

TD-TP : LIENS ENTRE CLASSES

MISE EN PLACE

1. Deux classes

Etant données la classe `Voiture` dont les objets sont caractérisés par une marque et une plaque d'immatriculation, **et** la classe `Individu` dont les objets sont caractérisés par un nom et un prénom, chaque classe ayant un constructeur.

Travail à faire

1.a Ecrire pour chacune de ces 2 classes une méthode `string toString()` qui retourne la description de l'objet concerné,

Et écrire un `main()` qui :

1.b Crée une `Voiture` `voit1` «RenaultClio, 123AB64», une `voit2` «Peugeot106, 678CD96», `voit3` «CitroenPicasso, 456EF75» et les `Individu` `ind1` «Dupond, Pierre», `ind2` «Martin, Louis» et `ind3` «Durand, Marcel».

1.c Affiche les attributs de chacun des objets créés en utilisant les méthodes `toString()` respectives.

RELATION SIMPLE ENTRE DEUX CLASSES

Tout `Individu` peut avoir au plus une `Voiture` et une `Voiture` peut avoir zéro ou un `Individu` comme pilote.

Travail à faire : Représenter cette relation avec un schéma UML.

*Note : Lorsqu'il existe une relation entre deux classes **A** et **B**, il faut se poser des questions de conception (cf. questions conceptuelles), quant à la **navigabilité entre les objets**. A savoir :*

- *Un objet de **A** doit-il connaître l'objet de **B** avec lequel il est en relation ?*

et réciproquement

- *Un objet de **B** doit-il connaître l'objet de **A** avec lequel il est en relation ?*

Les réponses à ces deux questions guideront quant à l'implémentation la plus adaptée.

☹ *Quelle que soit la réponse à ces questions, en C++ une mauvaise implémentation consiste à déclarer dans la classe **A**, un attribut `monPoteB` qui serait du type classe **B**.*

En effet, l'attribut `monPoteB` sera initialisé aux mêmes valeurs que celles contenues par l'objet avec qui se lie, mais `monPoteB` ne sera pas impacté par les changements de valeurs que subira l'objet lié.

☺ *Une bonne implémentation consiste à déclarer dans la classe **A**, un attribut `monPoteB` qui serait du type pointeur de **B**.*

En effet, l'attribut `monPoteB` sera initialisé par l'adresse de l'objet avec qui se lie, `monPoteB` permettra d'accéder à l'objet lié et à ses attributs qui enregistreront les éventuels changements.

2. Implémenter une bonne relation entre objets

Travail à faire

- 2.a Modifier le code des classes `Voiture` et `Individu` en intégrant respectivement l'attribut `Individu* monPilote;` dans la classe `Voiture` (cf. `Voiture.h`) et l'attribut `Voiture* maVoiture;` dans la classe `Individu` (cf. `Individu.h`).
- 2.b Coder dans le main le fait que la `Voiture` `voit3` et l'`Individu` `ind3` sont en relation.
- 2.c Afficher la plaque d'immatriculation de la voiture que conduit `ind3`.
- 2.d Changer la plaque d'immatriculation de la `voit3` en lui attribuant la valeur « 77777NO22 ».
- 2.e Afficher la plaque d'immatriculation de la voiture que conduit `ind3`. Aura-t-elle changé ?

3. Afficher les attributs de l'objet avec lequel on est en relation

Nous envisageons une méthode `string toStringAndLink()` pour chacune des deux classes, qui retourne la description complète de l'objet concerné.

Travail à faire

- 3.a Dans le cas de la classe `Individu`, en plus de la description de l'individu, si ce dernier possède une voiture, `toStringAndLink()` retourne les attributs de la voiture en question.
- 3.b Dans le cas de la classe `Voiture`, en plus de la description de la voiture, si cette dernière est pilotée par un individu, `toStringAndLink()` retourne les attributs de l'individu en question.
- 3.c Dans le main afficher la description complète de `ind3` et `voit3`, qui sont liés entre eux, ainsi que la description complète de `ind2` et `voit2` qui eux, ne sont pas liés.

4. Assurer la symétrie de la relation

Si une relation entre une classeA et une classeB est réciproque (cf. navigable dans les 2 sens), on peut considérer que :

1. **dès lors qu'un lien est établi** d'un objet a vers un objet b,
2. alors **le lien réciproque** de b vers a **est également à établir**

Dans la version de programme C++ réalisée jusqu'ici, le pointage réciproque automatique n'est pas implémenté.

Si on souhaite implémenter, un lien réciproque entre a et b, il faut préalablement s'assurer que a (respectivement b) n'est pas déjà lié avec un objet x (respectivement y). En effet, si un des objets est lié, il faut préalablement le délier de son correspondant.

Pour cela...

... Travail à faire

- 4.a** Ecrire le code la méthode `majMaVoiture(Voiture*)` de la classe `Individu`, qui modifie simplement l'attribut `maVoiture`. De façon symétrique, écrire la méthode `majMonPilote(Pilote*)` de la classe `Voiture`, qui modifie simplement l'attribut `monPilote`.
- 4.b** Ecrire le code de la méthode `supprimerLien()` pour un objet de la classe `Voiture` et de façon symétrique, écrire le code de la méthode `supprimerLien()` pour un objet de la classe `Individu`.
- 4.c** Ecrire le code de la méthode `setMaVoiture(Voiture*)` dans la classe `Individu` de sorte que son utilisation délie les objets éventuellement liés et assure l'initialisation du pointeur local ainsi que l'initialisation de son vis-à-vis dans l'autre objet. Idem pour la méthode `setMonPilote(Pilote*)` de la classe `Voiture`.
- 4.d** Dans le main, créer le lien entre `ind3` et `voit3` en utilisant `setMaVoiture` de la classe `Individu`, vérifier que les 2 liens sont effectifs. Puis lier `voit3` avec `ind2` en utilisant `setMonPilote` et vérifier que `ind3` n'est plus lié, alors que `voit3` et `ind2` le sont.
- 4.e** Un schéma générique pour la réciprocité des liens :
- Faire le schéma UML correspondant aux méthodes codées.
 - En quoi les méthodes codées jusqu'à présent s'apparentent au schéma général ci-dessous ?

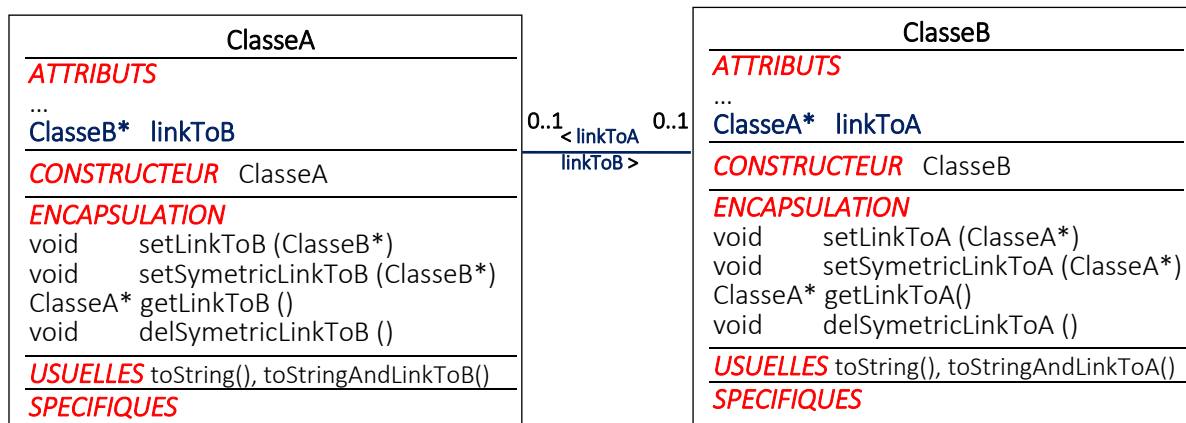


Schéma UML relatif à l'implémentation de liens symétriques

5 –Destruction d'un objet, lorsqu'il est lié

La destruction d'un objet lorsqu'il est lié à un autre objet doit supprimer le lien qui pointe sur lui.

Travail à faire

5.a Ecrire les destructeurs `~Voiture()` et `~Individu()` en conséquence.

5.b Dans le `main` afficher la description complète pour `ind2` et `voit3`, liés entre eux, détruire `voit3`, puis afficher la description complète de `ind2` pour constater que son lien vers `voit3` a été supprimé.