Compte rendu TP

Yvana Touko / Yanis Pella
Robotique 3A

Premier TP : TP1

Le TP1 est le premier sur lequel nous avons travaillé. Nous l'avons choisi car nous voulions commencer par un sujet plus simple afin d'apprendre à programmer à plusieurs qui est un exercice difficile. De plus, nous n'avions ni l'un ni l'autre déjà programmé en orienté objet. N'étant pas expérimentés nous avons préféré nous tourner vers ce projet dans le but de nous familiariser avec le C++.

Pour la répartition du travail, nous avons commencé par coder chacun deux classes : Les classes *Date* et *Livre* (Yanis) et les classes *Lecteur* et *Auteur* (Yvana). La classe *Date* étant très rapide, j'ai (Yanis) pû coder la classe *Emprunt*. Nous avons ensuite mis en commun notre travail.

C'est à ce moment-là que nous avons réalisé la réelle difficulté du travail en groupe quand il s'agit de programmation. En mettant nos codes en commun (via GitHub), nous nous sommes rendu compte que nous n'avions pas la même manière de coder, une logique différente mais aussi une différence au niveau de la nomenclature. Heureusement, nous avons remarqué ce problème rapidement et nous avons pu discuter à propos de cela pour nous mettre d'accord sur une manière de faire et une organisation à avoir. A partir de là, il a été plus simple de coder en simultané sur le même TP.

Etant donné qu'il ne restait qu'une classe, nous avons décidé qu'il était plus simple que je finisse le TP. Yvana a donc commencé le TP3.

Nous avons chacun codé sur nos propres pc étant donné que travailler sur les ordinateurs de l'Esirem nous ajoute une difficulté supplémentaire. Concernant les outils que nous avons utilisés, j'ai utilisé Visual Studio tandis qu'Yvana à travaillé sur Visual Studio Code. Cette différence d'outils ne nous a pas posé de problèmes étant donné que nous sommes passés par GitHub pour nous échanger nos codes.

GitHub est un outil que nous avons eu du mal à utiliser au début étant donné qu'il n'est pas très intuitif. Cependant, une fois maîtrisé, il facilite grandement les choses pour mettre en commun notre travail.

Développement du code :

Dans cette partie, nous allons vous présenter l'organisation de nos classes dans notre code ainsi que les choix que nous avons fait pour permettre de créer un code fonctionnel.
Le TP1 est composé de six classes : *Date*, *Livre*, *Lecteur*, *Auteur*, *Emprunt* et *Bibliotheque*. Pour chaque classe, nous avons commencé par définir leurs variables en *private*. Puis, nous avons créé tous les getters mais aussi les setters si nous pensions que ces variables allaient devoir être modifiées par la suite.

La classe *Date* ressemble beaucoup à celle présente dans le cours. Nous avons cependant ajouté des années. Nous avons décidé de ne pas faire de surcharge d'opérateur pour cette classe puisqu'il nous suffisait d'ajouter une ligne aux surcharges des autres classes pour nous permettre d'afficher la date.

La classe *Livre* comporte beaucoup de variables de classe qui sont donc accompagnées de leurs getters. Nous avons décidé d'ajouter une variable booléenne permettant de savoir si le livre est disponible ou non.

Pour cette classe, nous avons surchagé l'opérateur << pour nous permettre d'obtenir facilement des informations à propos d'un livre. Cette classe comporte une variable *_auteur* de type *Auteur*. Je n'étais pas très à l'aise avec le fait de mettre des objets d'une classe à l'intérieur d'une classe différente mais j'ai fini par comprendre comment faire.

Pour afficher l'auteur du livre dans la surcharge, nous avons décidé de ne pas utiliser la surcharge que nous avons fait dans la classe *Auteur*. La décision peut paraître étrange mais en l'utilisant, le programme affichait trop d'informations et le rendu était très brouillon. Cette classe possède aussi un *vector* de chaînes de caractère pour stocker les isbn des lecteurs l'ayant empruntés.

La classe *Auteur* est une classe assez simple qui comporte seulement quatre variables de classe ainsi que leurs getters, et une surcharge de l'opérateur <<.

La classe *Lecteur* est aussi une classe plutôt simple. Sa particularité est le fait qu'elle dotée d'un *vector* de chaînes de caractère (std::string) permettant de stocker les id des différents livres empruntés par un lecteur.

La classe *Emprunt* est quant à elle une peu plus complexe. Elle a trois variables de classe qui sont trois objets des classes *Dates*, *Livre* et *Lecteur*. Elle a aussi deux méthodes qui permettent d'emprunter et de restituer un livre.

Le dernière classe est la classe *Bibliothèque* qui est de loin celle qui a le plus de méthodes. Elle comporte cependant seulement trois variables de classes qui sont trois *vector* afin de stocker des listes d'auteurs, de lecteurs et de livres.

Dans les méthodes, le plus difficile a été de gérer les erreurs dans le cas où l'utilisateur écrit un texte non valide. Pour les méthodes qui retournent un objet, j'ai fait un test pour savoir si les paramètres d'entrée sont valides. Si ce n'est pas le cas, elles retournent objet vide.

En plus de ces classes, j'ai codé un menu pour permettre à l'utilisateur de pouvoir tester facilement le programme sans devoir lire le code. Pour réaliser cela, j'ai utilisé un switch. La principale difficulté que j'ai rencontrée est encore une fois le fait de gérer les erreurs rentrées (volontairement ou involontairement) par l'utilisateur. Pour palier au fait que l'utilisateur peut rentrer des caractères spéciaux, j'ai entre autres utilisé *isdigit*.

En conclusion, ce TP à été très important pour nous car il nous a permis deux choses (au minimum). Tout d'abord, nous avons pu nous familiariser avec le C++. Nous avons aussi commencé à apprendre comment travailler en groupe (au moins à deux) ce qui est très important pour un futur ingénieur. Faire ce TP, nous a aussi permis d'essayer de résoudre un problème par nous même sans aide ce que nous n'avions pas encore fait.

Second TP : TP3

Le TP3 est le second sur lequel nous avons travaillé. Nous avons choisi ce TP dans le but de se mettre au défi sur un sujet plus difficile que le premier que nous avons fait. Le but est de s'améliorer de plus en plus. Nous n'avons pas tenté les sujets encore plus durs par peur de ne pas pouvoir avancer à cause d'un trop gros gap de niveau, c'est plus nous somme resté sur le choix d'un sujet de niveau moyen.

La répartition du travail s'est faite de manière plus naturelle, Yvana à commencé le TP pendant que je finissais le TP1. Une fois celui-ci fini, j'ai pu venir l'aider sur des problèmes rencontrés lors de la programmations de différentes classes et j'ai pû compléter la classe *Client* ainsi que faire l'ajout de fonctionnalité à la classe *Magasin*. Pendant ce temps, Yvana s'est occupée de la création de la classe *Commande*.

Pour éviter de reproduire la même erreur que dans le TP1, nous nous sommes concertés avant de commencer le TP pour se mettre d'accord sur le rôle des différentes classes et les solutions que nous allions utiliser. De plus, nous avons discuté de la nomenclature que nous utiliserons dans le TP pour éviter les problèmes au moment de réunir nos codes sur GitHub.

La communication entre nous était une des parties les plus difficiles et nous nous sommes rendu compte du défi que cela doit être de travailler dans des grands groupes de programmeurs en entreprise.

Durant ce TP, nous étions plus à l'aise avec GitHub et nous avons gagné beaucoup de temps en utilisant cet outil.

Développement du code :

Dans cette partie, nous allons vous présenter l'organisation de nos classes dans notre code ainsi que les choix que nous avons fait pour permettre de créer un code fonctionnel.

Le TP3 est composé de quatre classes : *Produit*, *Client*, *Commande* et *Magasin*.

Comme pour le TP1, pour chaque classe, nous avons commencé par définir leurs variables en *private*. Puis, nous avons créé tous les getters mais aussi les setters si nous pensions que ces variables allaient devoir être modifiées par la suite.

Dans ce TP, nous avons besoin de moins de classes que dans le TP1, cependant elles sont toutes plus complexes car elles sont plus enchevêtrées les unes dans les autres ce qui rend l'exercice bien plus difficile.

La première classe que nous avons créée est la classe *Magasin*. Nous avons créé cette classe en deux parties. Nous avons tout d'abord ajouté ses variables de classes qui sont trois vectors pour stocker des produits, des clients ainsi que des commandes. Puis dans un second temps, après la création des autres classes, nous l'avons grandement améliorée en ajoutant des fonctionnalités au fur et à mesure de l'avancée du projet.

La seconde classe est la classe *Produit*. Un produit est défini par son nom, sa description, une quantité et son prix. Ces quatre paramètres forment les quatre variables de classe.

De plus, nous avons surchargé l'opérateur << pour afficher facilement les différents produits.

Après la création de la classe *Produit*, nous avons ajouté un certain nombre de fonctionnalités à la classe *Magasin*. La première est l'ajout de produits à un magasin. Pour cela il faut absolument vérifier que le produit ne se trouve pas déjà dans ce magasin et c'est ce que nous avons fait.

Nous avons rajouté une méthode permettant d'afficher tous les produits d'un magasin. Dans la réalité, pour éviter de mettre de l'affichage dans une fonction, ce qui poserait problème dans le cas où nous voulions faire une interface graphique, cette fonction ne fait que renvoyer la liste (vector) de produits au main qui s'occupe de l'affichage.

Nous avons aussi créé une méthode permettant d'afficher les informations d'un produit en le sélectionnant par son nom. Pour ce faire, il faut que la fonction retourne l'objet dont le nom est similaire à celui rentré en paramètre de la méthode. Nous avons rencontré un problème, si le nom du produit en paramètre ne correspond à aucun produit. Pour résoudre ce problème, la méthode retourne un objet *produit* vide dans ce cas.

La dernière méthode que nous avons ajoutée permet de mettre à jour la quantité d'un produit sélectionné par son nom. Pour faire cela, nous avons utilisé la méthode précédente dans cette méthode ce qui nous a permis d'en apprendre plus sur le fait d'utiliser des méthodes d'une classe dans une méthode de la même classe. C'était un cas sur lequel nous n'étions pas encore tombés.

La classe *Client* est composée des différentes informations du client comme son nom et prénom, un numéro d'identification pour permettre d'éviter les problèmes de doublons de prénoms.

De plus, un objet *client* possède un panier qui est en réalité un *vector* contenant des objets *Produit*. La classe *Client* est aussi composée de plusieurs méthodes. Ces méthodes servent à ajouter et supprimer un produit du panier, de vider entièrement le panier d'un client au cas où celui-ci change d'avis. La dernière méthode sert à modifier la quantité de produits disponibles.

La classe comporte aussi une surcharge de l'opérateur << pour permettre d'accéder facilement aux informations d'un client en particulier.

Nous avons encore une fois amélioré la classe *Magasin* en lui rajoutant encore de nombreuses fonctionnalités afin de gérer la relation entre cette classe et celle de la classe *Client*.

La dernière classe que nous avons créée est la classe *Commande*. Elle sert à acheter ou non les articles présents dans le panier d'un client. Cette classe est composée

de trois variables de classe. Un client, le statut de la commande représenté par un booléen et le contenu de la commande qui est un *vector* de produits.

Nous avons encore une fois surchargé l'opérateur << pour afficher les différentes informations d'une commande.

Nous utilisons la classe *Commande* via la classe *Magasin* c'est pourquoi elle n'a pas d'autres méthodes que les getters des variables de classe.

Pour finir nous avons encore une fois ajouté plusieurs fonctionnalités à la classe *Magasin* afin de gérer les commandes. Nous avons ajouté une méthode pour valider une commande. Cette méthode met simplement les produits du panier dans le *vector* de produits de la classe *Commande*.

Nous avons aussi rajouté une méthode permettant de mettre à jour le statut d'une commande afin de permettre de la définir comme "achetée". Les deux dernières méthodes nous permettent d'afficher les commandes passées ainsi que toutes les commandes d'un client en particulier.

Dans ce TP, par manque de temps, nous n'avons pas pu écrire le code permettant l'enregistrement des données. De plus, il y aurait plusieurs choses à améliorer. Par exemple, contrairement au premier, nous n'avons pas fait en sorte de prévoir les erreurs et si on rentre des caractères non prévus, le code n'a pas de solution à cela. Dernièrement, nous avons eu un problème au niveau de la partie *Commande* et celle du menu pour une raison qui nous échappe (sûrement un problème de référence pour la commande).

Le menu n'étant pas fonctionnel, il faut lire le code et décommenter les parties que vous avez envie de tester, désolé pour ce problème.

En conclusion, en réalisant ce sujet plus difficile, nous avons pu nous améliorer dans notre logique et nous entraîner à solutionner par nous même des problèmes un peu plus complexes. Cette dernière chose n'est pas anodine et avoir essayé cela nous sera très utile pour la suite de nos études. Nous pouvons encore grandement nous améliorer en travail de groupe et ce TP y a participé. Pour finir, ce TP a été utile pour nous montrer et nous faire apprendre comment gérer des classes "entremêlées" entre elles.