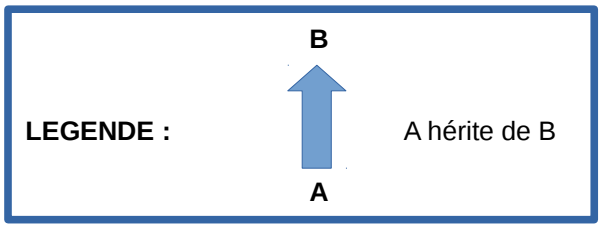
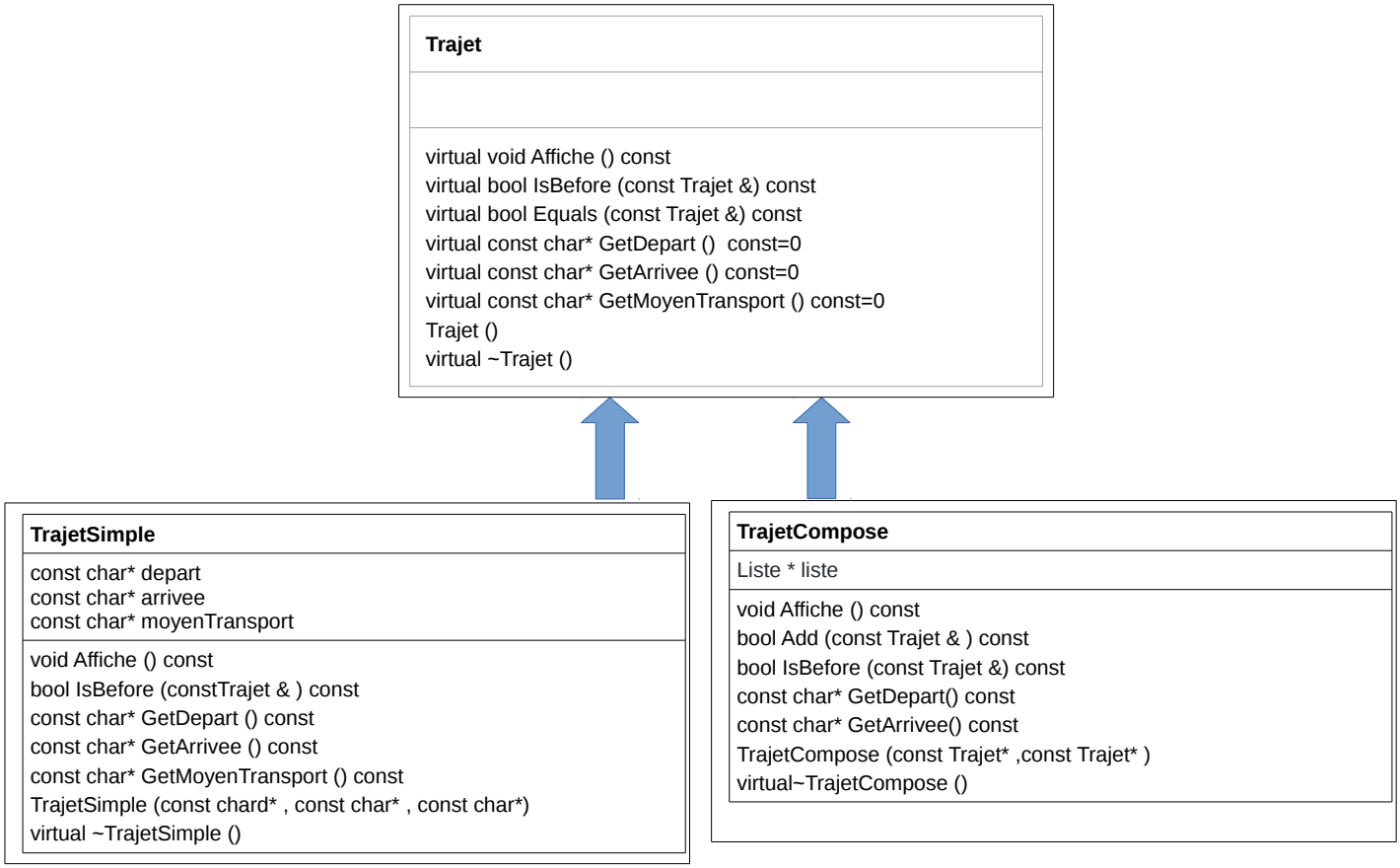
Florian MUTIN 3IF4 binôme B3414
Loïc CASTELLON

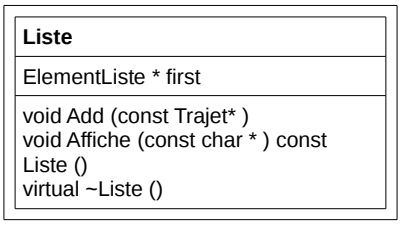
Compte rendu TP C++ n°2 : Héritage - Polymorphisme

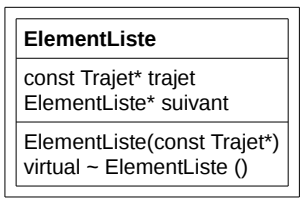
SOMMAIRE :

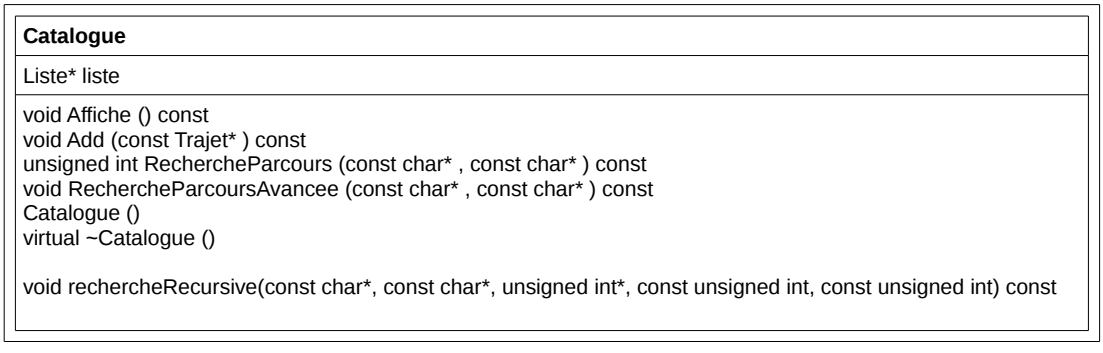
I) Description des classes – graphe d'héritage











- Nous avons choisi de ne mettre aucun attribut dans la classe Trajet car les départs et arrivées se retrouvent à partir des TrajetSimples.
- Les attributs de la classe TrajetSimple sont en private, on peut y accéder uniquement en lecture grâce à des méthodes Get spécifiques.
- Les classes TrajetCompose et Catalogue reposent fortement sur la classe Liste, cela nous permet d'éviter une redondance de code entre ces deux premières.
- Les classes Liste et ElementListe sont expliquées ci-dessous.

II) Description de la structure de donnée – dessin mémoire

Un catalogue possède un pointeur sur une Liste. Liste est une classe qui implémente une liste chaînée d'ElementListe, quand on ajoute un ElementListe il est inséré en fin de liste. Un ElementListe possède un pointeur sur un Trajet et un pointeur sur un ElementListe suivant. Un TrajetCompose possède un pointeur sur une Liste , il fonctionne de la même manière.

Le dessin mémoire donné en annexe illustre cela.

III) Conclusion

Nous avons rencontré plusieurs problèmes lors de ce projet. Le premier est que nous avions au départ mal implémenté la façon de faire une Liste, nous avions essayer de gérer ça avec de l'héritage entre une classe Liste et les classes de Trajets. Nous nous sommes ensuite souvenu qu'il fallait séparer tout cela et que de l'héritage entre les deux n'était pas la bonne solution.

Ensuite, nous avons eu du mal à implémenter la méthode RechercheParcoursAvancee. Ce n'était pas tant le fait de combiner les trajets qui nous a paru difficile mais plutôt de trouver comment cette méthode pouvait afficher tous les trajets possibles. Pour résoudre cela nous avons dû faire une recherche récursive avec parcours en profondeur de la Liste du catalogue.

Le constructeur de TrajetCompose n'est pas optimal lorsque l'on veut créer un TrajetCompose dans notre menu. Nous nous en sommes rendu compte sur le fait. Nous avons décidé d'assumer notre choix de constructeur et d'adapter notre menu.

La méthode IsBefore nous a servi en début de projet pour assurer la continuité des trajets dans un TrajetCompose. Cependant notre menu assure la continuité. Cette méthode est maintenant obsolète mais nous l'avons conservée pour effectuer des tests et afficher d'éventuelles erreurs.

Quand on renseigne les attributs de Trajet ils sont transformés en majuscules. Cela nous permet d'assurer qu'une ville ne sera pas écrite en minuscule et en majuscule. Mais cela ne fonctionne pas pour les caractères spéciaux tels les accents ou la cédille. Cela pourrait donc faire l'objet d'une amélioration.

De plus quand on ajoute un TrajetSimple on vérifie s'il existe mais nous ne vérifions pas cela pour l'ajout de TrajetComposé. Cela n'a pas été implémenté par manque de temps mais peut faire l'objet d'une amélioration.

Dans notre menu, si l'utilisateur saisit autre chose qu'un entier le programme se ferme. De la même manière s'il saisit autre chose qu'un entier lors de l'ajout d'étapes pour un TrajetComposé le programme entre dans une boucle infinie. Ces constatations rendent l'expérience utilisateur difficile. Le menu est donc améliorable.

IV) Annexes : Dessin mémoire et listing des classes