TP6_DM-CR_CANO_COUPPOUSSAMY

Cano Gabriel et Couppoussamy Alex

3 Novembre 2017

1 Partie A et B (Implémentation)

Terminal

Cette classe est la classe mère. Plusieurs classes héritent d'elle. Cette classe contient une méthode virtuelle, la méthode ajoutLigne, elle est donc inconstatable.

L'attribut liaison est une liste de pointeurs de Ligne. Ligne est un template. Ici, Ligne a comme argument Moyens.

L'attribut tempsMoyenCorrespondance est un double qui est le temps moyen à attendre avant de prendre le prochain transport depuis le terminal courant. Nous avons pris la décision de mettre le même temps d'attente quelques soit la destination depuis un terminal.

L'attribut flux est une liste de nombre de passager qui arrive depuis toutes les lignes qui arrivent dans le terminal courant. L'indice à laquelle figure un flux correspond à l'indice de la ligne à laquelle il correspond. Dans les classes filles, la méthode ajoutLigne(Ligne¡Moyens¿* l, int f), teste si la ligne arrive au Terminal courant. Si c'est le cas, le flux passé en paramètre est ajouté, sinon 0 est ajouté.

On déclare un template Ligne au lieu d'inclure le fichier où il y a le template complet car on a inclus Terminal.h dans celui-ci. On a donc voulu éviter l'inclusion infini.
La méthode suppLigne(Ligne¡Moyens¿* l), on supprime une ligne de la liste.

Gare

Cette classe hérite de Terminal et redéfini les constructeurs. Elle implémente la méthode virtuelle de Terminal : ajoutLigne. Cette méthode vérifie que la ligne qu'on ajoute est bien une ligne de train. Puis, on ajoute la ligne à la liste. On ajoute aussi le flux correspondant si la destination de la ligne est le terminal courant.

AeroportRegional

Tout comme Gare, elle hérite de Terminal et redéfini ajoutLigne, mais vérifie que c'est une ligne d'avion et vérifie que la liste ne soit jamais plus de taille 2.

AeroportInternational

Cette classe est exactement comme AeroportRegional sauf qu'elle peut avoir jusqu'à 4 lignes d'avion.

HubAeroport

Cette classe est exactement comme AeroportInternational mais avec la possibilité de contenir 12 lignes.

HubMultimodal

Cette classe est comme HubAeroport avec en plus une relation de composition avec une gare. En effet, si le HubAeroport est détruit alors la gare aussi.

Ligne

C'est une classe template qui prend en argument un Moyens (de transport) : Une sorte d'avion ou un train. Cette classe contient une référence du moyen de transport de la ligne. Cette référence nous sert à utiliser la fonction qui va nous donner le type du moyen de transport. Il nous donnera aussi sa capacité, ce qui va nous servir à calculer la fréquence.

Moyens

C'est la classe mère des transports. Elle a des attributs qui seront utilisés par ses filles comme la vitesse, l'empreinteCarbone, la capacité et le nom.

Elle possède 2 méthodes virtuelles : affiche qui est définie dans chaque classe fille pour afficher différentes valeurs et la méthode gettype qui donne le type transport sous la forme d'un string. Elle est redéfinie dans chaque classe fille pour qu'elle renvoie le bon type.

Voyage

Cette classe permet d'utiliser les classes précédentes car elle permet de se balader d'un terminal à un autre en utilisant un type de ligne en connaissant le terminal de départ et celui de la destination. Puis il ajoute des correspondances jusqu'à atteindre la destination en prenant en compte le flux et l'émission de carbone.

2 Partie C

Une les partie A et B terminer nous avons convenue de l'utilité d'une classe qui permettrai de mettre en scène les différents éléments implémenté dans les parties précédentes afin de répondre aux problématique de la dernière partie de ce devoir. C'est le rôle de la classe Scenario.
Cette classe est déclarée dans le header associé, Scenario.h

2.1 Scenario : instanciation

La classe Scenario s'instancie avec un entier. Il s'agit de l'entier correspondant aux environnement proposés dans la partir C de l'énoncé.
Cette entier peut prendre les valeur de 1 à 6.

2.2 Les énumérations ville_e et typeLien_e

Des énumerations on été défini afin de simplifier la lecture du code.

ville_e Cette énumération correspond aux différentes ville évoqué par l'énoncé, les alias de cette énumération prennent des valeur spécifique de manière à pouvoir les utilisé comme arguments pour l'opérateur [] des attribut M, flux et Term.

typeLien_e Cette énumération permet d'indiquer le type de lien reliant deux ville dans la matrice d'adjacence M.

- TRAIN correspond à une liaison ferroviaire entre deux gares
- AVION correspond à une liaison aérienne assuré par un avion classique.
- AVIONELECTRIQUE correspond à une liaison aérienne assuré par un avion élèctrique.

2.3 Scenario : les attributs

M : Il s'agit d'une sorte de matrice d'adjacence du graphe des liaison entre les villes. Une sorte car ce n'est pas une matrice binaire. En effet chaque coefficient renseigne également le type de liaison reliant deux ville.
Le bout de code suivant créer un lien de villeA ver villeB dans ce sens, avec le type de lien typeDeLien :

```
M[villeA][villeB] = typeDeLien;
```

villeA et villeB peuvent être des entiers de 0 à 4 ou bien l'une des valeurs de l'énumération ville_e.

Term : Il s'agit d'un tableau de Terminal, il regroupe les différents terminaux associé à chaque ville.
On peut récuparer le terminal associé à une ville grâce aux valeur de l'énumération ville_e.

La liste l : Il s'agit de la liste regroupant l'ensemble des lien instancié par l'instance courante de Scenario. Elle permet normalement de libérer la mémoire dynamiquement allouer.

La matrice v Il s'agit de la matrice des Voyage d'une villeA à une ville B. Elle fonctionnent de la même manière que la matrice M.

```
v[villeA][villeB]
```

L'exemple ci-dessus correspond donc au pointeur vers le voyage allant de la villeA à la villeB dans ce sens.

Les attribut static ville et typeLien : Ces attribut permettent d'instancié de manière global pour la classe Scenario des instances des énumération ville_e et typeLien_e.

L'attribut static flux Il s'agit de la matrice des flux proposé par la partie C de l'énoncé.

2.4 Scenario : les méthodes

ajouterLienMatrice Cette méthode ajoute un lien (dont on peut précisé le type) bidirectionnelle entre deux villes.

buildVoyage Cette mathode prend l'entier correspondant aux scénario courant (le même que celui indiquer aux constructeur lors de l'instanciation de la classe).
Cette méthode à pour rôle d'allouer chaque Voyage de la matrice v et de construire les trajets pour chacun d'entre eux en fonction du scénario considéré.

ajouterLigneVoyage Ajoute de manière bidirectionnelle une correspondance pour les voyage de villeA à villeB.

```
ajouterLigneVoyage(OriginA, OriginB, CorrespA, CorrespB
```

Le bout de code ci-dessus ajoute corespondance de CorrespA à CorrespB pour le voyage de villeA à villeB et une correspondance de CorrespB à CorrespA pour le voyage de villeB à villeA.
On parle des voyage dans la matrice v.

afficherFlux Cette méthode affiche sur la sortie standard la matrice flux ;

afficherMatrice Cette méthode affiche sur la sortie standard la matrice M

affiche Cette méthode affiche sur la sortie standard une vu de l'instance courante de la classe Scenario.

3 Compilation

Une fois l'archive fourni décompressée, il faut ouvrir un terminal, se placer dans le dossier source de l'archive décompressé et taper la commande "make". Un éxécutable sera alors généré dont le nom sera "TP6-DM_CANO_COUPPOUSSAMY.exe"