

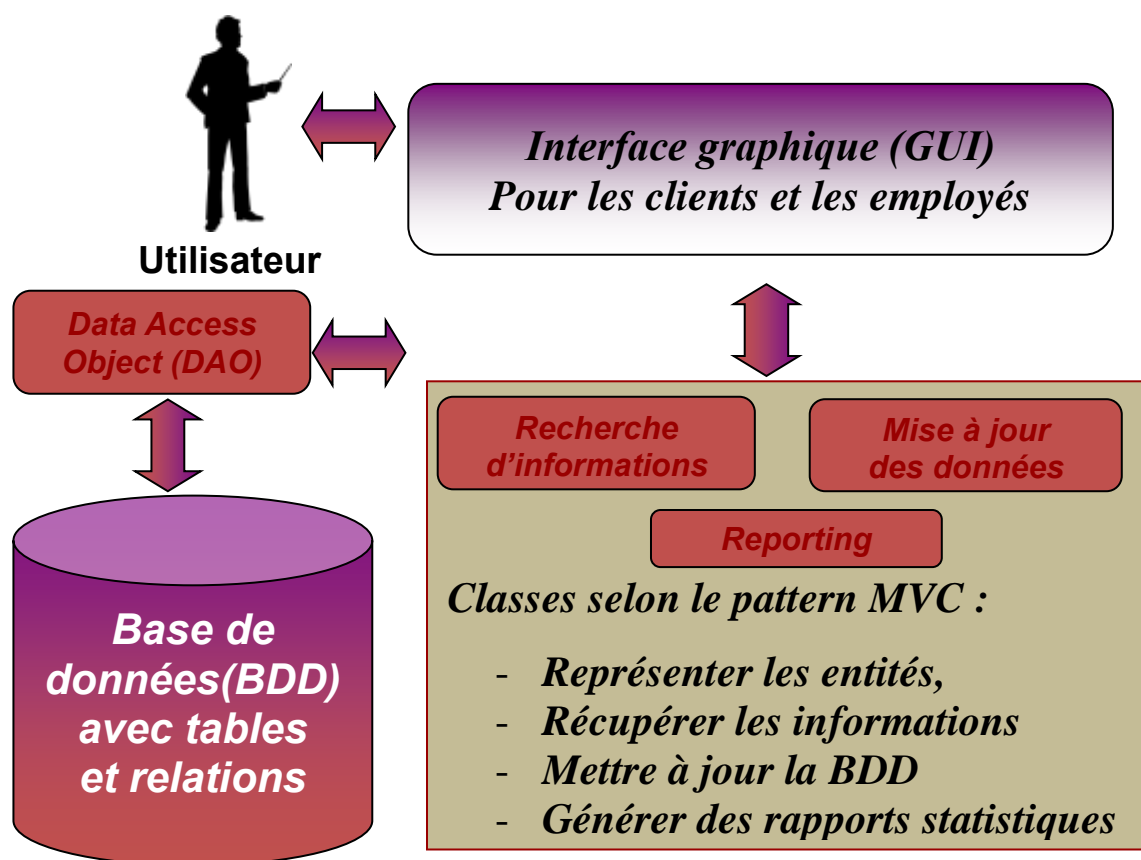
Projet Java ING3 2023. ECE Shopping

Table des matières

Objectif du projet.....	2
Description du programme.....	3
Exigences de mise en œuvre.....	4
Architecture générale du système (Modèle MVC)	4
Conseils pour le développement structuré du projet.....	5
Étape 1 : Modèle relationnel	5
Étape 2 : Création de la base de données.....	5
Étape 3 : Recherche d'informations.....	5
Étape 4 : Accès aux données.....	5
Étape 5 : interface graphique (GUI) et création de rapports (reporting)	5
Livrables	6
Pénalités et Plagiat.....	6
Ressources externes sur campus	7

Objectif du projet

Le but du projet est d'écrire une application de panier d'achat qui permettra au client d'effectuer l'achat d'articles, avec remises possibles et facturation. L'application aidera également l'organisation commerciale de ECE Shopping à conserver et mettre à jour ses archives des ventes et des clients.



Description du programme

Dans ce programme, vous écrirez un ensemble de classes de support pour un programme de shopping. La figure ci-dessous montre un exemple d'interface graphique.



Le programme affiche un catalogue d'articles qui peuvent être achetés. Le client peut acheter plusieurs exemplaires d'un même article.

Certains articles bénéficient d'une réduction lorsque vous en achetez suffisamment. Ces articles ont, en effet, deux prix : un prix d'article unique et un prix d'article en vrac pour une quantité en vrac.

Lorsque vous calculez le prix d'un article en vrac à prix réduit, appliquez autant de quantité en vrac que possible, puis utilisez le prix de l'article unique pour les restes. Par exemple, l'utilisateur commande 12 « briquets » qui coûtent 0,50 € chacun, mais peuvent être achetés en gros (groupe de 10) pour 4 €. Les 10 premiers sont vendus à ce prix de gros (4 €) et les deux extras sont facturés au prix de l'article unique (0,50 € chacun) pour un total de 5 €.

L'application d'achat concerne principalement les articles à vendre, les commandes client générées et maintenues.

Vous devez créer un écran factice pour indiquer le traitement des paiements.

L'application doit être développée pour deux types d'utilisateurs :

1. Clients - Achat, calcul de la facture avec remises, navigation dans la disponibilité, etc.
2. Employés - Gestion de l'inventaire, introduction de diverses offres de rabais, maintien des dossiers des clients, examen des ventes, etc.

Vous êtes censé concevoir et développer la base de données de cette application ainsi que les classes Java nécessaires à la mise en œuvre de l'application.

Exigences de mise en œuvre

- Votre application doit être 100% Java, y compris en cas d'intégration de Framework. De même, les bibliothèques graphiques et API utilisées sont libres, mais 100% Java.
- Les classes, méthodes et attributs nécessaires doivent être conçus en utilisant la notation de diagramme des classes, selon la conception UML. Toutes les classes, méthodes et attributs doivent être expliqués dans votre documentation. Veuillez discuter de la conception avec votre chargé de TP (et futur jury) avant de commencer la mise en œuvre.
- Vous devez être en mesure d'identifier et d'introduire une relation d'héritage et d'agrégation, le cas échéant
- Les interfaces graphique (GUI) Java nécessaires, libres sous leur forme, doivent être ajoutés pour une exécution réussie.
- Les enregistrements doivent être conservés dans la base de données. Votre code Java est censé lire et écrire dans plusieurs tables selon les besoins.
- Chaque table doit contenir au moins 6 enregistrements.

Architecture générale du système (Modèle MVC)

Dans cette section, l'architecture générale de gestion de ce planning a été décrite. Ce système compte principalement 5 modules :

- Le module **Recherche d'informations** : toutes les demandes possibles dans la base de données, selon plusieurs critères de recherche
- Le module **Mise à jour des Données** : toute opération de modification, d'ajout ou de suppression dans la base de données
- Le module **Reporting** : statistiques sous forme de graphiques (camemberts, histogrammes etc.)
- Le module **Data Access Object (DAO)** interroge ou met à jour les données dans la base de données et communique avec les 3 modules précédents
- **L'Interface graphique** (GUI) communique avec les 3 premiers modules pour visualiser graphiquement les informations extraites de la base de données

Selon le pattern **MVC**, votre interface graphique constitue la **Vue** (uniquement l'affichage) dépendante des actions de l'utilisateur (gestion des événements) au niveau du **Contrôleur** (modules Recherche, Mise à jour et reporting). Celui-ci demandera au Modèle de récupérer ou de mettre à jour - via le module d'accès aux données (DAO) - les informations de la base de données, de les organiser et de les assembler (par exemple, en les stockant dans des collections). Ensuite, le contrôleur demandera les données au modèle, les analysera, prendra des décisions et renverra le texte à la vue.

Il vous est conseillé d'adopter le modèle MVC pour le développement d'un projet cohérent. Vous pouvez en savoir plus sur le modèle MVC avec les ressources suivantes :

<https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php/4678736comment-fonctionne-une-architecture-mvc>

https://accu.org/journals/overload/16/88/grenyer_1524/

<https://www.oracle.com/technical-resources/articles/javase/mvc.html>

Conseils pour le développement structuré du projet

Étape 1 : Modèle relationnel

Passez en revue toutes les exigences possibles de la base de données et les critères de recherche. Identifiez les entités possibles, les attributs dans la base de données. Il est important de reconnaître soigneusement le rôle de chaque attribut, puis de décider du type de données de l'attribut. Il jouera également un rôle clé dans la détermination des attributs de clé primaire et étrangère. Documenter les relations entre les entités.

Étape 2 : Création de la base de données

Sur la base du modèle relationnel ci-dessus, créez des tables et des relations à l'aide de MySQL. Insérez les enregistrements dans les tables.

Étape 3 : Recherche d'informations

Passez en revue les exigences de l'utilisateur pour identifier la gamme possible d'informations que vous devez extraire de la base de données. Plus précisément dans le cas des organisations commerciales, il est important que les employés analysent les ventes. Il est également essentiel pour les clients d'analyser leurs achats passés. Par exemple:

- Le montant de la vente d'un produit particulier
- Le nombre de commandes passées par le client au cours des trois derniers mois
- Les offres de réduction qui ont été très bien accueillies

Développez les classes nécessaires pour représenter les entités qui permettront à l'utilisateur d'interroger la base de données.

Étape 4 : Accès aux données

Ce module représente la couche d'accès aux données (DAO) dans la base de données. Via un accès JDBC à la base de données, ce module exécute les requêtes chargées de récupérer ou de mettre à jour les données de la base de données. Il s'agit d'un type d'objet qui se charge pour récupérer les données dans la base de données et qu'un autre type d'objet soit utilisé pour manipuler ces données (couche métier).

Étape 5 : interface graphique (GUI) et création de rapports (reporting)

Une fenêtre d'accueil permettra à l'utilisateur de se connecter à la base de données en saisissant son EMAIL et son MOT DE PASSE. Ces informations, si elles sont stockées dans une table USER, lui donneront des droits d'accès ou/et de mise à jour sur certaines données du planning.

Votre interface graphique affichera de manière ergonomique, claire et fluide toutes les informations pertinentes. Il vous permettra de naviguer intuitivement d'une page à l'autre. Par exemple, une page de votre graphique d'interface peut contenir des menus avec des éléments de menu ou des onglets si vous préférez.

Ce module permet de générer des statistiques (camemberts, histogrammes, etc.) à l'aide de JFreeChart. Vous pouvez trouver les détails dans la section de la page BoostCamp [Cours : Projet POO Java, Section : Ressources \(omneseducation.com\)](#).

Livrables

Le livrable sera un fichier .zip ou .rar contenant les documents suivants :

1) Powerpoint avec les slides suivants :

- Page de garde avec titre, noms coéquipiers et groupe de TD (1 slide)
- Sommaire (1 slide)
- **Répartition des tâches** par fonctionnalités sous forme de tableau (1 slide)
- **Diagramme de classes** avec l'outil [Draw.io](https://draw.io) ou équivalent, selon les patterns **MVC** et **DAO** et présentant les attributs (pas d'attribut objet !), les méthodes, les cardinalités, sans constructeurs ni getters/setters (1 slide ou plus si le diagramme est illisible : par exemple, 1 slide présente l'architecture générale du pattern MVC avec seulement les noms des classes, puis un slide pour chacun des 3 types de modules MVC détaillant le contenu des classes).
- **Design de la maquette de votre interface graphique** principalement composée des 2 éléments suivants :
 - Le *storyboard* : liens entre les pages, symbolisés par une flèche pour naviguer d'une page à une autre (1 à 2 slides)
 - Des *wireframes* de certaines de vos pages (par exemple, 1 page pour une recherche, 1 pour une mise à jour et 1 autre pour le Reporting) : composants graphiques légendés avec les conteneurs encadrés, leur mise en page layout (au plus 3 slides)
- **Versioning GIT** : screenshot et lien avec login et passwd, montrant clairement la bonne utilisation et la répartition des tâches sur la (ou les) version(s) du code partagé entre coéquipiers (1 slide)
- **Bilan individuel et collectif** sans blabla (par exemple sous forme de tableau) sur l'état du travail effectué, des compétences acquises et des points d'amélioration. (1 à 2 slides).
- **Sources** : web avec les liens, livres, supports de cours en citant les auteurs. **Toute source non citée est considérée de facto comme un plagiat.** (1 slide)

2) Code :

- Tous les dossiers et fichiers du projet développé de préférence sur IntelliJ, sinon Eclipse ou Netbeans, avec vos sources **.java**, les bibliothèques **.jar** nécessaires (exemples : accès au serveur de la BDD, JFreeChart, etc.), l'exécutable .jar en mode graphique de votre programme, la documentation **Javadoc** commentée pour les classes et les méthodes (respectez bien le format Javadoc des commentaires au-dessus de chaque classe et méthode de votre code) et tous les fichiers utiles au bon fonctionnement de votre projet (exemples : images, etc.)
- Le **script du fichier des requêtes SQL** généré par l'exportation de votre base de données

Le PowerPoint et le code sont à déposer sur campus le **dimanche 16 avril 2023 à 23h dernier délai** dans la section de la page campus [Cours : Projet POO Java, Section : Livrable à déposer \(omneseducation.com\)](https://omneseducation.com). Ils seront archivés dans un fichier **.zip** ou **.rar** portant les noms des coéquipiers, avec un seul dépôt par équipe (par exemple *hina-maaradji-palasi-segado.zip*).

Pénalités et Plagiat

Comme mentionné clairement au-dessus du drive d'inscription des équipes sur BoostCamp : « **Au-delà de la deadline d'inscription, nous affecterons d'office les étudiants non-inscrits en appliquant une pénalité de non-inscription de -2 points dans votre note de projet.** »

Comme spécifié aussi clairement, les pénalités possibles pour le livrable sont les suivantes :

- Indiquez bien tous les noms des coéquipiers dans le(s) fichier(s) rendu(s) sous peine de pénalités : tout nom absent dans le(s) rendu(s) vaut 0 sans compromis possible, y compris si l'étudiant qui poste le rendu a oublié les noms des coéquipiers.

- Tout code sans les fichiers sources .java vaudra 0.
- 2 pts de pénalité sur la note globale de projet par heure de retard.
- Tout plagiat sera sévèrement sanctionné par 0 et un avertissement, pouvant aller même jusqu'à un conseil de discipline selon la gravité du plagiat.



- L'utilisation d'outil comme chatGPT ou équivalent, sera tracée et aussi très sévèrement pénalisée par 0 et un avertissement sans négociation possible.

Ressources externes sur campus

Les ressources suivantes se trouvent dans la section de la page campus

Cours : Projet POO Java, Section : Ressources (campusonline.me) :

- Le **pattern MVC** : <https://fr.wikipedia.org/wiki/Mod%C3%A8le-vue-contr%C3%B4leur>
- Le **pattern DAO** : [Manipulez vos données grâce aux DAO - Gérez vos données localement pour avoir une application 100 % hors-ligne - OpenClassrooms](#)
- **JDBC** (Java Data Base Connectivity) : [Développons en Java - JDBC \(Java DataBase Connectivity\) \(jimdoudoux.fr\)](#) (Auteur: Jean-Michel Doudoux)
- **My SQL**: <http://dev.mysql.com/doc/refman/5.7/en/>
- **JFreeChart**:
 - [The JFreeChart Class Library](#) (Auteur : David Gilbert)
 - <http://www.jfree.org/jfreechart/api/javadoc/index.html>
 - <http://www.java2s.com/Code/Java/Chart/CatalogChart.htm>
 - <http://www.jfree.org/forum/>
- **Storyboard** : [Exemple de StoryBoard](#)
- **Wireframe**: [Exemples de WireFrame](#)