Université Paris Diderot **Langages à objets avancés**
M1 **Héritage Multiple**

Exercice 1 [Examen 2023-2024 (session 2) - 5 points]

1. Quel est l'affichage produit par le code suivant ?

```
class A {
 public:
  void show() { cout << "A"; }
};

class B : public virtual A {
 public:
  virtual void show()    {
    cout << "B";
  }
};

class C : public virtual A,
          public B {
 public:
  void show() { cout << "C"; }
};

class X {
 public:
  virtual void show()   {
    cout << "X";
  }
};
```

```
class Y : public X {
 public:
  void show() { cout << "Y";}
};

class AX :  public virtual A,
            public virtual X {
 public:
  void show() { cout << "AX"; }
};

class BY : public B, public Y {
 public:
  void show() { cout << "BY"; }
};
```

```
int main() {
  A* a    = new B();
  B* b    = new C();
  X* x_ax = new AX();
  BY* by  = new BY();
  A* a_by = by;
  B* b_by = by;
  Y* y_by = by;
  X* x_y  = new Y();
  // ... a suivre
```

```
// ... suite du main, et
    affichage
// utilisez le numero dans votre
    reponse
a     -> show();    // cas 1
b     -> show();    // cas 2
x_ax -> show();    // cas 3
by    -> show();    // cas 4
a_by -> show();    // cas 5
b_by -> show();    // cas 6
y_by -> show();    // cas 7
x_y   -> show();    // cas 8
}
```

2. Qu'est ce qui a été construit exactement avec `new C()` et `new BY()` ?

Exercice 2 On veut pouvoir représenter pour des aliments, si les données sont disponibles, leur coût écologique (en grammes de CO2) et le nombre de calories qu'ils représentent.

— Aliment ceux dont on ne connaît que le nom ;

— AlimentCaloriMesurable ceux dont on connaît le nom et le nombre de calories ;

— AlimentEcoMesurable ceux dont on connaît le nom et le coût écologique ;

— AlimentAllMesurable ceux pour lesquels on a toutes les données.

Écrivez ces 4 classes, en utilisant une hiérarchie naturelle. Munissez chaque classe d'une méthode virtual string getData() const et testez les.

Que se passe t'il si on ne redéfini pas getData dans AlimentAllMesurable ?

Exercice 3 Dans cet exercice on prévoit d'étendre une classe `Figure` avec par exemple `Triangle` ou `Carré` pour lesquelles on calculera la `surface`, et dont on voudra connaître la `nature` (qui sera respectivement la chaîne "triangle" et "carré"). Vous utiliserez la classe std::pair pour modéliser un point du plan par deux coordonnées entières :

```
#include <utility> // pour std::pair
#include <ostream> // pour ostream
using namespace std;
int main() {
  pair<int,int> a{10,20};
  cout << a.first << a.seconde;
  a.first=- a.first; // exemples d'opérations
  a.second= -a.second;
}
```

1. Déclarez une classe Figure qui possède un attribut de type string représentant le nom de l'objet. Ajoutez, en anticipation, deux déclarations de méthodes *virtuelles pures* : string nature() et double surface().

   On sait que le nombre de points des figures sera fixe selon leur type réel : les triangles auront toujours exactement trois points, les quadrilatères quatre etc ... Nous souhaitons stocker ces points (chacun représenté par `pair<int,int>`) dans une structure adaptée au niveau de la classe mère Figure dans une variable allPoints. Le constructeur Figure (string, int) prendra pour argument un string utilisé comme nom, et une valeur entière qui permettra de fixer définitivement le nombre de points.

   Une réflexion doit être faite sur le type le plus convenable pour `allPoints` :

   — A ce moment là nous ne connaissons pas la valeur exacte des points, elles devront donc être modifiables.

   — `vector<pair<int,int>>` présente l'inconvénient d'être un objet dont la longueur est potentiellement variable (car `remove` et `push_back` sont autorisés), il est donc à écarter.

— `array<pair<int,int>, N>` a bien une taille invariante N, mais a pour inconvénient que la valeur de N doit être une constante connue à la compilation, or on souhaite que ce paramètre puisse être librement choisit à l'appel du constructeur, cette valeur est donc indéterminée à priori.

— il nous reste la possibilité d'utiliser un pointeur vers un tableau de `pair<int,int>` construit dynamiquement à l'aide de `new pair<int,int>[n]`.

Terminez cette réflexion en fixant le type pour `allPoints`. N'oubliez pas qu'on ne pourra pas récupérer la taille du tableau simplement à partir de la variable `allPoints`, et donc qu'il vous faudra la stocker.

Ecrivez Figure (string, int).

2. Ecrivez aussi Figure(string nom, int nbPoints, int const *x_values, int const *y_values) qui construit les nbPoints dont les coordonnées sont décrites par une suite d'absisses x_values et une suite d'ordonnées y_values.
3. Le choix précédent à des conséquences sur la manipulation des figures : copie/affectation/destruction. Assumez les.
4. Définissez une classe Triangle qui hérite de Figure. On veux la munir d'un seul constructeur compatible avec les appels suivant :

```
int x_values[3] {0,1,2};
int y_values[3] {0,1,0};
Triangle t{"T1", x_values, y_values};
Triangle t{"T2", {0,1,2}, {0,1,0}}; // ici des données non nommées
```

mais incompatible avec :

```
int tab[] {0,1,2,3};
int * p= new int[3]{1,2,3};
// Triangle t3{"T3", tab, p}; // echoue à la compilation
// car ni tab ni p n'ont un type déclaré qui garantit
// que la taille des données est 3
```

Pour cela vous utiliserez le type const int (& values) [3] qui permet de typer une variable values comme référence vers un tableau de trois entiers constants.

Expliquez (testez) le rôle que joue const.

5. Redéfinissez l'opérateur d'affichage (au niveau des figures) pour qu'il produise un affichage du genre : "`triangle [son_nom]:` " suivi des coordonnées des trois points.
6. Définissez la méthode surface() de façon à calculer la surface du triangle.
7. Déclarez une classe Quadrilatere qui hérite de Figure (ils ont quatre points). Dans la littérature vous pouvez trouver comment calculer la surface d'un quadrilatère, cette classe n'est donc pas abstraite.
8. On pourrait aller plus loin, et intégrer des classes Rectangle Carre et Losange. Quelle serait la forme de la hiérarchie ? Dans ce cas quels seraient la nature des héritages ?

9. (Si vraiment vous avez du temps, et une bonne aide documentaire en géométrie).
   - Définissez la classe Rectangle dont le constructeur prendra la position du sommet inférieur gauche, la longueur de sa base, celle de sa hauteur, et l'angle que forme sa base avec l'axe des abscisses.
   - Définissez la classe Losange dont le constructeur prendra la position de son sommet inférieur, la longueur de ses cotés, et les 2 angles que forment avec l'axe des abscisses le bord gauche de ce sommet, et son bord droit.
   - Définissez la classe Carre dont le constructeur prendra le sommet inférieur gauche, la longueur des cotés, et l'angle que forme sa base avec l'axe des abscisses