

Programmation orienté objet avancée  
*Dossier d’énoncé du travail de Java*

*Informatique de gestion*

*UE 230 : Projet informatique intégré*

Vous devez remplir chacune des zones bleues encadrées dans le document.

# Identification des étudiants du groupe

## Etudiant 1

Prénom et nom :

|  |
| --- |
| Olivier Didier |

Bloc (1, 2 ou 3) :

|  |
| --- |
| **2** |

Groupe (A, B, C …) :

|  |
| --- |
| **A** |

## Etudiant 2

Prénom et nom :

|  |
| --- |
| Rollus Simon |

Bloc (1, 2 ou 3) :

|  |
| --- |
| **2** |

Groupe (A, B, C …) :

|  |
| --- |
| **A** |

# Domaine d’application

## Sujet

|  |
| --- |
| **Gestion des commandes et du stockage d’une grande surface** |

## Description du domaine d’application

|  |
| --- |
| **Une application destinée aux clients et employés d’une grande surface.**  **L’application comportera trois espaces d’utilisation : client, employé et adminstrateur.**  **En ouvrant l’application, l’utilisateur sera confronté à une page de login.**  **En rentrant ces identifiants, l’application reconnaîtra le rôle de l’utilisateur en question. Et lui proposera les fonctionnalités en accords avec son rôle.**  **La personne qui ne dispose pas d’identifiants pourra à travers un formulaire d’inscription, se créer un compte « client ».**  **Les comptes « employées » seront eux créer par l’administateur, et les identifiants seront transmis aux personnes concernées en interne.**  **En fonction du rôle de la personne connecté, différentes pages et fonctionnalités sera proposé à l’utilisateur.**  **D’une part, les clients pourront utiliser l’application pour passer leurs commandes et garder un oeil sur l’état d’avancement de celles-ci.**  **Et d’autre part, les employées pourront mettre à jour la base des données du stocks en fonction des approvisionnements quotidiens. L’application mettra à jour automatiquement les stocks en fonction des commandes passées.** |

# Base de données

## Schéma entités-associations

|  |
| --- |
|  |

## Schéma des tables

|  |
| --- |
| jj |

## Documentation des tables

Pour chacune des tables, remplissez les zones ci-dessous.

*N.B. Dans la description des colonnes, placez une croix dans "Oblig" si la colonne est obligatoire et placez une croix dans "ID" si la colonne est identifiante à elle seule (clé primaire ou secondaire (unique)). S’il s’agit d’une colonne clé étrangère, placez une croix dans "FK" et précisez le nom de la table référencée.*

Nom de la table :

|  |
| --- |
| **User** |

Clé primaire :

|  |
| --- |
| **id** |

Description des colonnes :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de colonne | Type(longueur) | Oblig | ID | FK | Table référencée |
| id | int | X | X |  |  |
| roleId | int | X |  | X | Role |
| Username | Varchar(32) | X |  |  |  |
| password | Varchar(128) | X |  |  |  |
| firstName | Varchar(32) | X |  |  |  |
| lastName | Varchar(32) | X |  |  |  |
| phoneNumber | varchar(15) | X |  |  |  |

Nom de la table :

|  |
| --- |
| **Role** |

Clé primaire :

|  |
| --- |
| **id** |

Description des colonnes :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de colonne | Type(longueur) | Oblig | ID | FK | Table référencée |
| id | Int | X | X |  |  |
| title | Varchar(32) | X |  |  |  |

Nom de la table :

|  |
| --- |
| **Order** |

Clé primaire :

|  |
| --- |
| **id** |

Description des colonnes :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de colonne | Type(longueur) | Oblig | ID | FK | Table référencée |
| id | Int | X | X |  |  |
| dateTimeCompletion | DateTime | X |  |  |  |
| dateTimeDelivery | DateTime |  |  |  |  |
| isComplete | Bit | X |  |  |  |
| note | Varchar(256) |  |  |  |  |
| employeeID | Int |  |  | X | User |
| userID | int | X |  | X | User |
| addressID | int |  |  | X | Address |
| statusID | int |  |  | X | Status |

Nom de la table :

|  |
| --- |
| **Address** |

Clé primaire :

|  |
| --- |
| **id** |

Description des colonnes :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de colonne | Type(longueur) | Oblig | ID | FK | Table référencée |
| id | int | X | X |  |  |
| street | varchar(32) | X |  |  |  |
| postalCode | int | X |  |  |  |
| city | varchar(32) | X |  |  |  |
| country | varchar(32) | X |  |  |  |
| deliveryInformation | varchar(128) | X |  |  |  |
| userID | int | X |  | X | User |

Nom de la table :

|  |
| --- |
| **Status** |

Clé primaire :

|  |
| --- |
| **id** |

Description des colonnes :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de colonne | Type(longueur) | Oblig | ID | FK | Table référencée |
| id | Int | X | X |  |  |
| title | varchar(32) | X |  |  |  |

Nom de la table :

|  |
| --- |
| **OrderLine** |

Clé primaire :

|  |
| --- |
| **id** |

Description des colonnes :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de colonne | Type(longueur) | Oblig | ID | FK | Table référencée |
| id | int | X | X |  |  |
| orderID | int | X |  | X | Order |
| articleID | int | X |  | X | Article |
| quantity | int | X |  |  |  |

Nom de la table :

|  |
| --- |
| **Article** |

Clé primaire :

|  |
| --- |
| **id** |

Description des colonnes :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de colonne | Type(longueur) | Oblig | ID | FK | Table référencée |
| id | Int | X | X |  |  |
| label | Varchar(64) | X |  |  |  |
| availableQuantity | int | X |  |  |  |
| description | varchar(128) |  |  |  |  |
| categoryID | int | X |  | X | Category |

Nom de la table :

|  |
| --- |
| **Category** |

Clé primaire :

|  |
| --- |
| **id** |

Description des colonnes :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nom de colonne | Type(longueur) | Oblig | ID | FK | Table référencée |
| id | Int | X | X |  |  |
| label | Varchar(64) | X |  |  |  |

# Fonctionnalités

## CRUD : Listing, insertion, suppression, modification

Vous devez prévoir les fonctionnalités complètes d’administration d’une table de la base de données. Les opérations CRUD (**C**reate, **R**ead, **U**pdate et **D**elete) doivent être faites sur une table qui contient :

* Des **colonnes de types différents** (au moins une colonne de type **texte**, au moins une colonne de type **numérique**, au moins une colonne de type **date** et au moins un **booléen**) ;
* Plusieurs **colonnes facultatives**;
* Au moins une **colonne clé étrangère** vers une autre table de la base de données.

*N.B. 4 Fonctionnalités seront donc implémentées sur une même table, à savoir le listing du contenu de la table (dans une JTable), un formulaire d’insertion d’une nouvelle ligne dans la table, la suppression d’une ou plusieurs lignes choisie(s) par l’utilisateur et la possibilité de modifier une ligne de la table (en remplissant d’abord les champs du formulaire d’insertion avec les valeurs de la ligne choisie, en permettant ensuite à l’utilisateur de modifier les champs du formulaire et enfin de sauver la ligne ainsi modifiée).*

Nom de la table pour les opérations CRUD :

|  |
| --- |
| **Orders** |

*Vérification du respect des contraintes :*

Colonne de type date :

|  |
| --- |
| **dateTimeCompletion**  **dateTimeDelivery** |

Colonne de type booléen :

|  |
| --- |
| **isComplete** |

Colonnes facultatives :

|  |
| --- |
| **dateTimeCompletion**  **dateTimeDelivery**  **note**  **employeeID**  **addressID**  **statusID** |

Colonne clé étrangère (précisez vers quelle table) :

|  |
| --- |
| **employeeID (vers User)**  **userID (vers User)**  **addressID (vers Address)**  **statusID (vers Status)** |

## Recherches

### Recherche 1

Titre de la recherche :

|  |
| --- |
| **Commandes entre deux dates données** |

Objectif de la recherche :

|  |
| --- |
| **Informations sur les commandes passées entre deux dates sélectionnées par l’utilisateur.** |

Jointure entre les tables suivantes (**au moins 3**) :

|  |
| --- |
| **Table 1 : Order**  **Table 2 : User**  **Table 3 : Role (Dans le cas de la recherche à partir d’une session client, les informations concernant son nom et prénom n’apparaîtront pas dans la JTable)**  **Table 4 : Address** |

Description des entrées (introduites par l’utilisateur) :

|  |  |
| --- | --- |
| Critère de recherche | Format (type de composant Swing) |
| Date début | JSpinner |
| Date de fin | JSpinner |

Sorties (dans une JTable) :

|  |  |
| --- | --- |
| Nom de la colonne | Provenant de la table (nom de la table) |
| dateTimeCompletion | Order |
| dateTimeDelivery | Order |
| lastName | User |
| firstName | User |
| username | User |

### Recherche 2

Titre de la recherche :

|  |
| --- |
| **Lignes de commande d’un statut donné** |

Objectif de la recherche :

|  |
| --- |
| **Cette recherche permettra à un utilisateur de regarder toutes les commandes qui possèdent un certain statut. Il pourra dès lors regarder quelles sont les commandes que la société considère comme finies et celles qui sont encore en cours de traitement, envoyées, etc** |

Jointure entre les tables suivantes (**au moins 3**) :

|  |
| --- |
| **Table 1 : Order**  **Table 2 : OrderLine**  **Table 3 : Article**  **Table 4 : Statut**  **Table 5 : Address** |

Description des entrées (introduites par l’utilisateur) :

|  |  |
| --- | --- |
| Critère de recherche | Format (type de composant Swing) |
| statut | JComboBox proposant la liste de tous les noms de statuts existants dans la BD |

Sorties (dans une JTable) :  le tout trié par date de commande (croissante/décroissante au choix)

|  |  |
| --- | --- |
| Nom de la colonne | Provenant de la table (nom de la table) |
| id | Order |
| dateTimeCompletion | Order |
| dateTimeDelivery | Order |
| address | Address |
| label | Article |
| quantity | OrderLine |

### Recherche 3

Titre de la recherche :

|  |
| --- |
| **Utilisateurs des commandes comportant un article d’une catégorie donnée** |

Objectif de la recherche :

|  |
| --- |
| **La recherche permettra à un utilisateur d’effectuer une recherche sur l’ensemble de ses commandes (accessibles) et de trouver la liste des commandes contenant au moins un article d’une catégorie donnée.** |

Jointure entre les tables suivantes (**au moins 3**) :

|  |
| --- |
| **Table 1 : Order**  **Table 2 : OrderLine**  **Table 3 : Article**  **Table 4 : Category**  **Table 5 : User** |

Description des entrées (introduites par l’utilisateur) :

|  |  |
| --- | --- |
| Critère de recherche | Format (type de composant Swing) |
| category | JComboBox proposant la liste des libellés des catégories existantes dans la BD |

Sorties (dans une JTable) : le tout trié par article

|  |  |
| --- | --- |
| Nom de la colonne | Provenant de la table (nom de la table) |
| label | Article |
| availableQuantity | OrderLine |
| id | Order |
| dateTimeCompletion | Order |
| dateTimeDelivery | Order |
| username | User |
| Title | Role |

## Tâche métier

Vous devez implémenter au moins une fonctionnalité relative à votre projet autre que les précédentes (exemples : planification, règles métier, statistiques…). Cette tâche sera implémentée dans la couche métier (calculs, règles métier…). L’exécution de cette tâche métier peut nécessiter éventuellement l’affichage de plusieurs écrans successifs, l’obtention d’informations auprès de l’utilisateur ou des accès en lecture et/ou écriture à la base de données.

Description de la tâche métier :

|  |
| --- |
| **La tache métier de notre application consistera à aider les employés à effectuer l’empaquetage des commandes. Pour ce faire, nous voulons pouvoir afficher l’ensembles des commandes à préparer pour un employé.**  **Pour lancer le processus un admin devra lancer la tache métier.**  **Il s’ouvrira devant lui une fenêtre où il pourra voir les commandes possédant le statut « en cours de préparation » et la date à laquelle celles-ci ont été commandées. Il pourra choisir de les afficher sur un intervalle de dates et ces dernières seront triées par dates.** **Les commandes affichées posséderont un bandeau déroulant qui permettra de sélectionner un employé pour accomplir la tâche.**  **Les employés concernés pourront visualiser les commandes dans une option accessible dans le menu.** |

Entrées (saisies auprès de l’utilisateur) :

|  |
| --- |
| **User = Admin** **Il devra entrer :**   * **Deux dates à l’aide de deux JSpinner.** * **L’employé sur un JComboBox qui proposera une liste de tous les employés** |

Sorties (affichage) :

|  |
| --- |
| **Dans un onglet des sessions employés, il y aura :**   * **La liste des commandes que l’employé devra effectuer** * **Et un bouton pour changer le statut de la commande (de “en cours de préparation” à “en cours d’expédition”)** |

## Thread

Description du processus parallèle (autre que l’affichage de l’heure) :

|  |
| --- |
| **Nous comptions mettre le logo du magasin en fond qui tournera lorsque l’application effectue une recherche.**  **Par exemple lors du login d’un utilisateur. Le temps que la DB cherche la personne nous pourrions faire tourner le logo du magasin. Ou encore lors d’une recherche faite par un utilisateur lors d’une recherche** |