Compte rendu TP3

Introduction

Le binôme étant constitué de débutants en programmation C++ orientée objet, pour commencer, un travail pratique de niveau débutant a été choisi. Dans le but de pouvoir encore améliorer notre niveau et notre expérience en programmation C++, nous avons, pour cette fois, choisi un travail pratique de niveau intermédiaire. Ce nouveau travail pratique consiste à concevoir une application de magasin en ligne nommée EasyStore. Un magasin étant alors bien sûr constitué de produits, de clients et de commande, il serait alors normal comment nous y sommes pris pour implémenter un tel magasin grâce au langage C++. Nous répondrons donc à cette question dans ce rapport en présentant les différentes classes que nous avons créées.

I - Organisation du travail

Le binôme travail en même temps sur le même travail pratique. Pour une meilleure compilation du code, l'outil utilise ici sera GitHub. Ainsi, chaque membre du groupe peut envoyer du code et le recevoir tout en réglant les conflits qu'il peut y avoir entre les différents bouts de codes générés. Pour ce travail pratique, les différentes questions à traiter ont été réparties entre les membres du groupe. Le code source du projet est disponible dans le dépôt git ayant été utilisé lors de la réalisation de celui-ci. Ainsi, chacun sait ce qu'il doit faire et comment le partager avec l'autre. Dans la suite nous entrons donc plus en détail dans le code de programmation disponibles dans le dépôt git ayant été utilisé.

II - Développement du code

Pour la réalisation du magasin EasyStore, les différentes classes ayant été créées ont été les classes Produit, Client, Commande et magasin. Dans cette partie du compte rendu, nous étudierons le contenu de ces classes une par une.

1 - Classe Produit

Dans notre programme, la classe Produit dispose des attributs suivants :

- Un titre : Étant donné qu'un titre est évidemment une chaine de caractères cet attribut est de type **string**
- Une description : De même que le titre, cet attribut est de type **string**
- Une quantité : Dans un magasin, on ne peut effectivement pas voir de demi produit d'où le choix du type **int** pour cet attribut
- Un prix unitaire : contrairement à la quantité, un prix ne peut pas uniquement être un entier d'où le choix du type **float**.

Un client pouvant lors d'une commande commander plusieurs produits du même type, il serait intéressant de savoir quelle quantité de produit un client veut commander et lorsque celui-ci valide un panier, cette quantité est remise à zéro d'où l'utilité d'un attribut **quantite_client** de type **int**. Nous avons également surchargé des opérateurs tels que << pour rendre l'affichage plus facile, == pour nous aider lors de comparaisons.

2 - Classe Client

Dans notre programme, la classe Produit dispose des attributs suivants :

- Un nom : qui bien sûr est de type **string**
- Un prénom : évidemment de même type que le nom
- Un panier d'achat : Comme dans presque tous les magasins en ligne, il correspond à une sélection d'articles qui sont en attente de validation afin de créer une commande. On a donc un attribut qui doit stocker des produits sans avoir exactement leur nombre nous avons donc donné à cet attribut le type **Vecteur.**

Un autre souci étant que chaque client doit être unique et le nom et le prénom de celui-ci ne suffisant pas à cause de l'existence d'homonymes, nous avons ajouté un attribut **id** de type **string**. Nous avons également ajouté des méthodes permettant chacune d'ajouter un produit au panier d'achat, vider le panier d'achat, de modifier la quantité d'un produit ajouté au panier d'achat, de supprimer un produit du panier d'achat et sans oublier les getters et setters. Nous avons également surchargé des opérateurs tels que << pour rendre l'affichage plus facile, == pour nous aider lors de comparaisons.

3 - Classe Commande

Une commande sera ici constituée d'un client, de son panier d'achat et de son statut. Le statut ici permet de savoir si la commande a été livrée ou non d'où le choix du type **booléen**. Nous avons donc créé la classe avec ses différents attributs, et nous avons créé des getters et setters. Une surcharge de l'opérateur << a été faite afin de faciliter l'affichage d'une commande.

4 - Classe Magasin

Dans notre programme, le magasin dispose des attributs suivants :

- Un tableau de Produits : le type ici est **Vecteur**
- Un tableau de Commandes : le type ici est **Vecteur**
- Un tableau de Clients : le type ici est **Vecteur**

Nous avons eu, dans cette classe, à créer plusieurs méthodes ayant plusieurs fonctions afin d'ajouter des fonctionnalités au magasin. Nous avons donc des méthodes permettant chacune :

⇒ D'ajouter un nouveau produit au magasin. (**void ajout_produit()**)

⇒ D'afficher à l'écran tous les produits référencés dans le magasin. (**void afficher_produit()**)

⇒ D'afficher à l'écran un produit sélectionné par son nom. (**void afficher_produit(std ::string nom)**)

⇒ De mettre à jour la quantité d'un produit sélectionné par son nom. (**void update_produit(std ::string nom, int n)**)

⇒ D'ajouter un nouveau client au magasin. **(void update_client(Client& C))**

⇒ D'afficher à l'écran tous les clients du magasin**. (void afficher_clients())**

⇒ D'afficher à l'écran un client sélectionné par son nom ou son identifiant. **(void afficher_clients(std ::string nom_ou_id))**

⇒ D'ajouter un produit au panier d'achat d'un client. **(void ajout_achat(Produit& P, Client& C, int quantite))**

⇒ De supprimer un produit au panier d'achat d'un client. (void supprimer_achat(Produit& P, Client& C))

⇒ Modifier la quantité d'un produit du panier d'achat d'un client. **(void modifier_achat(Produit& P, Client& C, int quantite))**
⇒ Valider une commande **(void valider_commande(Commande& Com))**
⇒ Mettre à jour le statut d'une commande (void update_commande_status(Commande& Com))
⇒ D'afficher toutes les commandes passées. **(void afficher_commandes())**
⇒ D'afficher toutes les commandes d'un client donné. **(void afficher_commandes(Client& C))**

Conclusion

Au terme de ce travail pratique, il en ressort que nous ne sommes plus juste des débutants en programmation orientée objet C++. En effet, nous maitrisons les syntaxes de base de programmation et nous savons même comment les appliquer pour résoudre un problème de la vie réelle. Notre application de gestion de magasin par exemple, en étant un peu améliorée bien sur car nous ne sommes pas encore des experts peut être utilisée par un vrai magasin. Ce deuxième travail pratique tout comme le premier nous a donc aide à améliorer nos compétences en programmation orientée objet ce qui nous sera évidemment très utile dans la suite de notre formation.