Université Paris Diderot **Langages à objets avancés**
M1 **Année 2015-2016**

TP n° 6 : Héritage

Exercice 1 (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;
}
```

Héritage, méthodes virtuelles, méthodes non virtuelles

Exercice 2

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

Exercice 3 On reprend nos classes `Article` et `ArticleEnSolde`. On suppose maintenant que les articles ont une date de péremption (notée en nombre de jours depuis une date fixe).

On veut maintenant simuler un magasin qui va acheter des articles à un fournisseur, les revendre à des clients et les solder éventuellement, les articles achetés et pas encore vendus sont gardés en stock.

Pour éviter d'avoir à écrire les classes `Fournisseur` et `Client` - qui ne nous intéresse pas ici - on va simplifier en considérant que lors de l'achat à un fournisseur, le magasin crée les articles qu'il achète, et qu'il détruit les articles qu'il a vendus.

Par ailleurs, il solde un article lorsqu'il est à trois jours de sa date d'expiration, (dans ce cas, il détruit un article non soldé, et crée un article soldé.) Un article qui expire le jour même est, lui, donné (il est donc aussi détruit).

Implémentez la classe `Magasin`. Il y aura dans cette classe, entre autres, un attribut date qui donnera la date du jour (avec la même convention que pour les articles). Le stock sera un tableau de pointeurs sur articles de taille fixée par le constructeur. Par convention, les cases où il n'y a pas d'article seront égales à `nullptr`.

On aura entre autres les méthodes suivantes

— la méthode `achat` créera un nouvel article non soldé en utilisant les arguments de la méthode, et le mettra dans le stock.

— la méthode `vente` détruira un article, par exemple, le premier article du tableau.

— la méthode `passerJour` qui va augmenter la date d'un jour et fera le nécessaire pour les articles près d'être périmés ou déjà périmés. (Faire des méthodes intermédiaires.)

Comme on ne sait pas encore trouver le type d'un objet, pour éviter de solder plusieurs fois le même objet, on soldera les articles qui sont exactement à 3 jours de la date de péremption. Et on pourra éventuellement modifier achat pour qu'il crée un article soldé au lieu d'un article si la date de péremption est trop proche.

S'il vous reste du temps, vous pouvez simuler les entrées et sorties d'argent du magasin. On suppose qu'il achète les articles la moitié du prix de vente normal. Et vous déduirez chaque jour des frais de fonctionnement fixe : salaires, loyer, etc.