TD-TP : Le Design Pattern Structurel « Composite »

Il s'agit d'expérimenter le patron de conception Composite à un programme d'affichage de graphiques. Les objets élémentaires seront des instances des classes Point, Ligne ou Rectangle. Ils servent de base à la composition d'objets graphiques composés. Les opérations métier seront afficher, effacer, dePlacerDe. Dans un premier temps on se contentera de combiner des objets de la classe Point. Dans un deuxième temps, on pourra développer les caractéristiques et méthodes métier de Ligne et Rectangle.

Travail à faire représentation UML

1. Etant donnés les fichiers d'entête .h ci-dessous, (r)établir le schéma de classe UML correspondant au Design Pattern Composite sous-jacent à cette application.

```
/**
 * @file Graphique.h
 * @brief spécifie Graphique
Méthodes : ajouter, supprimer, jeSuisCompos
           afficher, effacer, deplacerDe
 * @author Lopistéguy
 * @version 0.1
 * @date jj/mm/aaa
 */

#ifndef GRAPHIQUE_H
#define GRAPHIQUE_H
#include <iostream>
#include "Terminal.h"

using namespace std;

class Graphique {
/** Membres propres au patron Composite **/
public :// Méthodes : ajouter, supprimer, jeSuisComposé
    virtual bool ajouter  (Graphique*);
    virtual bool supprimer(Graphique*);
    virtual bool jeSuisCompose ()  { return false;  }; // a priori NON composé

/** Membres métier à spécialiser **/
public: // Méthodes : afficher, effacer, déplacerDe
    virtual void afficher () = 0;           /** A définir dans les sous-classes **/
    virtual void effacer  () = 0;           /** A définir dans les sous-classes **/
    virtual void deplacerDe (int, int) = 0; /** A définir dans les sous-classes **/
};


#endif // GRAPHIQUE_H
```

```
/**
 * @file GraphiqueSimple.h
 * @brief spécifie GraphiqueSimple - hérite de Graphique
Méthodes : afficher, effacer, deplacerDe, dessiner
 * @author Lopistéguy
 * @version 0.1
 * @date jj/mm/aaa
 */

#ifndef GRAPHIQUESIMPLE_H
#define GRAPHIQUESIMPLE_H

#include "Graphique.h"

class GraphiqueSimple : public Graphique {
/** Membres propres au patron Composite - NON - **/

/** Membres métier **/
public :// Méthodes : afficher, effacer, deplacerDe, dessiner
    void afficher ();
    void effacer ();
    virtual void deplacerDe (int, int) = 0; /** A définir dans les sous-classes **/
    virtual void dessiner (bool) = 0;       /** A définir dans les sous-classes **/
};

#endif // GRAPHIQUESIMPLE_H
```

```
/**
 * @file Point.h
 * @brief spécifie Point - hérite de GraphiqueSimple
Attributs : ligne, colonne, motif
Méthodes  : Point, definir, deplacerDe, dessiner
 * @author Lopistéguy
 * @version 0.1
 * @date jj/mm/aaa
 */

#ifndef POINT_H
#define POINT_H

#include "GraphiqueSimple.h"

class Point : public GraphiqueSimple {
/** Membres propres au patron Composite - NON - **/

/** Membres métier  **/
private :
    int ligne, colonne; // Ou sera affiché le motif
    char motif;         // Représente le point
public :
    Point (int, int, char='o'); // Paramètres : ligne, colonne, motif
    void definir (int, int);    // Nouvelle emplacement : ligne, colonne
    virtual void deplacerDe (int, int); // ... de deltaLigne et deltaColonne
protected :
    void dessiner (bool);// si bool = vrai, écrit le motif
};                       // si bool = faux, écrit des espaces (cf. efface)

#endif // POINT_H
```

```
/**
 * @file GraphiqueCompose.h
 * @brief spécifie GraphiqueCompose - hérite de Graphique
Attributs : enfants
Méthodes  : Point, definir, deplacerDe, dessiner
 * @author Lopistéguy
 * @version 0.1
 * @date jj/mm/aaa
 */

#ifndef GRAPHIQUECOMPOSE_H
#define GRAPHIQUECOMPOSE_H

#include "Graphique.h"
#include <set>

class GraphiqueCompose : public Graphique {

/** Membres propres au patron Composite **/
private: // Attributs : enfants
    set <Graphique*> enfants;
public:// Méthodes : GtraphiqueComposé, ajouter, supprimer, jeSuisComposé
    GraphiqueCompose();
    virtual bool ajouter        (Graphique*);
    virtual bool supprimer      (Graphique*);
    virtual bool jeSuisCompose  () {  return true;  }; // retourne VRAI

/** Membres métiers **/
public : // Méthodes : afficher, effacer, deplacerDe
    virtual void afficher ();
    virtual void effacer ();
    virtual void deplacerDe (int, int);
};

#endif // GRAPHIQUECOMPOSE_H
```

Travail à faire : opérations métier d'un GraphiqueSimple (cf. afficher, effacer et deplacerDe)

2. Donner le code du constructeur de Point

Etant donné le code des méthodes ci-dessous

```
void GraphiqueSimple::afficher () {
   dessiner (true);
}

void GraphiqueSimple::effacer () {
   dessiner (false);
}

void Point::dessiner (bool marquer) {
   Terminal::charLigneColonne (ligne, colonne, (marquer ? motif : ' '));
}

// @fct charLigneColonne : écrit motif en ligne l et colonne c
//                         en partant du point haut gauche
static void Terminal::charLigneColonne (int l, int c, char motif);
```

3. Quelles sont les conséquences des instructions ci-dessous ?

```
Point point1 (6, 10, '*');
point1.afficher();
point1.effacer();
```

Recommandation : suivre le diagramme UML

4. Donner le code de la méthode Point::deplacerDe (int int)

Travail à faire : composer un GraphiqueComposé

5. Sachant que pour un set la méthode insert retourne une pair d'informations

> *pair< iterator,bool>* ***set::insert*** *(const value_type& val);*
>
> *pair::first set to an iterator pointing to either the newly inserted element or to the equivalent element already in the set. The pair::second element in the pair is set to true if a new element was inserted or false if an equivalent element already existed*

donner le code de la méthode ajouter, de la classe GraphiqueComposé

6. Donner le code du constructeur de GraphiqueComposé

7. Sachant que pour un set la méthode erase retourne le nombre d'éléments enlevés

> *int* ***set::erase*** *(const value_type& val);*

donner le code de la méthode supprimer, de la classe GraphiqueComposé

Travail à faire créer un GraphiqueComposé

8. Créer 3 Point, et les regrouper dans un objet GraphiqueComposé

Travail à faire opérations métier d'un GraphiqueComposé (cf. afficher, effacer et deplacerDe)

9. Donner le code de la méthode afficher de la classe GraphiqueComposé

10. Faire de même pour les autres méthodes métier effacer et deplacerDe

Note : une fois apportées toutes ces modifications sur le projet incomplet présent sur eLearn, votre programme fonctionnera.

Vous observerez sur le net que certaines mises en œuvre du patron Composite placent les méthodes : ajouterComposant, supprimerComposant dans la sous-classe Composé et non dans la super-classe Composant. De plus la méthode jeSuisComposé n'apparaît plus.

Travail à faire

11. Pour cela dupliquez le répertoire Code::Blocks qui contient l'ensemble des fichiers de votre projet actuel, puis modifiez le code de ce nouveau projet pour implémenter la solution alternative décrite ci-dessus.