CORRECTION

TD 2 Musee

C++ L3 FLIN 606, 2011

ObjMus.h

```
#ifndef ObjMus_h
#define ObjMus_h
using namespace std;
#include<iostream>
#include <string>
class ObjMus
{
protected:
  int ref; string descr;
public:
 ObjMus();
 ObjMus(int,string);
 virtual ~ObjMus();
 virtual int getRef()const;
 virtual void  setRef(int);
 virtual string getDescr()const;
 virtual void setDescr(string);
 virtual void saisie(istream&);
 virtual void affiche(ostream&)const;
 };

#endif
```

ObjetLeg.h

```
#ifndef ObjLeg_h
#define ObjLeg_h
#include "ObjMus.h"
class ObjLeg :virtual public ObjMus
{
private:
  string donateur;
  int annDon;
public:
 ObjLeg();
 ObjLeg(int,string,string,int);
 virtual ~ObjLeg();
 virtual string getDonateur()const;
 virtual void setDonateur(string);
 virtual int getAnnDon()const;
 virtual void  setAnnDon(int);
 virtual void saisie(istream&);
 virtual void affiche(ostream&)const;
 };


#endif
```

ObjetSign.h

```
#ifndef ObjSign_h
#define ObjSign_h
#include "ObjMus.h"
class ObjSign :virtual public ObjMus
{
private:
  string auteur;

public:
 ObjSign();
 ObjSign(int,string,string);
 virtual ~ObjSign();
 virtual string getAuteur()const;
 virtual void setAuteur(string);
 virtual void saisie(istream&);
 virtual void affiche(ostream&)const;
 };


#endif
```

SalleExpo.h

```
#ifndef SalleExpo_h
#define SalleExpo_h
#include "ObjMus.h"

/* Ne pas faire de methode saisie dans cette classe (pb du
type du new!!):
    on remplit la salle avec la methode ajoute.
  Ne pas faire non plus de constructeur par copie (même pb
que saisie + dynamic_cast)*/

class SalleExpo
{
 private:
  int capa;//capacite de la salle
  int rempl;//nb d'objets actuellement exposes
  ObjMus** t;// tableau dynamique de pointeurs sur les
objets exposés

 public:
  SalleExpo();//salle de capcite 10 par defaut
  SalleExpo(int c);//salle de capacite c
  virtual ~SalleExpo();
  virtual int getCapa() const;
  virtual int getRempl()const;
  virtual bool ajoute(int , ObjMus*);
  virtual ObjMus* enleve(int);
  virtual void affiche(ostream&) const;
};

#endif
```

ObjMus.cc

```
#include "ObjMus.h"

 ObjMus::ObjMus(){ref=0;descr="rien";}
 ObjMus::ObjMus(int r,string d){ref=r;descr=d;}
 ObjMus::~ObjMus(){}
 int ObjMus::getRef()const{ return ref;}
 void  ObjMus::setRef(int r){ref=r;}
 string ObjMus::getDescr()const{return descr;}
 void ObjMus::setDescr(string d){descr=d;}
 void ObjMus::saisie(istream& is)
  {is>>ref>>descr;}
 void ObjMus::affiche(ostream& os)const
  {os<<ref<<" "<<descr<<endl; }
```

ObjLeg.cc

```
#include "ObjLeg.h"

 ObjLeg::ObjLeg(){donateur="personne";annDon=0;}
 ObjLeg::ObjLeg(int r,string d, string don,int ad):ObjMus(r,d)
   {donateur=don;annDon=ad;}
 ObjLeg::~ObjLeg(){}
 string ObjLeg::getDonateur()const{return donateur;}
 void ObjLeg::setDonateur(string aut){donateur=aut;}
 int ObjLeg::getAnnDon()const{return annDon;}
 void  ObjLeg::setAnnDon(int ad){annDon=ad;}
 void  ObjLeg::saisie(istream& is)
   {ObjMus::saisie(is);is>>donateur>>annDon;}
 void ObjLeg::affiche(ostream& os)const
  {ObjMus::affiche(os);os<<" donateur: "<<donateur<<" date
du don: "<<annDon<<endl;}
```

ObjSign.cc

```
#include "ObjSign.h"

 ObjSign::ObjSign(){auteur="personne";}
 ObjSign::ObjSign(int r,string d, string
aut):ObjMus(r,d){auteur=aut;}
 ObjSign::~ObjSign(){}
 string ObjSign::getAuteur()const{return auteur;}
 void ObjSign::setAuteur(string aut){auteur=aut;}
 void ObjSign::saisie(istream& is)
  {ObjMus::saisie(is); is>>auteur;}
 void ObjSign::affiche(ostream& os)const
  {ObjMus::affiche(os);os<<" auteur: "<<auteur<<endl;}
```

SalleExpo.cc

```
#include "SalleExpo.h"

SalleExpo::SalleExpo()
  {capa=10; t=new ObjMus *[capa];
   for (int i=0;i<capa;i++)t[i]=NULL;
   rempl=0;}

SalleExpo::SalleExpo(int c)
 {capa=c; t=new ObjMus *[capa];
  for (int i=0;i<capa;i++)t[i]=NULL;
  rempl=0;}

SalleExpo::~SalleExpo()
  {for (int i=0;i<capa;i++)
     if (t[i]!=NULL) delete t[i];
   delete[] t;}

int SalleExpo::getCapa() const {return capa;}

int SalleExpo::getRempl() const {return rempl;}

bool SalleExpo::ajoute(int i, ObjMus* obj)
  {if (t[i]!=NULL) return false;
    else { t[i]=obj; rempl++;return true;}
  }

ObjMus*  SalleExpo::enleve(int i)
   {if (t[i]==NULL) return NULL;
    else { ObjMus* s=t[i]; t[i]=NULL;rempl--;return s;}
   }

void SalleExpo::affiche(ostream& os ) const
{  os<<"contenu de la salle:"<<endl;
   for (int i=0;i<capa;i++)
     {if (t[i]!=NULL)
        {os<<"place "<<i<<" : "<<endl;
          t[i]->affiche(os); os<<endl;}
       // else os<<"place "<<i<<"vide"<<endl;
     }
   os<<"- - - - - - - - - - - - - - - "<<endl;
   }
```

mainMus.cc

```
# include <iostream>
#include "ObjMus.h"//pas obligatoire (inclus dans les suivants)
#include "ObjSign.h"
#include "ObjLeg.h"
#include "SalleExpo.h"

int main()
{
 cout<<"########################################"<<endl;
 cout<<"Essai des classes objets du musee"<<endl;
ObjMus Vase(1,"vase a vin");
Vase.affiche(cout);

ObjMus* Alb=new ObjMus(2,"Le Grand Albert");
Alb->affiche(cout);
cout<<"--------------"<<endl;

ObjSign Joc(3, "La Joconde","Leo");
Joc.affiche(cout);

cout<<"---------------"<<endl;

ObjSign* Ven=new ObjSign(4,"Venus avec bras", "Mimile");
Ven->affiche(cout);

ObjMus* x; x=new ObjSign(10, "Tournesols","Auguste");
x->affiche(cout);

cout<<"----------------------"<<endl;

ObjLeg Ren(5,"renard empaille","M.Laglu",1925);
Ren.affiche(cout);

ObjLeg* Laj=new ObjLeg(6,"portrait du genereux donateur","M.Lajoie",1919);
Laj->affiche(cout);


cout<<"----------------------------"<<endl;
```

```
 cout<<"########################################"<<endl;
 cout<<" Essai de la classe SalleExpo "<<endl;
 cout<<" ------------------------- "<<endl;
 SalleExpo s;
 s.affiche(cout);
 s.ajoute(0,x);s.ajoute(1,Ven);s.ajoute(2,Laj);
 s.affiche(cout);
 ObjMus* r=s.enleve(3);cout<<(void*)r<<endl;
 r=s.enleve(1);cout <<"objet enleve :"<<endl;;r->affiche(cout);
 cout<<"____"<<endl;
 s.affiche(cout);
}

/*
########################################
Essai des classes objets du musee
1 vase a vin
2 Le Grand Albert
--------------
3 La Joconde
 auteur: Leo
---------------
4 Venus avec bras
 auteur: Mimile
10 Tournesols
 auteur: Auguste
----------------------
5 renard empaille
 donateur: M.Laglu date du don: 1925
6 portrait du genereux donateur
 donateur: M.Lajoie date du don: 1919
----------------------------
########################################
 Essai de la classe SalleExpo
 ---------------------------
contenu de la salle:
- - - - - - - - - - - - - - -
contenu de la salle:
place 0 :
10 Tournesols
```

```
 auteur: Auguste

place 1 :
4 Venus avec bras
 auteur: Mimile

place 2 :
6 portrait du genereux donateur
 donateur: M.Lajoie date du don: 1919

- - - - - - - - - - - - - - -
0
objet enleve :
4 Venus avec bras
 auteur: Mimile
____
contenu de la salle:
place 0 :
10 Tournesols
 auteur: Auguste

place 2 :
6 portrait du genereux donateur
 donateur: M.Lajoie date du don: 1919

- - - - - - - - - - - - - - -

 */
```

Quelques Explications

```
Objets de musee:
--------------
1)une classe ObjMus et deux sous classes ObjSign et ObjLeg

Remarque importante:
La classe ObjLeg a comme attributs supplementaires par rapport a ObjMus
une chaine de caracteres (le nom du donateur) et un entier (la date du
don).

Si on ne se réfère pas au sens des attributs, on peut aussi dire que ObjLeg
a les memes attributs que ObjSign, plus un entier (la date du don), et
faire
de ObjLeg une sous-classe de ObjSign.
Ca suppose qu'on confond l'auteur de l'objet signe et le donateur de
l'objet
legue. Expliquer qu'il vaut mieux eviter ca!!

2)
Ne pas oublier dans toutes les  classes:
 -  const pour  toute methode qui ne modifie pas l'objet;
 -  virtual systematique.

Pour les sous classes:
 - le constructeur avec parametres doit transmettre  au constructeur de
ObjMus les
   parametres qui le concernent, et non faire les inits lui meme;
 - ne redéfinir que les méthodes qui ont besoin de l'être;

Pour le td, inutile d'ecrire les .cc entiers: les constructeurs ( pour la
transmission de param) et les methodes affiche et saisie (pour
l'utilisation
de la méthode de la classe mere) suffisent.

3) les laisser écrire quelque chose du genre:

 ------------------------
 ObjMus Vase(1,"vase a vin"); Vase.affiche(cout);

 ObjMus* Alb=new ObjMus(2,"Le Grand Albert"); Alb->affiche(cout);

 ObjSign Joc(3, "La Joconde","Leo");Joc.affiche(cout);

 ObjSign* Ven=new ObjSign(4,"Venus avec bras", "Mimile");Ven-
>affiche(cout);

 ObjLeg Ren(5,"renard empaille","M.Laglu",1925); Ren.affiche(cout);

 ObjLeg* Laj=new ObjLeg(6,"portrait du genereux donateur","M.Lajoie",1919);
 Laj->affiche(cout);
------------------------


et leur proposer ensuite:

 ----------------------------
ObjMus* x; x=new ObjSign(10, "Tournesols","Auguste");
 x->affiche(cout);
```

```
------------------------

en leur demandant ce qui s'affiche et pourquoi.

Salles d'exposition:
--------------------
L'important est de les amener à découvrir la représentation des objets
dans la salle par un tableau de pointeurs sur la classe mere: un tel
pointeur
est le seul moyen de stocker indifféremment tous les types d'objets.
(le baratin à propos d'une seule methode pour ajouter essaie d'eviter
qu'ils proposent autant de methodes que de types d'objets, en changeant le
 paramètre).
Ensuite (pour l'affichage par exemple) on obtient un comportement correct
pour
les méthodes redéfinies, à condition de ne pas avoir oublié virtual pour
garantir la liaison dynamique.

Surtout NE PAS definir de methode de saisie: si on doit créer les objets,
il
faut connaître le type exact pour pouvoir faire le new... et on termine
avec
un switch où on passe en revue toutes les classes d'objets possibles.
Ce n'est pas faux, mais le but de cette question est de leur montrer le
fonctionnement de la liaison dynamique, et non les cas où on ne peut pas
s'en servir. Le cas de la saisie sera traité dans le td Entrepot en fin
de semestre.

Respecter l'absence demandée de constructeur par copie explicite: il est
encore
pire que la saisie (il faut en plus utiliser un dynamic-cast...ils verront
ça en iup2).

Remarque: ils ont vu au 1er semestre une classe "liste doublement chainee",
qu'ils peuvent vouloir utiliser à la place du tableau. Ne pas les laisser
faire: on veut que les places vides existent vraiment.
Et en plus, ça allonge le code, ça n'apporte rien pour la compréhension, et
il y a quelques complications sous-jacentes qu'ils n'ont pas besoin de
voir maintenant.


Annexe:
------
 a ne pas aborder, sauf si la question est posée par les étudiants:

 on peut appeler dans un constructeur une methode de la classe,
 mais...:

 si on appelle dans le constructeur de ObjMus une methode qui est redefinie
 dans ObjSign,que se passe-t-il lors de la creation d'une instance de
ObjSign?
 la reponse est que C++ n'est pas Java, et que la methode appelee est
 toujours celle de ObjMus (motif: un constructeur n'est jamais virtual,
pour
 la bonne(?) raison que le type d'un objet est connu quand il est fini de
 construire; en clair, quand le constructeur de ObjMus s'execute, il ne
sait
 pas dans quel type d'objet il est).
```