Auteurs : Clélie JALBERT et Cyprien BOUVIER
Groupe : B3205

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

Compte rendu de TP n°2 : Héritage, Polymorphisme - Optimaps

I. Introduction

Le but de ce TP est de réaliser une application en C++ mettant en avant l'approche orientée objet. L'application doit permettre à l'utilisateur de saisir des trajets et de rechercher des trajets possibles.

- Les trajets sont soit un couple (Départ – Destination) avec un moyen de transport,
- soit une liste de trajets (Départ – Correspondance(s) – Destination).

Tous les trajets sont compilés dans une collection qui gère la recherche et la création des trajets.

II. Structure du projet

Pour réaliser ce logiciel, nous avons essayé de mettre autant que possible en avant la philosophie de la programmation orientée objet. Par exemple :

- Un **Noeud** est un élément d'une liste chaînée. Il possède des méthodes virtuelles telles que `Affiche()`, `Depart()`, … Il possède également une méthode `Poids` qui permet à la recherche de trajets de choisir le trajet le plus simple (un trajet composé ayant un poids plus élevé).
- Ensuite, presque toutes les classes héritent du **Noeud**, par exemple :
  - la classe **Trajet**, qui stocke simplement une ville de départ, une ville d'arrivée et le moyen de transport ;
  - la classe **ListeChainee**, qui est également un Noeud. Elle gère les trajets composés. Ainsi, dans notre `Catalogue`, chaque entrée peut être soit une **ListeChainee** (dans le cas d'un trajet composé), soit un simple trajet.
- Puisque le **Catalogue** est une liste de trajets, nous avons finalement décidé de le faire hériter de **ListeChainee**, car le catalogue n'est en réalité qu'une implémentation spécialisée d'une liste chaînée. Il sert de collection de trajets afin de permettre les recherches.
- La **Pile** est une pile de pointeurs vers des **Noeud**, utilisée en interne pour la recherche de trajets en profondeur.
- Le type **Ville** représente une chaîne de caractères allouée et gérée par un **Noeud**.

III. Pistes d'amélioration

- Utilisation de dijkstra
- Filtrer la recherche pour un mode de transport précis
- Utilisation de la STL et de templates pour les types de ListeChainee et Pile plutôt que l'approche orienté objet pure.
- Extraction du code de main dans une classe d'interface IHM.

IV. Annexes

Voici le graphe des classes et de leurs relations :

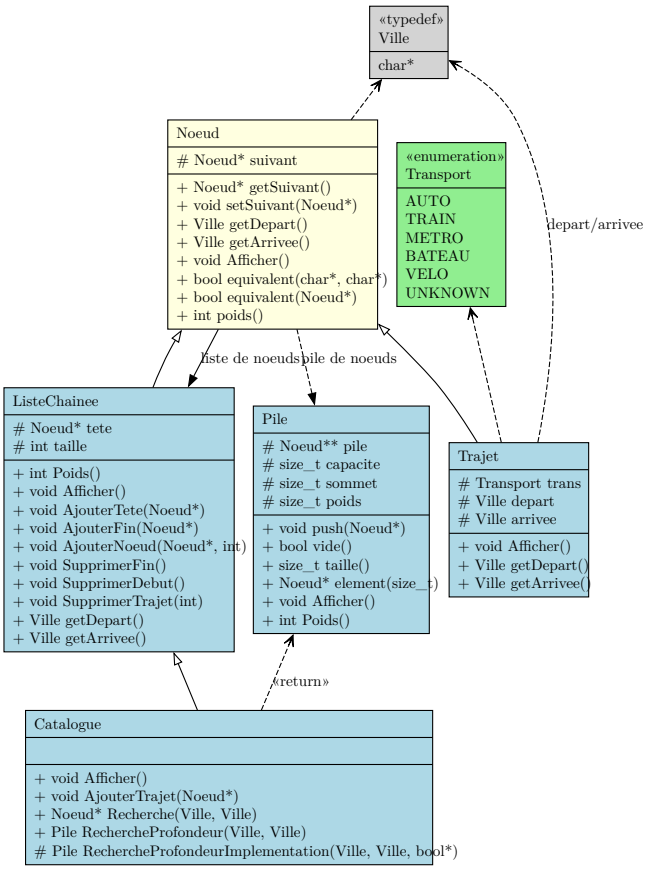

Figure 1 : ***« Diagramme de tout les objets et comment ils intéragissent entre-eux »***

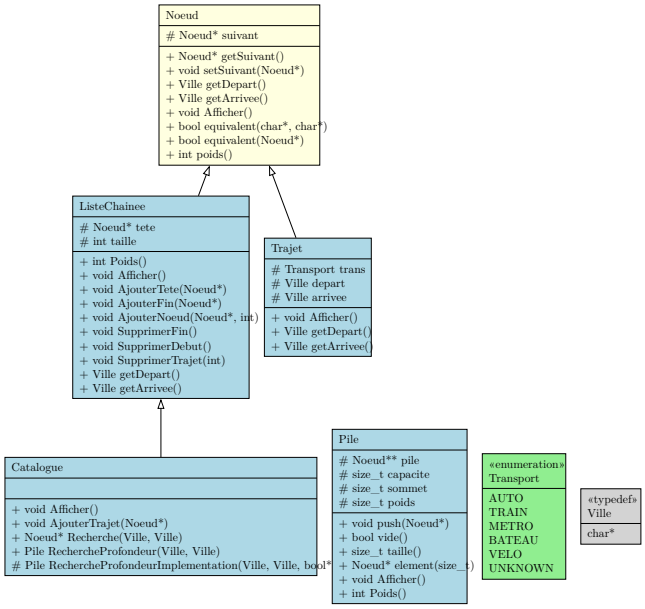

Figure 2 : ***« Diagramme de tout les objets et des héritages »***

Et voici un diagramme de la mémoire lors de l'éxécution:

Auteurs : Clélie JALBERT et Cyprien BOUVIER
Groupe : B3205

Vue simplifiée de la mémoire