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

TP n° 10 : Héritage – templates

Exercice 1 Si vous ne l'avez pas fait finissez le TP 9.

Héritage Multiple

Exercice 2 On veut représenter les articles d'un magasin et pour chacun d'eux savoir son prix et, si les données sont disponible et/ou ont un sens, le coût écologique (en grammes de CO2) et le nombre de calories qu'ils représentent.

On aura 4 classes :

- `Article` pour les articles dont on ne connaît que le prix ;
- `ArticleCaloriMesurable` pour ceux dont on connaît le prix et le nombre de calories ;
- `ArticleEcoMesurable` pour les articles dont on connaît le prix et le coût écologique ;
- `ArticleMesurable` pour ceux pour lesquels on a toutes les données.

Écrivez ces 4 classes. On y mettra une méthode `affiche()` que l'on redéfinira dans toutes les classes. Pensez à définir également les destructeurs et les constructeurs par copies. Et testez tout cela.

Que se passse-t'il si, dans `ArticleMesurable`, on ne redéfinit pas la fonction `affiche()`. ?

Définition de l'opérateur `()`

Exercice 3 On va définir l'"interface" (i.e. classe avec seulement des méthodes virtuelles pures) ci-dessous.

```
class ValeursAdmises {//"interface"
 public :
  virtual  bool operator()(char val) = 0;
};
```

Dans cette interface, on indique que l'opérateur "`()`" est défini comme prennant un argument `val` de type `char`. En gros, cette interface représente une fonction qui va vérifier que la valeur `val` répond à certains critères.

On va écrire deux sous-classes concrètes de `ValeursAdmises` :

- `ValeursAdmisesIntervalle` qui va contenir une valeur min et une valeur max et dont la "fonction" va vérifier que la valeur donnée en argument est dans cet intervalle.

— `ValeursAdmisesTableauValeurs` qui va contenirun tableau de valeurs et dont la "fonction" va vérifier que la valeur donnée en argument est dans ce tableau.

Exemple d'utilisation :

```
ValeursAdmisesIntervalle inter('a', 'd');
 if(inter('e'))
   cout<< " la valeur 'e' est ok"<< endl;
 else
    cout<< " la valeur 'e' n'est pas ok"<< endl;
 if(inter('c'))
   cout<< " la valeur 'c' est ok"<< endl;
 else
    cout<< " la valeur 'c' n'est pas ok"<< endl;

 char tab[] = {'b', 'o', 'n', 'j', 'u', 'r'};
 ValeursAdmisesTableau tableau(tab, 6);
  if(tableau('j'))
   cout<< " la valeur 'j' est ok"<< endl;
 else
    cout<< " la valeur 'j' n'est pas ok"<< endl;
 if(tableau('c'))
   cout<< " la valeur 'c' est ok"<< endl;
 else
    cout<< " la valeur 'c' n'est pas ok"<< endl;
```

Exercice 4

— Reprenez l'exercice précédent en remplaçant `char` par un type `T` générique.

— Écrivez une fonction générique `filtre` qui va prendre en argument une liste `l` d'éléments de type `T` et un objet de type `ValeursAdmises<T>` et retournera une liste ne contenant que les éléments `l` qui sont admis.

On pourra tester avec des types genre `int`, `char`, mais aussi les fractions en (re)définissant les opérateurs adéquats.

1 Si vous avez le temps...