HLIN603
Feuille de TD N°1 : Rappel C++ et Héritage simple

Exercice 1 : Rappel C++ - Strophe et Vers

Nous reprenons l'exemple vu en cours et qui porte sur la représentation des vers et des strophes.
Dans cet exemple, un vers est essentiellement une suite de mots à laquelle on peut attacher une rime, la sonorité terminale (écrite sous une forme normalisée). Un vers peut être saisi et affiché (voir la Figure 1). Aux deux attributs *suiteMots* et rime sont associées deux opérations particulières, habituellement appelées des accesseurs car elles se spécialisent dans l'accès en lecture (ex. *getSuiteMots*) ou en écriture (ex. *setSuiteMots*).
Une strophe est une suite de vers. Dans le cas général (qui admet la poésie en vers libres) nous admettrons que l'on peut saisir une strophe par saisie du nombre de vers puis saisie successive des différents vers (dans l'ordre d'apparition dans la strophe). L'affichage d'une strophe est l'affichage de ses vers successifs.

- Donnez le code C++ correspondant au modèle de la figure 1.
- Proposez une fonction main pour tester le fonctionnement

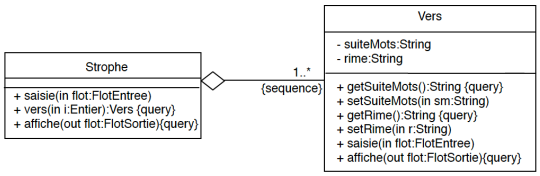

FIGURE 1 – Eléments de la représentation des strophes

Exercice 2 : Héritage simple - Objets de musée et salle d'exposition

a) Objets de musée

Un musée de campagne possède des objets variés. On veut pouvoir stocker pour chacun de ces objets un descriptif (texte libre qui peut être vu comme une chaîne de caractères), et sa référence, nombre entier qui l'identifie de façon unique.

Parmi ces objets, certains sont de véritables objets d'art signés ; pour ceux-là, on veut stocker aussi le nom de l'auteur.
Un musée de campagne possède aussi, hélas, des objets légués par d'honorables citoyens et qu'il n'a pas pu refuser : renard empaillé, portrait du généreux donateur, etc.
Pour ces objets regrettables, on veut pouvoir stocker en plus le nom du donateur et l'année du don.

1) Quelles classes imaginez-vous pour représenter tous ces objets
2) Écrivez les fichiers .h et .cc correspondants, en supposant que chacune de ces classes possède :
- Des attributs privés ;
- Un constructeur sans paramètres qui initialise tous les attributs à des valeurs par défaut ;
- Un constructeur avec paramètres qui remplit tous les attributs avec les valeurs données ;
- Des accesseurs de consultation et de modification pour chaque attribut ;
- Une méthode saisie qui saisit les valeurs des attributs sur un flot d'entrée ;
- Une méthode affiche qui affiche les valeurs des attributs sur un flot de sortie.

3) Écrivez un main qui, pour chacune des classes, crée et affiche deux objets : un directement (déclaration de variable), et l'autre à travers un pointeur.

b) Salles d'exposition

Le musée possède des salles d'exposition d'une certaine capacité (capacité = nombre maximum d'objets qu'on peut mettre en exposition dans la salle).

1) Écrivez une classe qui représente une salle, en respectant les caractéristiques suivantes :
- Une salle est créée vide, avec une certaine capacité (donnée, ou 10 par défaut) ;
- On la remplit en ajoutant un objet donné à une place donnée ; la place est représentée par un numéro ; les objets peuvent appartenir à n'importe laquelle des classes précédentes, mais on veut une seule méthode ajoute, la même quel que soit le type de l'objet ;
- On peut enlever un objet : on donne le numéro de place et on récupère l'objet ;
- On veut pouvoir afficher le contenu de la salle ;
- On veut pouvoir connaître le nombre d'objets présents dans la salle.
- Information technique : on est sûr de ne jamais faire de copie d'une salle, on demande donc que la classe ne contienne pas de constructeur par copie.

2) Faites un main pour tester le fonctionnement

Exercice 3 : Héritage simple - Compte bancaire

1) Proposez trois classes C++ représentant respectivement :

   - Une classe ***CompteBancaire*** disposant d'un attribut ***solde*** et d'un destructeur qui affiche la valeur du solde que la banque est supposée rendre au client lors de la fermeture du compte.

   - Une classe ***CompteRemunere*** représentant les comptes auxquels on sert un intérêt. En particulier, lors de la fermeture, le solde est augmenté de 10%.

   - Une classe ***CompteDepot*** représentant les comptes de dépôt classiques. Lors de la fermeture, on prélève sur ces comptes des frais de gestion de 100 euros.

2)

   - Ajoutez dans ***CompteBancaire*** une méthode ***deposer*** (avec un seul paramètre qui représente le montant déposé et dont le type de retour est Void), spécialisez-la dans le cas des ***CompteRemunere*** en ajoutant un intérêt de 1% à la somme déposée, et dans le cas des ***CompteDepot*** en retirant 1euro de frais de gestion et en ajoutant 10 euros si le dépôt est supérieur à 1000 euros.

   - Définissez un tableau ***ComptesVP*** de *N* ***CompteBancaire.*** Ajouter à ce tableau des instances de ***CompteRemunere et CompteDepot***. Effectuez un dépôt sur tous les comptes de ce tableau. Que se passe-t-il dans chacun des deux cas : méthode **deposer** virtuelle ou non ?

3) Ajoutez une classe ***CompteRemunereAvecCarteCredit.*** Lors de la fermeture de ce type de compte, on prélève des frais de gestion de la carte d'un montant de 5 euros. Ajoutez une méthode ***deposer*** avec un seul paramètre (montant déposé) et qui retourne la valeur du solde.

   - Donnez l'ordre des opérations effectuées lorsqu'un **CompteRemunereAvecCarteCredit** est fermé (l'instance est « créée » ou »"détruite" au sens de C++).
   - Reprenez le tableau ***ComptesVP*** et ajoutez, en plus, quelques comptes ***CompteRemunereAvecCarteCredit.*** Effectuez un dépôt sur tous les comptes de ce tableau. Est-ce que la méthode ***deposer*** de ***CompteRemunereAvecCarteCredit*** va être appelée ? pourquoi ?