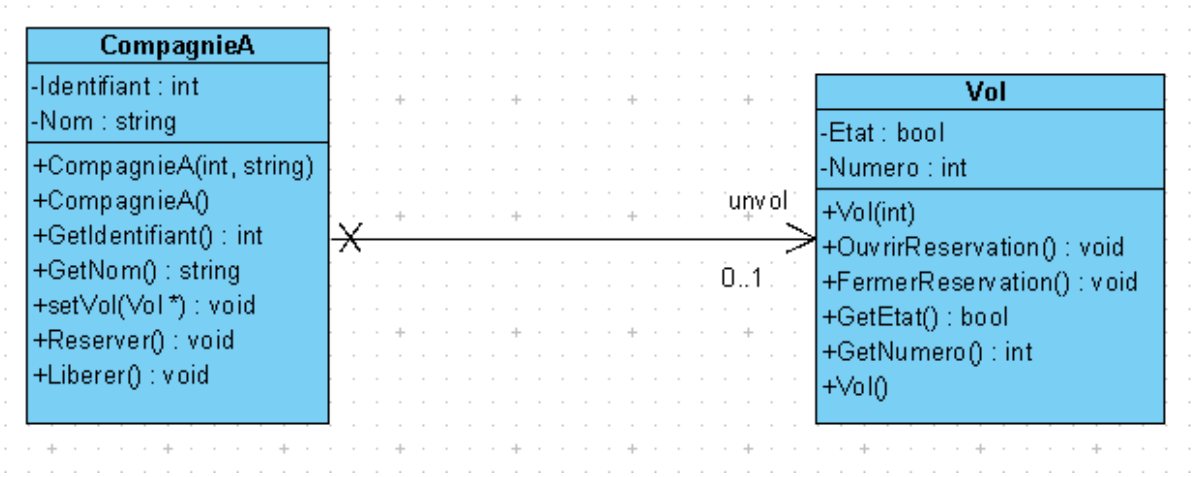
Associations UML et C++

Exercice 1

Soit le diagramme de classe suivant :



Association

Il y a association entre deux classes si une instance d'une classe (**CompagnieA**) doit connaître l'autre (**Vol**).

Navigation d'association

Les associations peuvent être dirigées ou non-dirigées. Un lien dirigé pointe sur la classe **Vol** (la cible). Une flèche de navigation sur une association montre la direction dans laquelle l'association peut être traversée ou interrogée. La flèche vous informe aussi sur le "propriétaire" de l'implémentation de l'association. Les associations sans flèche de navigation sont bidirectionnelles.

Rôles. Une association a deux extrémités. Chaque extrémité peut comporter un nom de rôle pour clarifier la nature de l'association.

Multiplicité. Une extrémité d'association peut comporter une multiplicité. La multiplicité d'une extrémité d'association est le nombre d'instances possibles de la classe associée à une instance unique de l'autre extrémité. Les multiplicités sont des nombres ou des intervalles de nombres.

En C++ une association unidirectionnelle de multiplicité 0..1 se traduit par une donnée membre qui peut être soit une référence sur l'autre classe soit un pointeur sur l'autre classe.

Cette donnée membre est soit initialisée dans le constructeur soit une méthode spécifique.

1. **Ecrire le fichier de déclaration de la classe Vol et celui de la classe CompagnieA.**

2. Coder chacune des méthodes suivantes.

Vol.cpp

- **public Vol(int _Numero){}**

Comment se nomme cette méthode ?

Cette méthode en plus d'initialiser l'attribut privé Numéro, émet un message du type : « création du vol #Numero « vers la console.

- **public void OuvrirReservation( ) {}**

Cette méthode teste la valeur du booléen privé Etat.

Si sa valeur est à false, Etat passera à true et un message " Les réservations pour le vol #Numéro sont ouvertes" sera envoyé vers la console.

Sinon elle restera à true et un message " Les réservations pour le vol #Numéro sont déjà ouvertes" sera envoyé vers la console.

- **public void FermerReservation( ){}**

Cette méthode teste la valeur du booléen privé Etat.

Si sa valeur est à true, Etat passera à false et un message " Les réservations pour le vol #Numéro sont fermées" sera envoyé vers la console.

Sinon Etat restera à false et un message " Les réservations pour le vol #Numéro sont déjà fermées" sera envoyé vers la console.

- **public boolean GetEtat( ){}**

Cette méthode renvoie la valeur du booléen Etat;

- **public int GetNumero( ){}**

Cette méthode renvoie la valeur de l'entier Numero;

CompagnieA.cpp

- **public CompagnieA(int _Id, String _Nom){}**

Comment se nomme cette méthode ?

Cette méthode en plus d'initialiser les attributs privés Identifiant et Nom émet un message du type : "Création de la Compagnie Aérienne #Nom numéro d'identification : #Identifiant " vers la console.

- **public String GetNom( ){}**

Cette méthode renvoie la valeur de l'attribut privé Nom.

- **public int GetIdentifiant( ) {}**

Cette méthode renvoie la valeur de l'attribut privé Identifiant.

- **public void setVol(Vol _vol){}**

Cette méthode met à jour l'attribut vol avec la référence de l'objet _vol.

- **public void Reserver(){}**

Cette méthode permet d'ouvir la réservation d'un vol (Vol :OuvrirReservation).

- **public void Liberer (){}**

Cette méthode permet de fermer la réservation d'un vol (Vol :FermerReservation).

Ecrire le fichier main de l'application

Main.cpp

- Créer un objet vol1 de type Vol ayant comme identifiant 1.
- Créer une compagnie aérienne CA1 ayant comme identifiant 1 et comme nom « AIR France »
- Appeler les méthodes Reserver() et Liberer().

Si l'association était bidirectionnelle

- Définir ce qui serait ajouter et dans quelle classe l'ajout aurait lieu.

Ecrire la nouvelle déclaration de la classe Vol.

Ecrire la méthode Vol ::setAffreteur(CompagnieA *)

Réécrire la méthode CompagnieA ::setVol(Vol *) afin qu'elle initialise l'association bidirectionnelle.