Semaine 6
Héritage (simple), Polymorphisme - Suite + Exemple des Formes

1 Fin du TD précédent

1. Revenir sur la notion de polymorphisme :
   - reprendre et commenter AquaV1
   - rappeler les mécanismes C++ du polymorphisme à travers l'exemple
   - réinsister sur l'intérêt du polymorphisme (et aussi quand déclarer une fonction virtuelle)
2. Traiter section Transtypage (sans trop insister non plus... c'est une notion difficile pour eux)
3. Traiter section Classes Abstraites et AquaV2

2 Exemple plus complet : les Formes

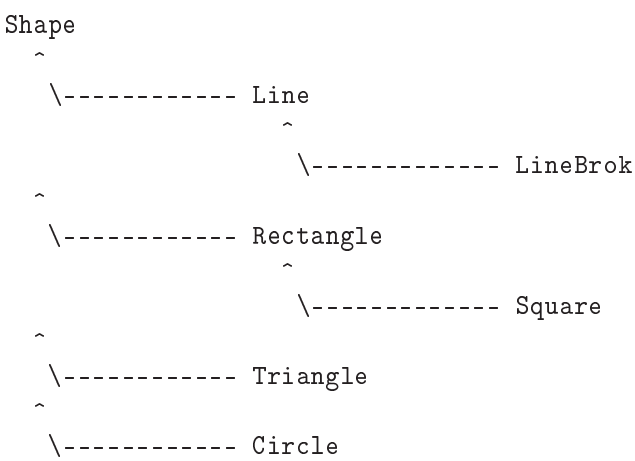
Voici un exemple assez "gros" permettant de bien illustrer la notion de classe abstraite et de polymorphisme. Il existe une multitude de façons d'implémenter cet exemple ! On en a choisi une qui servira aussi pour les 2 TPs qui viennent, pas trop difficile à mettre en oeuvre. C'est celle-ci qui va nous servir de fil conducteur ici.

Note aux enseignants : Se référer aux sources du TP.
En TD, on va étudier la hiérarchie des classes, leurs attributs, les méthodes à redéfinir ou non, virtuelles ou non, etc... Le contenu même des méthodes ne nous intéressera pas ici.
En particulier, on passera complètement sous silence la classe `Screen` qui va nous faciliter la tâche pour les affichages à l'écran (algorithmes de dessin de ligne ou de cercle fournis), on présentera les méthodes `draw` sans paramètres en TD par exemple.

Le but ici est d'amener les étudiants à réfléchir sur cette hiérarchie, les guider pour qu'ils devinent les principales caractéristiques. Ca peut être fait seul, par petits groupes, comme vous le sentez avec vos étudiants !

Le principal, c'est de les guider pour arriver vers une hiérarchie qui soit très proche de celle du TP (cf schéma ci-après). Il y a quelques trucs importants à leur faire remarquer :
- Classe Point : classe utile mais qui ne fait pas partie de la hiérarchie d'héritage !
- Classe Square : pas d'attribut en plus, juste un constructeur qui prend d'autres paramètres que Rectangle. Pas besoin non plus de redéfinir les méthodes draw et move, celles de Rectangle suffisent.
- Penser à tous les destructeurs virtuels !

On peut bien sûr "broder" autour de cette hiérarchie en TD (proposer d'autres méthodes comme perimetre ou aire etc), mais en TP, on se restreindra pour ne pas se perdre au milieu du code...




```
class Shape
{
  protected:
    char my_col;

  public:
    virtual ~Shape();
    virtual void draw() = 0;
    virtual void move( int dx, int dy ) = 0;

    void setColour( char col );
    char getColour( ) const;
};

class Line : public Shape
{
  protected:
    Point my_w, my_e;

  public:
    Line( char col, const Point & a,
                    const Point & b );
    Line( char col, const Point & a, int l );
    virtual ~Line();
    virtual void move( int dx, int dy );
    virtual void draw();
};

class Rectangle : public Shape
{
  protected:
    Point my_sw, my_ne;

  public:
    Rectangle( char col, const Point & a,
                         const Point & b );
    Rectangle( char col, const Point & a,
                         int width, int height );
    virtual ~Rectangle();
    virtual void move( int dx, int dy );
    virtual void draw( );
};
```

```
class Circle : public Shape
{
  protected:
    Point my_center;
    int my_radius;

  public:
    Circle( char col, const Point & a, int r );
    virtual ~Circle();
    virtual void move( int dx, int dy );
    virtual void draw( );
};

class Triangle : public Shape
{
  protected:
    Point my_p1, my_p2, my_p3;

  public:
    Triangle( char col, const Point & a,
                        const Point & b,
                        const Point & c );
    virtual ~Triangle();
    virtual void move( int dx, int dy );
    virtual void draw( );
};

class Square : public Rectangle
{
  public:
    Square( char col, const Point & a, int width );
    virtual ~Square();
};

class LineBrok : public Line
{
  protected:
    Point my_m;

  public:
    LineBrok( char col, const Point & a,
                        const Point & b,
                        const Point & m );
    virtual ~LineBrok();
    virtual void move( int dx, int dy );
    virtual void draw( Screen & s );
};
```