définition et gestion d'exceptions

TD 3 - Objectif

- Définir une classe *StockException* dans le package *production* :
  - qui se construit sur une chaîne indiquant la raison de l'exception
  - qui stocke la date de création

- Implémenter une gestion d'exception sur la méthode *distribuerRef()* de la classe *Stock*

TD 3 - Marche à suivre

- Créer une classe *StockException*
- Rajouter la classe dans le package *production*
- Modifier la méthode *distribuerRef()* de la classe *Stock* et son implémentation dans *StockVin*
- Modifier le code du *main* en conséquence

TD 1 : Interface
TD 2 : Classes abstraites
TD 3 : Exceptions
TD 4 : Entrées-Sorties
TD 5a : JUnit
TD 5b : API JDBC pratique
TD 6 : Servlets
TD 7 : Java Server Pages
TD 8 : Finalisation du site

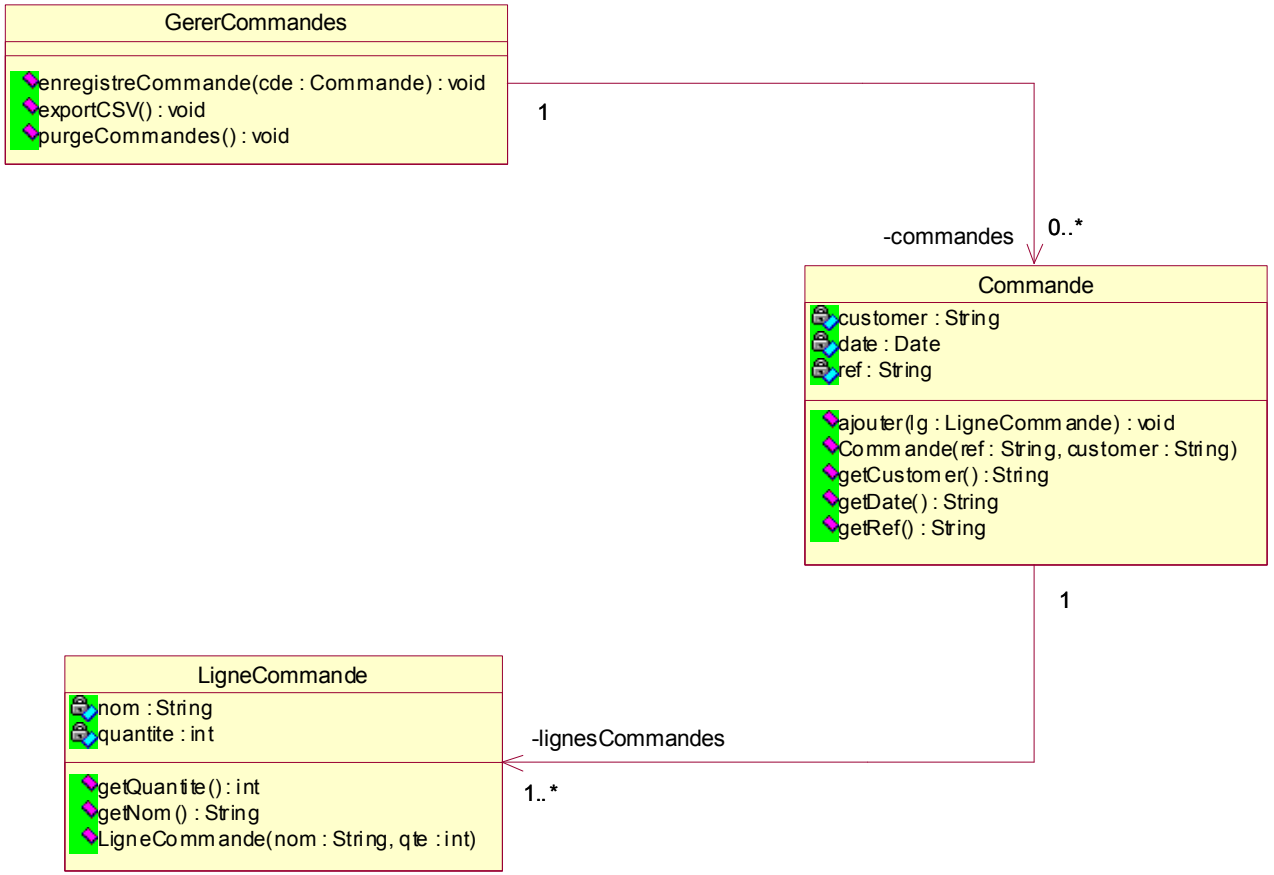
TD 4 - Entrées-Sorties

Gestion des IO

TD 4 – Objectif (1)

- Dans le package nommé commande, créer la classe GererCommandes qui :
  - reçoit les commandes et les enregistre
  - exporte les données en fichier CSV
  - purge la liste des commandes

TD 4 – Objectif (2)



TD 4 - Marche à suivre

- Créer la classe *GererCommande*
- Implémenter les méthodes de la classe *GererCommande* et notamment l'export des commandes sous la forme d'un fichier CSV (valeurs séparées avec comme séparateur le point virgule)
- Créer une méthode *main()* sur la classe *GererCommande* afin de :
  - créer des commandes
  - exporter les commandes

TD 1 : Interface
TD 2 : Classes abstraites
TD 3 : Exceptions
TD 4 : Entrées-Sorties
TD 5a : JUnit
TD 5b : API JDBC pratique
TD 6 : Servlets
TD 7 : Java Server Pages
TD 8 : Finalisation du site

TD 5a – JUnit

TD 5a - Objectif

- L'accès aux données est représentée par la classe ***AccesBD*** (fournie). Avant de l'utiliser, il faut tester son fonctionnement au travers de Junit.

TD 5a – Présentation JUnit

- Junit est un framework Open Source pour la réalisation de tests unitaires sur code Java.
- Objectif : Automatiser les tests

TD 5a - Marche à suivre (1)

- Créer un projet Java et vérifier si les librairies du framework sont rattachées au projet : ***Junit.jar***
- Ajouter la classe ***AccesBD*** dans le projet
- Créer une classe ***TestAccesBD*** qui va jouer le rôle de classe de test.
- Créer une classe ***AllTests*** qui va servir de classe Main pour permettre d'exécuter l'ensemble des tests.

TD 1 : Interface
TD 2 : Classes abstraites
TD 3 : Exceptions
TD 4 : Entrées-Sorties
TD 5a : JUnit
TD 5b : API JDBC pratique
TD 6 : Servlets
TD 7 : Java Server Pages
TD 8 : Finalisation du site

TD 5b – API JDBC pratique

TD 5b - Objectif

- Utiliser les méthodes d'accès à une base de données pour gérer la cave à vins (utilisation de la classe ***AccesBD*** fournie)
- Attention : ne pas oublier d'ajouter une source de données ODBC qui pointe sur la base Access ***Vins.mdb***

TD 5b - Base de données Vins *(1)*

- La base de données Vins est une base Access
- Contenu de la base
  - Table *Cave_A_Vin*

Colonnes	*Type*	*Description*
nom_du_chateau	Texte	Nom du château
appellation	Texte	Appellation de la bouteille de vin (ex graves)
categorie	Texte	Catégorie de la bouteille de vin (ex cru classé)
type	Texte	Nature de la bouteille de vin (ex rouge, rosé, blanc)
millesime	Numérique	Millésime de la bouteille de vin
ref	Numérique	Référence
stock	Numérique	Quantité en stock
prix_vente	Numérique	Prix de vente

TD 5b - Base de données Vins (2)

Contenu de la base (suite)

Table *Commande*

Colonnes	*Type*	*Description*
Ref	**Texte**	**Référence de la commande**
DateCde	**Texte**	**Date de création de la commande**
Client	**Texte**	**Nom du client**

Table *LigneCommande*

Colonnes	*Type*	*Description*
Ref	**Texte**	**Référence de la ligne de la commande**
Commande	**Texte**	**Référence de la commande**
Quantite	**Numérique**	**Quantité commandée**
Vin	**Texte**	**Libellé du vin commandé**

TD 5b - Marche à suivre (1)

- Modifier les classes qui utilisent *AccesBD* :
  - *commande.GererCommandes* :
    - `public void enregistreCommande(Commande cde)`
      insère des données dans la base
    - `public void purgeCommandes()`
      supprime des données de la base
    - `public void exportCSV()`
      récupère des données de la base et exporte de ces données dans un fichier
    - `main()`
      pour les tests

TD 5b - Marche à suivre (2)

- Modifier les classes qui utilisent *AccesBD* (suite) :
  - *production.caveVins. StockVin* :
    - `public int verifierStockRef(Integer ref)`
      renvoie le nombre de bouteilles en stock
      suivant la référence de la bouteille
    - `public void distribuerRef(int quantity, Integer ref) throws StockException`
      diminue le nombre de bouteilles en stock
      dans la base de données
    - `public Vector chateaux()`
      renvoie la liste des noms de châteaux
    - `public Vector vins(String nomChateau)`
      renvoie un vecteur d'objets Vin qui appartiennent
      au château nomChateau
    - `public Vin getVin(Integer ref)`
      renvoie un objet Vin
    - `main()` pour les tests

TD 1 : Interface
TD 2 : Classes abstraites
TD 3 : Exceptions
TD 4 : Entrées-Sorties
TD 5a : JUnit
TD 5b : API JDBC pratique
TD 6 : Servlets
TD 7 : Java Server Pages
TD 8 : Finalisation du site

TD 6 –Servlets

TD 6 - Objectif

- Créer une page WEB utilisant une servlet qui recherche un vin en fonction des critères définis dans le formulaire

TD 6 - Marche à suivre

- Dans la classe StockVin, créer la méthode suivante :
  - *getVins(String ref, String nomDuChateau, String type)* qui retourne les bouteilles de vin qui vérifient les critères en paramètres
- Créer une page HTML *vin.html* qui
  - contient un formulaire de type POST avec les critères suivants :
    - référence, nom du château, type du vin (rouge, rosé, blanc)
  - interroge la servlet *BdAccesServlet.java* qui
    - recherche les informations dans la base à partir d'une instance de StockVin et de la méthode getVins(…)
    - met en forme le résultat

TD 1 : Interface
TD 2 : Classes abstraites
TD 3 : Exceptions
TD 4 : Entrées-Sorties
TD 5a : JUnit
TD 5b : API JDBC pratique
TD 6 : Servlets
TD 7 : Java Server Pages
TD 8 : Finalisation du site

TD 7 – Java Server Pages (JSP)

TD 7 - Objectif

- Créer une page JSP d'interaction avec une base de données

TD 7 - Marche à suivre

- Créer la page *liste.jsp*
  - elle liste les noms de domaines
  - elle permet à l'utilisateur de choisir un domaine
  - elle lui permet alors de se diriger vers la page *choix.jsp*
- Créer la page *choix.jsp*
  - elle permet à l'utilisateur de visualiser toutes les bouteilles du château qu'il a précédemment sélectionné
  - elle lui permet de saisir la référence d'une bouteille qu'il veut commander, ainsi que la quantité désirée
  - Ceci le dirige vers la page *ajout.jsp (cf. TD8)*

TD 7 - Debug

- Placer un point arrêt dans la page *liste.jsp*
- Lancer l'éxécution avec le menu contextuel 'Déboguer sur le serveur…'
- Ouvrir la page *liste.jsp*
  - Le débogueur s'arrête alors dans la page JSP
  - L'utilisateur peut visualiser l'ensemble des variables de la classe Java (context, session, variables locales…) correspondant à la page JSP

TD 1 : Interface
TD 2 : Classes abstraites
TD 3 : Exceptions
TD 4 : Entrées-Sorties
TD 5a : JUnit
TD 5b : API JDBC pratique
TD 6 : Servlets
TD 7 : Java Server Pages
TD 8 : Finalisation du site

TD 8 – Finalisation du site web

TD 8 - Objectif

- Finaliser le site "CAVE A VINS"
  - Ajouter la fonctionnalité permettant d'identifier l'utilisateur.
  - Pour chaque utilisateur, créer un caddy permettant de commander de nouveaux vins
  - Ajouter les pages JSP nécessaires afin de constituer le caddy et ensuite enregistrer la commande en base de données

TD 8 - Marche à suivre (1)

- La fonctionnalité d'identification des utilisateurs sur le site est réalisée au travers d'une page JSP de connexion. Cette page JSP se connecte à un annuaire via l'API JNDI.
- Pour mettre en œuvre cette fonctionnalité, il faut :
  - Récupérer le fichier authentification.xml qui contient l'annuaire des utilisateurs,
  - Récupérer les librairies d'accès à cet annuaire : *dsml.jar* & *providerutil.jar*
  - Récupérer la page *authentification.jsp* qui fait la connexion.
  - Créer une page HTML *login.html* qui permet à l'utilisateur de saisir son login/password.

TD 8 - Marche à suivre (2)

- Créer une page JSP *newCommande.jsp* qui va permettre de saisir son numéro de commande et le nom du client. Une fois validé, le formulaire, cette page pointe sur la page *liste.jsp*
- Créer la classe *Caddy* qui contient une instance de commande. Cette classe doit être créée dans le package *commande.*
- Instancier la classe *Caddy* dès le premier accès à la page *liste.jsp*
- Créer la page *ajout.jsp* qui ajoute une ligne de commande à la commande du caddy et qui diminue le stock disponible
- Créer la page *enregistrer.jsp* qui enregistre la commande du caddy dans la base de données