```
////////////////////////////////////////////////////////////////////////////
// fichier Element.h
////////////////////////////////////////////////////////////////////////////
#ifndef ELEMENT_H
#define ELEMENT_H

class Element {
  protected:
    int my_x, my_y, my_z;  // coordonnées
  public:
    Element( int x, int y, int z );
//   virtual
    ~Element();
//   virtual
    bool mobile() const;
//   virtual
    void deplacer( int x, int y, int z );
};

#endif // #ifndef ELEMENT_H

////////////////////////////////////////////////////////////////////////////
// fichier Element.cc
////////////////////////////////////////////////////////////////////////////
#include <iostream>
#include "Element.h"

using namespace std;

Element::Element( int x, int y, int z )
{
  cout << "appel constructeur Element..." << endl;
  my_x = x;
  my_y = y;
  my_z = z;
}

Element::~Element()
{
  cout << "appel destructeur Element..." << endl;
}

bool
Element::mobile() const
{
  return false;
}

void
Element::deplacer( int x, int y, int z )
{
}
```



```
////////////////////////////////////////////////////////////////////////////
// fichier Animal.h
////////////////////////////////////////////////////////////////////////////
#ifndef ANIMAL_H
#define ANIMAL_H

#include <string>
#include "Element.h" // declarons la classe de base

using namespace std;

class Animal : public Element {   // heritage ...
 private:
   string my_n;   // on peut rajouter le nom de l'animal
 public:
   Animal( string n, int x, int y, int z );
   ~Animal();
   bool mobile() const;
   void deplacer( int x, int y, int z );
};

#endif // #ifndef ANIMAL_H

////////////////////////////////////////////////////////////////////////////
// fichier Animal.cc
////////////////////////////////////////////////////////////////////////////
#include <iostream>
#include <string>
#include "Element.h"
#include "Animal.h"

using namespace std;

Animal::Animal( string n, int x, int y, int z )
: Element (x, y, z)
{
  cout << "appel constructeur Animal..." << endl;
  my_n = n;
}

Animal::~Animal()
{
  cout << "appel destructeur Animal..." << endl;
}

bool
Animal::mobile() const
{
  return true;
}

void
Animal::deplacer( int x, int y, int z )
{
  cout << "deplacement de ("<< my_x << "," << my_y << "," << my_z << ")";
  my_x += x;
  my_y += y;
  my_z += z;
  cout << " vers ("  << my_x << "," << my_y << "," << my_z << ")" <<endl;
}
```

```
////////////////////////////////////////////////////////////////////////////
// fichier "main.cc"
////////////////////////////////////////////////////////////////////////////
#include <iostream>
#include "Element.h"
#include "Animal.h"

using namespace std;

int
main()
{
  Animal poisson("poisson", 2, 5,1);
  // utilise la fonction membre de Animal
  if (poisson.mobile()) poisson.deplacer(1,0,0);

  Element epoisson(5, 9, 10);
  // utilise la fonction membre de Element
  if (epoisson.mobile()) epoisson.deplacer(1,0,0);


  Element *pelement;

  // utilise la fonction membre de Element
  pelement = &epoisson;
  if (pelement->mobile())
    pelement->deplacer(1,0,0);

  // si pas virtual dans Element.h,
  // utilise la fonction membre de Element et non de Animal
  pelement = &poisson;
  if (pelement->mobile()) pelement->deplacer(1,0,0);

  // parcours d'un tableau de pointeurs
  cout << endl << "Parcours d'un tableau de pointeurs" << endl;
  Element* tab[2];
  tab[0] = &epoisson;
  tab[1] = &poisson;
  for (int i=0 ; i<2 ; i++)
    if (tab[i]->mobile()) tab[i]->deplacer(1,0,0);

  return 0;
}
```



```
****************************************************************************
* TRACE: sans virtual
****************************************************************************
appel constructeur Element...
appel constructeur Animal...
deplacement de (2,5,1) vers (3,5,1)
appel constructeur Element...

Parcours d'un tableau de pointeurs
appel destructeur Element...
appel destructeur Animal...
appel destructeur Element...

****************************************************************************
* TRACE: avec virtual
****************************************************************************
appel constructeur Element...
appel constructeur Animal...
deplacement de (2,5,1) vers (3,5,1)
appel constructeur Element...
deplacement de (3,5,1) vers (4,5,1)

Parcours d'un tableau de pointeurs
deplacement de (4,5,1) vers (5,5,1)
appel destructeur Element...
appel destructeur Animal...
appel destructeur Element...
```