TP 5 – Implémentation d'associations

1 Site de vente

On souhaite réaliser un site de vente par internet où un client peut passer ses commandes. Une commande est caractérisée par son numéro, le jour où elle a été effectuée, le prix total et l'adresse de livraison. Elle est composée de différentes lignes de commande indiquant le produit acheté et sa quantité. Un produit est caractérisé par son nom, son prix, son code et sa marque. Le client est caractérisé par son nom, son prénom et son adresse. On souhaite afficher toutes ses commandes en cours.

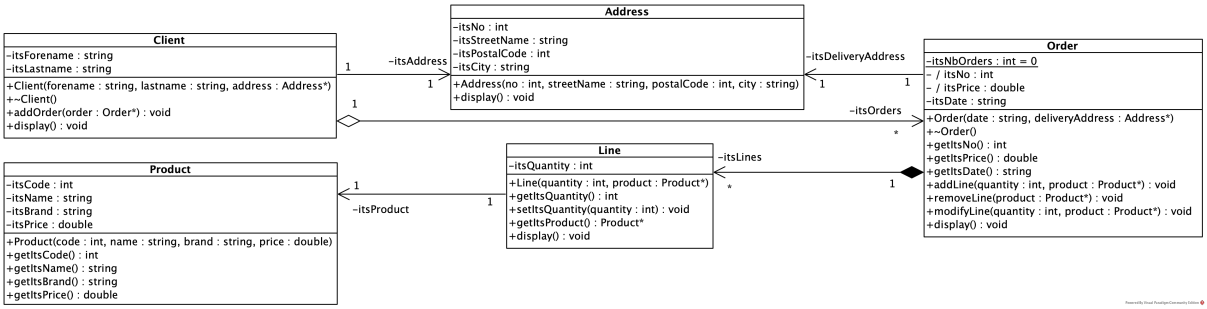

FIGURE 1 – Diagramme de classes de conception

Soit le diagramme de conception de la figure 1. Le « / » indique pour un attribut qu'il s'agit d'un attribut dérivé c'est-à-dire que sa valeur est obtenue ou calculée à partir des autres attributs de la classe. Un attribut souligné correspond à un attribut statique. Le numéro de la commande dépend du nombre de commandes qui est incrémenté pour chaque nouvelle commande. Le prix d'une commande C est calculé comme suit :

$$p(C) = \sum_{l} p(P_l) \cdot q(P_l) \tag{1}$$

où $p(C)$ est le prix de la commande, $p(P_l)$ le prix TTC du produit de la l-ième ligne de commande et $q(P_l)$ la quantité du produit de la l-ième ligne de commande.

1. Ecrire les déclarations des classes et les définitions des méthodes. On utilisera la collection `std::vector` pour implémenter les associations Client → Order et Order → Line.
2. Ecrire la fonction `main` qui
   (a) crée une instance de Client,
   (b) crée une instance de Commande,
   (c) ajoute 2 bouteilles d'eau de source d'Evian à la commande,
   (d) ajoute 1 paquet de mouchoirs de la marque U à la commande et l'affiche,
   (e) ajoute 1 bouteille d'eau de source d'Evian à la commande et l'affiche,
   (f) supprime les mouchoirs de la commande et l'affiche,
   (g) ajoute la commande au client et affiche le client ainsi que ses commandes,
   (h) modifie la quantité des bouteille d'eau de source d'Evian (2 bouteilles) et affiche la commande,
   (i) modifie la quantité des bouteille d'eau de source d'Evian (1 bouteilles) et affiche la commande,

(j) ajoute 2 paquets de mouchoirs de la marque U à la commande et l'affiche,

(k) modifie la quantité des bouteille d'eau de source d'Evian (0 bouteilles) et affiche la commande,

(l) crée une deuxième instance de Commande,

(m) ajoute 1 bouteille d'eau de source d'Evian à la deuxième commande,

(n) ajoute 1 paquet de mouchoirs de la marque U à la deuxième commande,

(o) ajoute la deuxième commande au client et affiche le client ainsi que ses commandes.

2 Gestion des groupes d'étudiants

On souhaite réaliser une application pour la gestion des groupes d'étudiants. On veut pouvoir afficher d'une part le groupe pour un étudiant et d'autre part les étudiants affectés à un groupe, menant à une association bidirectionnelle entre les classes `Student` et `Group`. Le diagramme de conception est donné dans la figure 2.

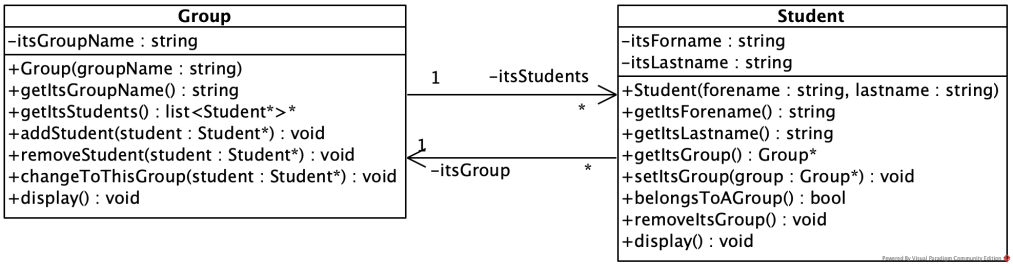

FIGURE 2 – Diagramme de classes de conception

1. Implementer les classes `Group` et `Student`. On utilisera une `std::list` pour réaliser l'association `Group` → `Student`.
2. Ecrire la fonction `main`.