Compte-rendu

TP C++ : Gestion des Entrées / Sorties

“Sauvegarde et restitution d’un catalogue de trajets”

AKAYDIN Aydın
BARANIAK Dawid
NOLAN Oisín

Description détaillée du format de fichier:

Chaque ligne du fichier représente une trajet composé ou une trajet simple, la restitution est faite en lisant ce fichier ligne par ligne. On a choisit de utiliser le format suivant pour la sauvegarde des trajets:

- Trajet Simple:

TS#VilleDepart:VilleArrivee:ModeDeTransport:

On marque le type de trajet par TS#, ville de départ, ville d'arrivée et le mode de transport sont séparés par des " : " et le fin est aussi marqué par un " : ".

Autrement dit on lit la ligne jusqu'à ce qu'on rencontre un marqueur qui désigne le type du paramètre qu'on vient de lire. '#' signifie qu'on a marqué le type de trajet… Les particularités de celui-ci sont ensuite marqués par de ':' après chaque paramètre.

- Trajet Composé:

TC#VD1:VA1:MDT1:& VD2:VA2:MDT2:

On marque le type de trajet par TC#, les différents trajets simples composant le trajet composé sont séparés par des " & ". VA1 et VD2 sont la même ville, l'arrivé d'un trajet étant le départ du suivant.

Fichier demo:

TS#Lyon:Bordeaux:Train:
TC#Lyon:Marseille:Bateau:&Marseille:Paris:Avion:
TS#Lyon:Paris:Auto:

Nouvelle Fonctionnalités:

On a dû ajouter des nouvelles méthodes pour assurer les nouvelles fonctionnalités demandés. Dans les classes Trajet, TrajetSimple et TrajetCompose; name( ) qui renvoie le type de trajet(TS ou TC) pour faciliter la distinction entre plusieurs objets de type trajet, format( ) qui renvoie un string contenant les attributs formatés et prêt pour être écrit dans un fichier.

La structure de données qu'on a défini préalablement pour l'implémenter le catalogue est TrajetList. Dans ce class on a ajouté les méthodes OutputFile( ) qui permet d'écrire son contenu dans une fichier et getFormat( ) qui est utilisé pour créer le string formaté décrivant un TrajetCompose qui utilise la même structure de données pour stocker ses TrajetSimples.

Les méthodes pour la sauvegarde sélective et restitution étaient ajoutés dans la class Catalogue.

Sauvegarde:

Tout les méthodes de sauvegarde créent un TrajetList selon les différents critères de sélection en parcourant le catalogue et ce TrajetList est sauvegardé dans une nouvelle fichier créé avec la méthode OutputFile( ) de TrajetList qui prend en paramètre une nom de fichier. Les différentes méthodes pour la sauvegarde sélective prennent en paramètre nom de fichier qui va être créé et aussi les parametres necessaires pour faire la sélection.

Restitution:

Pour la restitution on a une seule méthode qui traite tout les différent cas qui prends en paramètre le nom de fichier qui va être lu et une int qui désigne les différents cas de restitution sélective. Si le nom de fichier taper ne peut pas être trouvé, un message d'erreur s'affiche et restitution est annulé. On peut lire la même fichier plusieur fois et cela crée des doublons des trajets dans notre catalogue, les doublons sont permis selon notre implémentation.

Problèmes rencontrés et améliorations possibles

Dans un premier temps on a eu des problèmes d'accès entre nos différentes classes car on essayait d'ajouter une seule méthode pour la sauvegarde mais finalement on a du modifier tout les classes héritant de Trajet et les ajouter des méthodes pour faciliter l'écriture.

Pour la partie "restitution des trajets" il fallait décider comment les paramètres caractérisant les trajets seront récupérés à partir d'une ligne du fichier. Nous avons pris la décision de nous servir uniquement de la méthode substr(). Cela a permis de garder le format de fichiers le plus minimaliste possible et clair à la fois.
Il fallait aussi prendre une décision concernant l'implémentation des différentes options de restitution. Une de possibilités était de diviser le problème et créer plusieurs méthodes, chacune pour l'option différente. Néanmoins cette solution générerait beaucoup de répétitions dans le code (uniquement quelques conditions sont ajoutées au code de décodage du fichier par défaut). Nous avons donc décidé pour cette partie de garder toute la restitution dans une seule méthode mais avec plusieurs conditions qui à première vue ne sont pas très lisibles. Le code sera donc plus difficilement maintenable mais est plus concis.