09/10/2019

Compte rendu TP

Catalogue de trajets

Nel BOUVIER & Loïc DUBOIS-TERMOZ – 3IFA

Description des classes de l'application

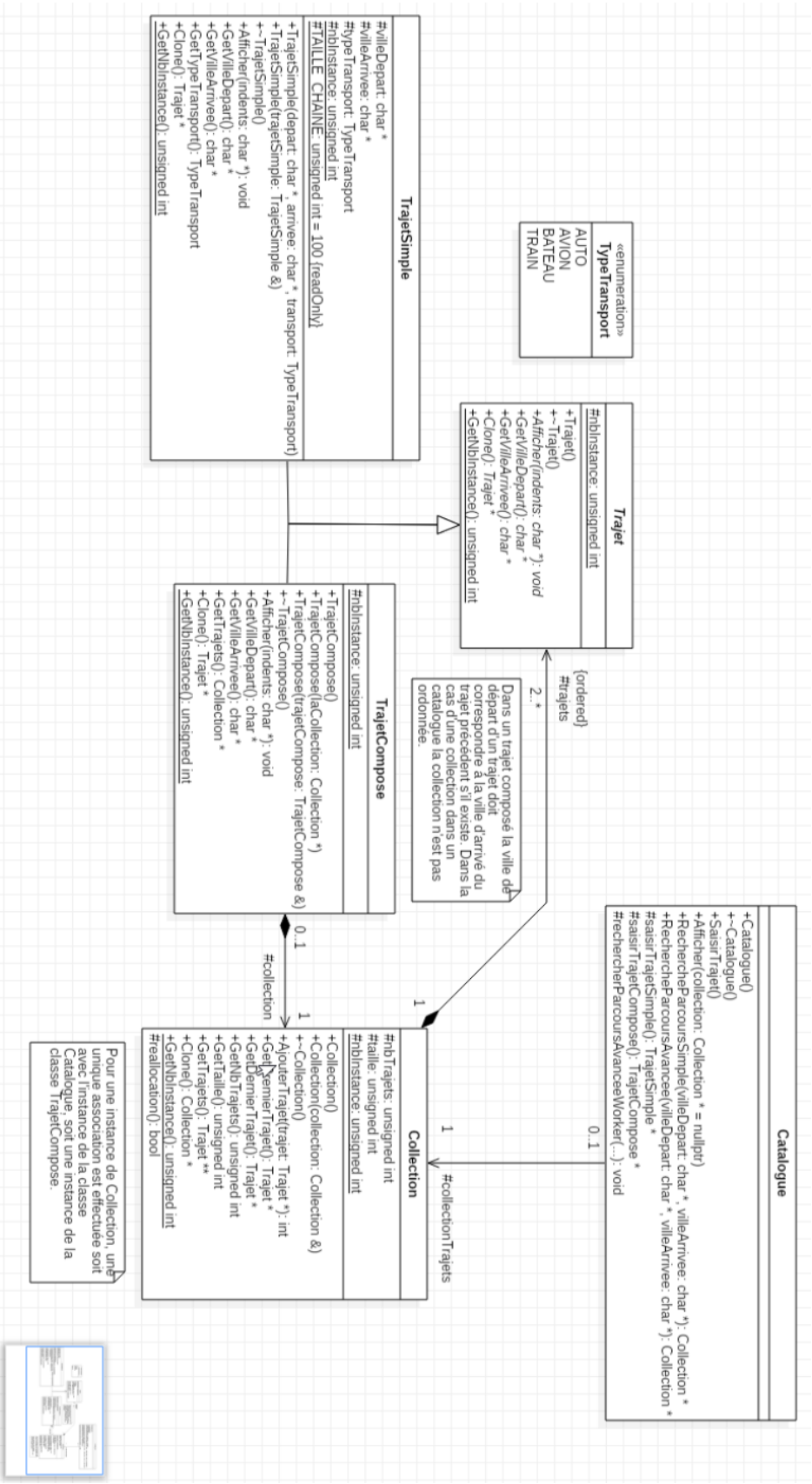


Figure 1 - Diagramme de classe

Un trajet pouvant correspondre à un trajet simple ou à un trajet composé, l'emploi de l'héritage se justifie. Par conséquent nous avons décidé de créer une classe abstraite « Trajet » héritée des classes « TrajetSimple » (caratérisée par une ville de départ, une ville d'arrivée et un type de transport) et « TrajetCompose » (caractérisée par une collection de trajets). Ainsi on voit apparaître la structure du pattern Composite avec une composition de trajets « Trajet » pour un trajet composé.

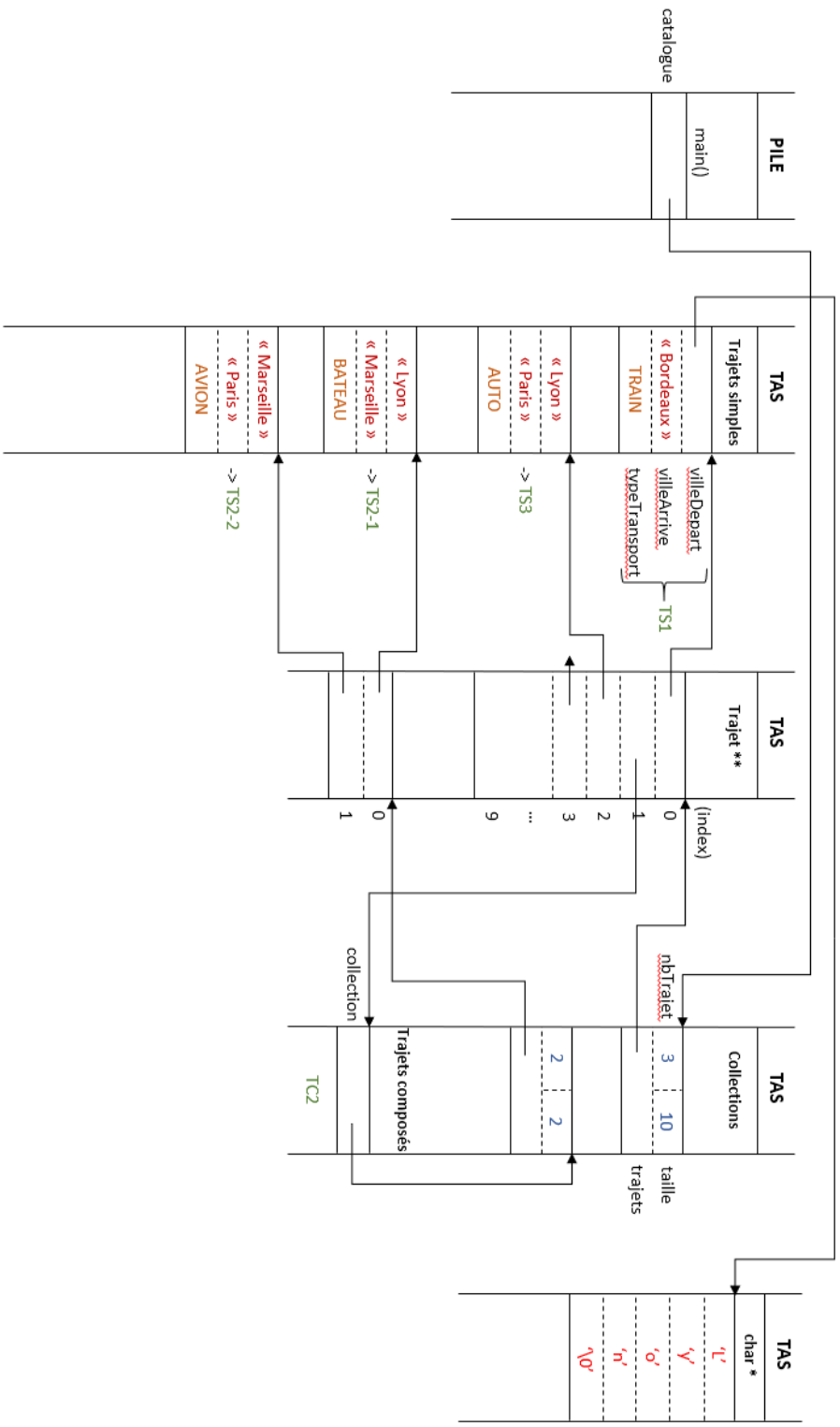
Pour la gestion de la collection de trajets, nous avons décidé d'ajouter une classe « Collection » qui correspond à la structure de données se trouvant au cœur du fonctionnement de l'application, c'est-à-dire au niveau des trajets composés et du catalogue. Cette structure repose sur un tableau dynamique qui est associé à une taille et un nombre d'éléments le constituant (à savoir que la taille du tableau est toujours supérieure ou égale au nombre d'éléments). Elle se charge notamment de l'ajout de nouveaux éléments en conservant la cohérence entre les attributs et effectuant une réallocation de la mémoire si le nombre d'éléments de la collection vient à dépasser la taille de la collection pour augmenter cette dernière.

Enfin, pour le stockage de la collection de l'ensemble des trajets du catalogue et la définition des actions devant être rendues par l'application (ajout de trajets, affichage et recherche de parcours), nous avons ajouté une classe « Catalogue » qui a un rôle proche d'un contrôleur entre la partie modèle et la partie graphique.

Il est important de noter (comme indiqué sur le diagramme) qu'une instance de la classe « Collection » est soit associée à l'instance de la classe « Catalogue », soit à une instance de la classe « TrajetCompose ». De même, si et seulement si la collection est associée à un trajet composé, les trajets doivent être ordonnés tels que la ville d'arrivée d'un trajet doit obligatoirement correspondre à la ville de départ du trajet suivant. Enfin, pour une question de cohérence nous avons décidé d'imposer le nombre minimum de trajets dans un trajet composé à 2.

Les recherches de parcours pouvant nécessiter la duplication d'objets, nous avons rajouté un système de clonage pour l'ensemble des classes excepté pour le catalogue (cette dernière ne possède qu'une unique instance durant toute la durée d'exécution de l'application). Ceci s'est avéré être la meilleure solution puisqu'il n'est pas possible de rendre virtuel un constructeur de copie dans le cas où l'on souhaiterait dupliquer un trajet de type « Trajet » (d'où la présence de la méthode Clone() qui retourne directement l'objet dupliqué).

Description de la structure de données utilisée



Lors du lancement de l'exécution de l'application, l'objet de la classe « Catalogue » est instancié et stocké dans la pile en créant également un objet « Collection » dynamiquement dont la valeur de l'adresse est stockée dans le catalogue. Dans cet objet collection est créée dynamiquement un tableau de pointeurs sur des types « Trajet » dont la taille est celle stockée dans la collection, à savoir 10 qui est la taille par défaut d'une collection. Suite à quelques ajouts de trajets dans le catalogue, la collection pointée par le catalogue se retrouve constituée de 3 trajets : 2 trajets simples et 1 trajet composé. Pour chaque trajet simple pointé par le tableau de pointeurs sur des types « Trajet » sont stockés dynamiquement 2 tableaux de caractères, l'un pour le nom de la ville de départ, l'autre pour la ville d'arrivée et une valeur entière représentant le type de transport utilisé. Pour les trajets composés, un simple pointeur sur une collection de trajets est stocké. Dans le cas de cette représentation la collection contient un pointeur sur un tableau de pointeurs de trajets d'une taille de 2 et comportant 2 trajets simples.

Problèmes rencontrés et axes d'évolution et d'amélioration de la réalisation

Actuellement, l'application ne garantie pas un fonctionnement correct en cas d'erreur de saisie de l'utilisateur : saisie d'une chaîne de caractères au lieu d'un nombre devant correspondre au type de trajet à saisir, mauvais numéro de trajet spécifié, etc. Pour cela il aurait été nécessaire de rajouter des vérifications de saisies plus avancées que celles présentes actuellement. Mieux encore : l'implémentation d'une interface graphique aurait été idéale.

L'application permet également de saisir autant de fois que voulu des trajets totalement identiques (même ville de départ, d'arrivée et même type de transport pour un trajet simple par exemple). Pour des questions de cohérence il aurait été bon de bloquer cette possibilité.

Le stockage des informations se faisant directement dans la mémoire vive, celles-ci sont perdues dès lorsque l'application est arrêtée. L'emploi d'une base de données ou d'un format de données plus généralement couplé avec un système de sauvegarde peut être un axe de développement à considérer.

Il pourrait être également intéressant de rajouter la possibilité de modifier / supprimer des trajets si on décide d'implémenter le système de sauvegarde évoqué précédemment.

Un autre point à prendre en compte : l'application n'est pas « Thread-Safe ». Un autre axe d'évolution pourrait donc se porter sur la partie multi-threading de l'application.

Une autre utilisation possible pourrait être au niveau réseau en découpant l'application en deux parties : une partie Back-End sur un serveur distant et une partie Front-End.

Enfin, il est fort probable que l'application comporte des failles de sécurité type « Buffer Overflow » ou autre. Il serait donc également appréciable de travailler sur cet axe en utilisant par exemple des fonctions plus sécurisées (utiliser la fonction strncpy à la place de strcpy peut être un début par exemple même s'il en existe d'autres encore plus sécurisées).

Même si actuellement l'application reste petite et consomme peu de ressource, l'axe d'évolution concernant l'optimisation des performances et de la mémoire n'est pas à oublier pour autant.