Compte rendu de TP : POO 2

Par : Loïc CASTELLON & Florian MUTIN, 3 IF 4

Le 19/12/2017

Table des matières

Fichier de sauvegarde

Type de fichier et format

Nous avons choisi d'enregistrer nos trajets dans un fichiers .txt pour les raisons citées ci-après. Nous pouvons écrire et lire facilement dans le fichier via un éditeur de texte, cela nous permet donc de débuguer facilement notre code. Nous n'avons pas à transformer nos données textuelles en données binaires.

Nous avons défini un protocole pour l'écriture et la lecture des trajets. Un trajet simple est écrit sur 4 lignes de la manière suivante :

TS // Trajet Simple, marqueur d'un trajet simple **VILLE DE DEPART** **VILLE D'ARRIVEE** **MOYEN DE TRANSPORT**

Un trajet composé est écrit sur 4N+4 lignes avec N le nombre de trajets simples dont il est composé. Il est écrit de la manière suivante :

TCD // Trajet Composé Début, marqueur du début du trajet composé **VILLE DE DEPART DU TRAJET COMPOSE** **VILLE D'ARRIVEE DU TRAJET COMPOSE** //écriture ordonnée des trajets simples qui composent le trajet composé **TCF** // Trajet Composé Fin, marqueur de fin du trajet composé

L'ajout des marqueurs **TS** et **TCD**, **TCF** nous permet d'effectuer rapidement la différence entre les différents types de trajets. L'écriture des villes de départ et arrivée de trajet composé juste après le marqueur TCD nous permet de faciliter la sélection des trajets selon les villes de départ et/ou arrivée.

Contenu exact de notre fichier demo.txt

demo.txt
TS LYON BORDEAUX TRAIN TCD LYON PARIS TS LYON MARSEILLE BATEAU TS MARSEILLE PARIS AVION TCF TS LYON PARIS AUTO

Modifications

Ci-dessous sont répertoriées les méthodes qui ont été ajoutée à chaque classe pour répondre aux attentes de ce TP. Les modifications liées à l'amélioration de l'expérience utilisateur dans le menu sont toutes dans le Main. Les modifications liées à l'ajout de sauvegarde sont dans les autres classes.

Amélioration de l'expérience utilisateur

Nous avons créé deux méthodes et modifié notre code afin d'améliorer l'expérience utilisateur.

La première amélioration est due à l'ajout de la méthode ReadChoice. C'est une méthode qui permet d'attendre l'entrée au clavier d'un entier compris entre les deux bornes renseignées. Cette méthode est utilisée à chaque fois qu'un choix multiple est proposé à l'utilisateur. Ainsi, le programme ne peut plus entrer dans une boucle infinie comme dans la version précédente.

La seconde amélioration est que l'utilisateur peut désormais saisir des espaces lorsqu'il renseigne une vile. En effet toutes les saisies au clavier sont rangées dans des strings et non plus dans des char*. Nous avons également créé la méthode MajusculeString, qui permet de remplacer toutes les minuscules d'un string en des majuscules. Cette méthode vient remplacer l'ancienne méthode qui effectuait cette modification sur des char*. Cette méthode est donc utilisée à chaque saisie textuelle de l'utilisateur.

La troisième et dernière amélioration est la mise en place d'un menu principal puis de menus secondaires afin de guider l'utilisateur. En effet cela évite que l'utilisateur se retrouve face à 14 choix peu explicites au premier coup d'œil. Cette modification se traduit par l'ajout dans main.cpp de switchs imbriqués.

Main

- ```
  bool ReadChoice(unsigned int & N, unsigned int min, unsigned int max );
  // Mode d'emploi : permet de récupérer un unsigned int entre min et max
  // Contrat : min est inférieur à max
  ```

- ```
  string MajusculeString(string s);
  // Mode d'emploi : retourne le string avec des majuscules
  // Contrat : aucun
  ```

Lecture et écriture de la sauvegarde

Les sauvegardes sont lues et écrites dans le fichier d'adresse fichierSauvegarde. Ce nouvel attribut est modifiable par l'utilisateur dans le menu avec l'appel de la méthode SetFichierSauvegarde.

Les méthodes GetNbTrajet et GetNbTotalDeTrajet sont appelées par le main afin de forcer l'utilisateur à la saisie d'un nombre cohérent.

La lecture de la sauvegarde se décline en 4 méthodes différentes pour répondre au cahier des charges du TP. Ces méthodes commencent par GetSauvegarde. Ces méthodes sélectionnent les trajets à charger selon leurs paramètres et les ajoutent au catalogue. La méthode AfficheSauvegarde permet d'afficher la sauvegarde avant de demander à l'utilisateur la saisie d'un intervalle.

L'écriture de la sauvegarde se décline également en 4 méthodes différentes pour répondre au cahier des charges du TP. Ces méthodes commencent par SaveSauvegarde. Ces méthodes sélectionnent les trajets à sauvegarder selon leurs paramètres et font appel aux méthodes Save des classes TrajetSimple et TrajetCompose afin d'écrire dans la sauvegarde.

Catalogue

- ```
  string fichierSauvegarde;
  // l'adresse du fichier qui sera utiliser pour la lecture et
  //     l'écriture des sauvegardes, l'attribut est private
  ```

- ```
  void SetFichierSauvegarde(string fichier);
  // Mode d'emploi : modifie le fichier de sauvegarde
  // Contrat : aucun
  ```

- ```
  unsigned int GetNbTrajet() const;
  // Mode d'emploi : renvoi le nombre de trajets présents dans le
  ```
  ```

```
//     catalogue
// Contrat : aucun
```

- ```
  int GetNbTotalDeTrajet() const;
  // Mode d'emploi : renvoie le nombre total de trajets sauvegardés dans
  //     le fichier de sauvegarde
  // Contrat : le fichier de sauvegarde est renseigné
  ```

- ```
  void GetSauvegarde() const;
  // Mode d'emploi : ajoute tous les trajets sauvegardés
  // Contrat : aucun
  ```

- ```
  void GetSauvegardeTypeTrajet(bool simple) const;
  // Mode d'emploi : ajoute les trajets simples (type=true) ou composés
  //      (type=false) sauvegardés
  // Contrat : aucun
  ```

- ```
  void GetSauvegardeDepartArrivee(string depart, string arrivee) const;
  // Mode d'emploi : ajoute les trajets sauvegardés selon la ville de
  //     départ et/ou d'arrivée
  // Contrat : aucun
  ```

- ```
  void GetSauvegardeDelta(int min, int max) const;
  // Mode d'emploi : ajoute les trajets sauvegardés entre les indices
  //     min et max
  // Contrat : min < max <= nombre total de trajet dans la sauvegarde et
  //     le fichier de sauvegarde est renseigné
  ```

- ```
  void AfficheSauvegarde() const;
  // Mode d'emploi : affiche la sauvegarde
  // Contrat : le fichier de sauvegarde est renseigné
  ```

- ```
  void Save() const;
  // Mode d'emploi : ajoute tous les trajets au fichier de sauvegarde,
  //     si le fichier fichierSauvegarde n'existe pas il est créé
  // Contrat : aucun
  ```

- ```
  void SaveTypeTrajet(bool simple) const;
  // Mode d'emploi : ajoute les trajets simples (type=true) ou composés
  //     (type=false) au fichier de sauvegarde, si le fichier
  //     fichierSauvegarde n'existe pas il est créé
  // Contrat : aucun
  ```

- ```
  void SaveDepartArrivee(string depart, string arrivee) const;
  // Mode d'emploi : ajoute les trajets au fichier de sauvegarde selon
  //     la ville de départ et/ou d'arrivée, si le fichier
  //     fichierSauvegarde n'existe pas il est créé
  // Contrat : aucun
  ```

- ```
  void SaveInterval(unsigned int start, unsigned int end) const;
  // Mode d'emploi : ajoute les trajets au fichier de sauvegarde qui
  //     se situent dans l'intervalle [start,end]. Si fichierSauvegarde
  //     n'existe pas il est créé
  // Contrat : start est inférieur à end
  ```

Trajet

- ```
  virtual void Save(fstream& fs) const = 0;
  // Mode d'emploi : écriture du trajet dans un fichier de sauvegarde
  //     via l'ofstream passé en paramètre
  // Contrat : aucun
  ```
  ```

TrajetSimple

- ```
  void Save(fstream& fs) const;
  // Mode d'emploi : écriture du trajet dans un fichier de sauvegarde
  //     via l'ofstream passé en paramètre
  // Contrat : aucun
  ```

TrajetCompose

- ```
  void Save(fstream& fs) const;
  // Mode d'emploi : écriture du trajet dans un fichier de sauvegarde
  //     via l'ofstream passé en paramètre
  // Contrat : aucun
  ```

Conclusion

Problèmes rencontrés

Nous n'avons pas rencontré de problème particulier lors de ce TP.

Améliorations possibles

Dans la méthode ReadChoice, si l'utilisateur saisie « 9dje5 » la méthode va considérer que l'utilisateur a saisi le nombre 9. Nous pourrions ainsi chercher à améliorer cette méthode pour interdire ce type de saisie.

La méthode MajusculeString ne s'occupe pas des accents, cédilles, et autres caractères spéciaux. Nous pourrions donc améliorer notre application en s'occupant de ces cas particuliers.

Si on charge dans la sauvegarde un trajet composé déjà présent dans le catalogue il est ajouté en un deuxième exemplaire et aucun message ne s'affiche. Nous pourrions prendre ce cas en compte en améliorant la méthode Equals dans Trajet.cpp.