Quentin DE ANDRIA / Louis ROB

Compte-rendu TP n°2 C++

I. Description générale des classes / Héritage

I.1. Classe Trajet

Un trajet est composé d'un point de départ (start) et d'une arrivée (end).

Cette classe est abstraite dans la mesure ou un trajet devra être soit simple soit composé (mais pas juste un trajet tout court). Les attributs et méthodes implémentés ici seront utilisés par toutes les classes héritières.

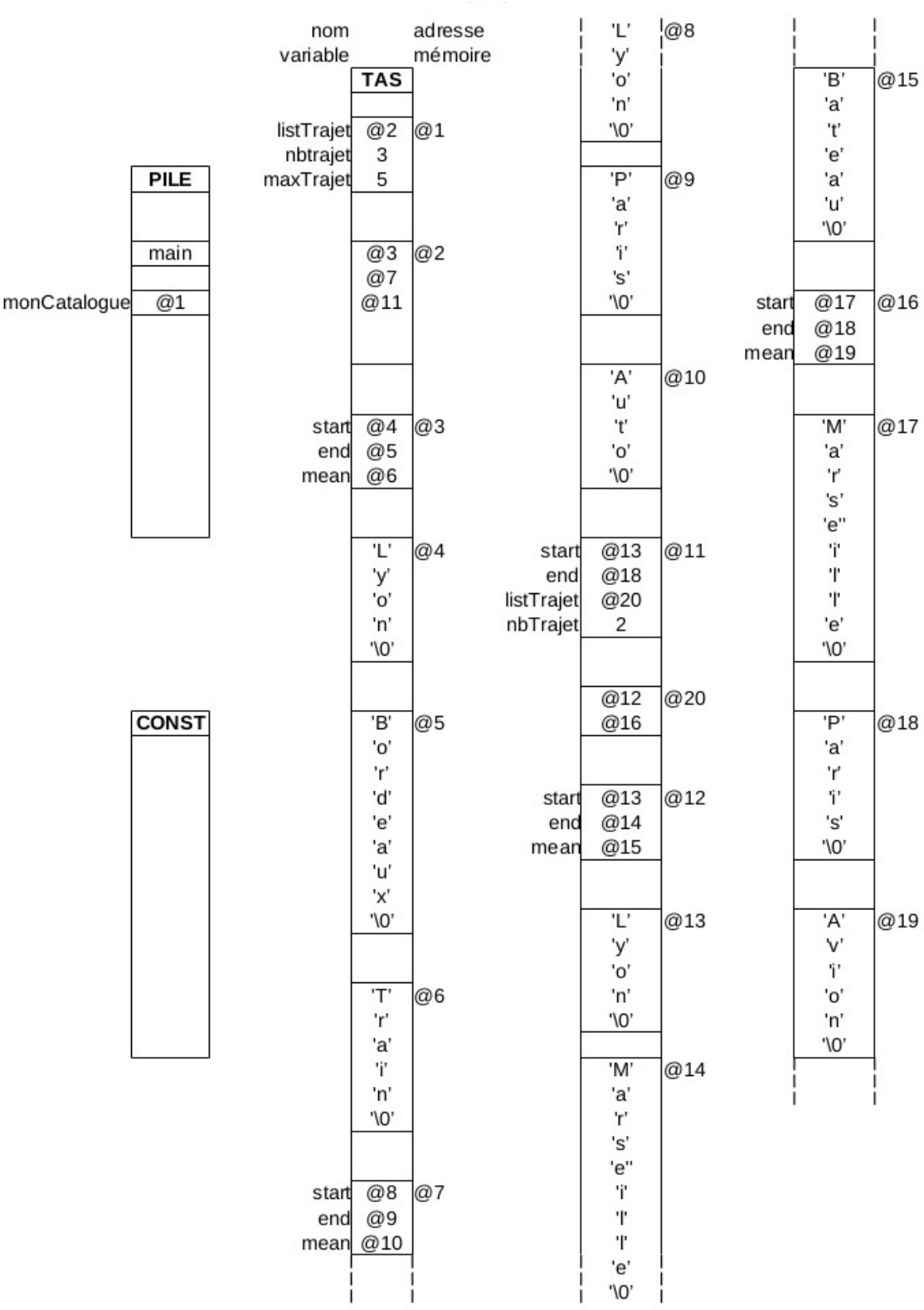
I.2. Classe TrajetSimple

Un trajet simple hérite de Trajet en spécifiant en plus le moyen de transport. En effet un trajet simple ne sera composé que de deux lieux (départ et arrivé, contrairement à un trajet composé).

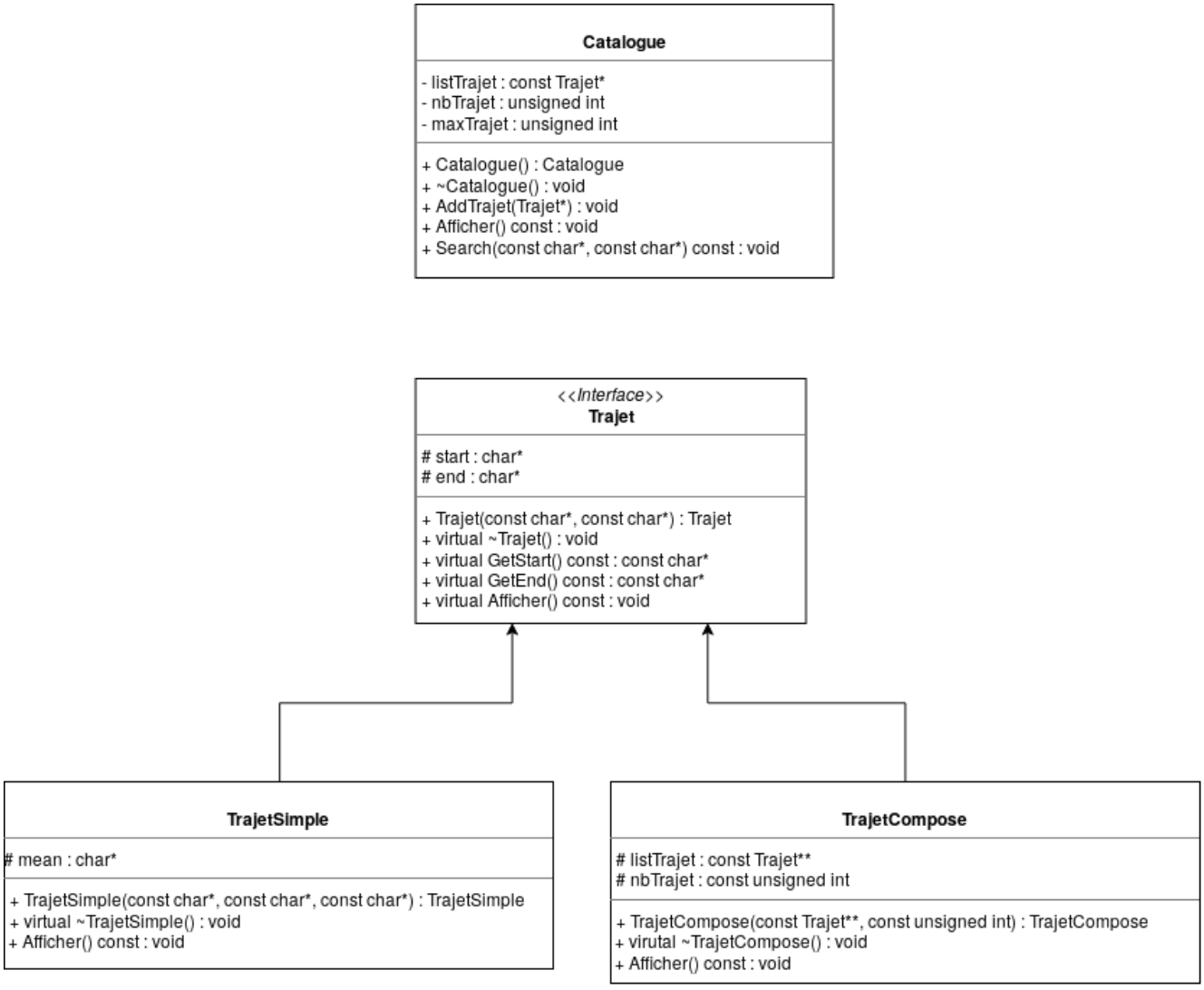
I.3. Classe TrajetComposé

Un trajet composé hérite de Trajet. Il s'agit en fait d'une liste de trajet simples. Le fait d'avoir un telle organisation des classes permet à chaque trajet simple d'un trajet composé de disposer de son propre moyen de transport.

II. Dessin de la structure de donnée (exemple concret)



III. UML


Catalogue
- listTrajet : const Trajet*
- nbTrajet : unsigned int
- maxTrajet : unsigned int
+ Catalogue() : Catalogue
+ ~Catalogue() : void
+ AddTrajet(Trajet*) : void
+ Afficher() const : void
+ Search(const char*, const char*) const : void
<<Interface>>
Trajet
start : char*
end : char*
+ Trajet(const char*, const char*) : Trajet
+ virtual ~Trajet() : void
+ virtual GetStart() const : const char*
+ virtual GetEnd() const : const char*
+ virtual Afficher() const : void
TrajetSimple
mean : char*
+ TrajetSimple(const char*, const char*, const char*) : TrajetSimple
+ virtual ~TrajetSimple() : void
+ Afficher() const : void
TrajetCompose
listTrajet : const Trajet**
nbTrajet : const unsigned int
+ TrajetCompose(const Trajet**, const unsigned int) : TrajetCompose
+ virutal ~TrajetCompose() : void
+ Afficher() const : void

IV. Listing des interfaces et réalisations

- IU : demo.cpp
- Catalogue : Catalogue.h / Catalogue.cpp
- Trajet : Trajet.h / Trajet.cpp
- Trajet simple : TrajetSimple.h / TrajetSimple.cpp
- Trajet composé : TrajetCompose.h / TrajetCompose.cpp

V. Problèmes rencontrés

Lors de l'écriture de l'UML nous avons déclaré des const partout où il nous le semblait nécessaire. Cependant, nous nous sommes rabattus sur l'idée d'en mettre moins car nous pensions que certaines variables ne devaient pas l'être. Mais finalement, l'UML était correct : il a donc fallu replacer certains const. Nous avons donc appris que des variables dynamiques pouvaient être déclarés const.

Concernant la fuite de mémoire, l'utilisation de Valgrind nous a permis de remplacer les delete par delete[] pour désallouer un tableau ainsi que de mettre en évidence que nous avions oublié d'implémenter le destructeur de trajet composé en entier.

V. Axes d'évolution

Une première possible serait d'ajouter un service au sein de notre classe Catalogue qui permettrait de supprimer un trajet.

Deuxièmement, nous pourrions effectuer une vérification de jointure entre deux trajets consécutifs d'un trajet composé.

Aussi, nous n'avons pas eu besoin d'implémenter les constructeurs de copie ainsi que l'opérateur d'affectation pour les classes ayant des attributs alloués dynamiquement (Catalogue, Trajet, TrajetSimple et TrajetCompose) : il serait préférable de les coder avant de distribuer le produit.

En dernier lieu, il serait utile de pouvoir créer un nouveau trajet composé à partir d'autres trajets composés.