LO43 Médian du 25 Novembre 2005

Durée 2 heures – aucun document autorisé – rendre une copie différente par partie

Partie A

On s'intéresse à une application d'édition d'objets graphiques qui propose les fonctionnalités suivantes:

- regroupement de plusieurs objets graphiques en un « groupe d'objets graphiques » pouvant être manipulé comme un objet graphique,
- sélection multiples d'objets graphiques pour leur appliquer un traitement particulier.

On dispose dans cette application des classes « ObjetGraphique », « Groupe » et « Selection » qui ont la forme suivante :

```
/**
* classe ObjetGraphique
*/
class ObjetGraphique {
protected:
float x, y ;
...
};
```

```
/**
* classe Groupe
*/
class Groupe : public ObjetGraphique {
protected:
int nbrObjetGraphique ;
ObjetGraphique** listeObjetGraphique ;
...
};
```

```
/**
* classe Selection
*/
class Selection {
protected:
int nbrObjetGraphique ;
ObjetGraphique** listeObjetGraphique ;
...
};
```

1. Est-il nécessaire de définir la forme canonique de Coplien pour ces classes ? Détailler.
2. Donner la forme canonique de Coplien de la classe Selection.
3. Préciser le destructeur des deux autres classes (déclaration et implémentation).

On désire maintenant mettre en oeuvre la fonctionnalité qui permet d'appliquer un traitement spécifique à tous les objets d'une sélection. On considère pour cette fonctionnalité qu'un groupe d'objets est traité comme un seul objet. Pour cela, on prévoit d'utiliser une classe « Iterateur » (externe aux autres classes existantes) qui permettra d'examiner tous les objets d'une sélection donnée, de la façon suivante :

```
void traitement(ObjetGraphique* obj) ;
...


int main ()
{
Selection selection ;
...
Iterateur iterateur(selection) ;
while (iterateur.hasNext()) {
traitement ( it.getNext() ) ;
}
};
```

4. Donner la déclaration de la classe Iterateur.
5. Ecrire l'implémentation des différentes méthodes de la classe Iterateur.

Après réflexion, on désire mettre en oeuvre l'itérateur en tant que classe imbriquée de la classe Selection. Une classe imbriquée est définie à l'intérieur du contexte d'une autre classe et peut être instanciée comme toute autre classe. Dans cette configuration, l'utilisation de l'itérateur sur une sélection se ferait de la façon suivante :

```
class Selection {
...
class Iterateur {
...
};
};


int main() {
Selection selection ;
...
for (Selection::Iterateur it=selection.begin();it!=selection.end();it++) {
traitement (*it) ;
}
}
```

6. Donner la déclaration de la classe Iterateur et de la classe Selection dans la configuration classe imbriquée.
7. Ecrire l'implémentation de toutes les méthodes des classes Itérateur et Selection dans ce cas là.
8. Que pensez-vous de la conception de cette application, est-il possible de l'améliorer ?

NOM : _____

Prénom :_____

Partie B

Exercice 1 - Indiquer quelles sont les erreurs de syntaxe C++ lorsqu'il en existe (souligner les erreurs et remplir dans tous les cas la case commentaire), liées aux droits d'accès sur les objets, dans les exemples suivants :

1)
```
class B {protected: int i; friend void f();};
class D : public B {};
void f( ) {B* p = new B; D* q = new D; int fi1 = p->i; int fi2 = q->i;}
```

Commentaire

2)
```
class B {protected: int i;}; class D : public B {};
void f( ) {B* p = new B; D* q = new D; int fi1 = p->i; int fi2 = q->i;}
```

Commentaire

3)
```
class A {protected: int a;}; class B : public A {};
class C : public B {void f(B* p);};
void C::f(B* p) {a = 1; p->a = 2;}
```

Commentaire

4)
```
class A {protected: int a;}; class B : public A {void f( );};
class C : private B {void f();};
void B::f( ) {a = 1;}
void C::f( ) {a = 1;}
```

Commentaire

5)
```
class A {protected: int a;}; class B : private A {void f( );};
class C : public B {void f( );};
void B::f( ) {a = 1;}
void C::f( ) {a = 1;}
```

Commentaire

6)
```
class B {protected: int a;};
class D : public B {friend void f( );public: int b;};
void f( ) {D* p; p->a = 1; p->b = 2; B* pp; pp->a = 1; pp->b = 1; }
```

Commentaire

7)
```
class A {protected: int a;}; class B : protected A {void f( );};
class C : private B {void f( );};
void B::f( ) {a = 1;}
void C::f( ) {a = 1;}
```

Commentaire

Exercice 2 - Considérons le type abstrait vecteur en C++. On s'intéresse à l'opérateur d'incrémentation postfixe v++ en comparaison avec celui préfixe ++v qui tous les deux incrémentent les composantes du vecteur appelant d'1 unité avec la différence que l'opérateur postfixe ne renvoie pas le vecteur incrémenté mais la valeur du vecteur avant l'incrémentation.

1) Donner les valeurs des résultats r1 et r2 et des vecteurs u et v après la séquence d'instruction suivante :

```
vecteur u("2 2 2"), v("2 2 2"); vecteur r1, r2;
r1 = u++; cout << r1 ; cout << u ;
r2 = ++v; cout << r2 ; cout << v ;
```

Donner les instructions du programme ci-dessus en notation fonctionnelle équivalente.

2) L'opérateur préfixe ++v s'écrit simplement en incrémentant les composantes et en renvoyant une référence sur l'appelant tandis qu'il n'en va pas de même pour l'opérateur postfixe qui renvoie un nouvel objet différent de l'appelant. Ecrivez les deux opérateurs sachant que leurs profiles sont donnés.

```
vecteur& operator++(void) { // préfixe ...}
vecteur operator++(int) { // postfixe ...}
```

Exercice 3 - Considérons le programme suivant :

```
class String {private : char* data ;
public :    String(const char* value = 0) ;
operator char*() const ; // conversion String en char*} ;

String::operator char*() const { return data ; }
```

Ainsi on peut écrire :

```
const String B("Hello World") ;
char* str = B ; // appelle B.operator char*
strcpy(str, "What I want !") ;
```

1) Qu'est-ce qui n'est pas correcte dans l'implémentation de l'opérateur de conversion, expliquer

Donner une version correcte de l'opérateur même si elle est plus lente. Quelle contrainte supplémentaire entraîne-t-elle ?

Exercice 4 - Considérons le type Vecteur en C++ tel que réalisé en TP.

a) Donner le profile de l'adition de deux vecteurs.

b) Donner la notation fonctionnelle équivalente des 5 instructions suivantes :

c = a + b; v[i] = v[j] = 2; a = (b = 2); (a = b) = 2; ((a = b) = c) = 2;

SIGNATURE :