

TP n°6

Héritage

Note : ces exercices necessitent la lecture des supports des six premiers cours que l'on peut retrouver a l'adresse <http://www.liafa.univ-paris-diderot.fr/~yunes/cours/cpp/>

Exercice 1 (Une classe d'association) Avant de commencer relisez la partie *La classe d'association* du cours 5.

Dans cet exercice on modelisera l'association de la classe `Etudiant` a la classe `Cours` par une classe d'association qu'on appelle `Association`. La classe `Cours` possede la methode private



FIGURE 1 { Classe d'association.

`void Cours::nouvelEtudiant()` qui incremente `nbetudiants`. Elle est appelee par la classe `Association` lorsqu'une nouvelle association est cree (similaire pour la classe `Etudiant`).

`Association` a les membres `Etudiant* petudiant` et `Cours* pcours` qui represente cette association. Creez le constructeur `Association::Association (Etudiant *petudiant, Cours *pcours)`. Le constructeur va *notifier* `petudiant` et `pcours` qu'une nouvelle association a ete cree en appelant leurs methodes `nouvelEtudiant()` et `nouveauCours()`. Pensez au concept *friend* lorsque vous appelez ces fonctions.

Pour tester votre programme, creez un tableau d'etudiants et un tableau de cours. Creez quelques associations et verifiez le bon fonctionnement des compteurs `nbours` et `nbetudiants`.

Exercice 2 (Contrôle d'accès)

Soient les classes suivantes :

```
class X
{
    int priv;
public:
    int pub;
    void m();
};

class Y : public X
{
    void my();
};
```

Est-ce que le code suivant est correct ?

```
1. void X::m()
{
    priv = 1;
    pub = 3;
}
```

```
2. void Y::my()
{
    priv=1;
    pub=3;
}
```

```
3. void f(Y * p)
{
    p->priv=1;
    p->pub=3;
}
```

Exercice 3 (Heritage, fonctions virtuelles)

1. Creez une classe de base `Article` qui contient 2 champs, son nom et son prix, ainsi que les accesseurs idoines.
2. Creez ensuite la classe `ArticleEnSolde` qui herite d'`Article` et qui contient en plus un champ `remise`. `getPrix` devra renvoyer le prix en tenant compte de la remise.
3. Finalement, Creez la classe `Caddy` qui aura pour vocation de gerer une collection d'`Articles`. Définissez entre autre la fonction `prixTotal()` qui renverra la somme des prix des articles du caddy.

Exercice 4 (Classes abstraites, fonctions virtuelles). On se propose de creer une classe permettant de gerer des listes de n'importe quoi.

Creez la classe abstraite `donnees` contenant un pointeur vers la donnee suivante, un entier qui representera un codage quelconque de la donnee codee, une fonction d'ajout, et une fonction de comparaison de l'objet courant avec une autre donnee (en utilisant l'int).

Creez la classe `ListeDeNimp` qui contiendra un pointeur vers la tête, une fonction permettant d'ajouter la liste, ainsi que la fonction d'ajout d'un nouvel element.

Creez enfin les classes `torchon` et `serviette` heritant de `donnees`. Testez vos classes.