TD/TP 3 C++

Polytech' Paris-Sud
Et4 info
Aleksandr Setkov et Marc Evrard

13 janvier 2015

1 Partie 1 : Abstraction

Il faut reprendre les classes **Personne, Employe, Etudiant et Manager** que vous aviez implémentées au dernier cours.

1. Faire le constructeur de la classe **Manager** public
2. Dans le main, créer les objets des classes **Employe, Etudiant, Manager**. Ensuite, il faut créer des pointeurs de type **Personne*** et leur assigner les références des objets créés. Par exemple :

```
        Employe e1(...);
        Personne *p1 = &e1
ou
        Personne *p1 = new Manager(...)
```

3. Que va afficher **p1→print()** dans chacun des cas ?
4. Faire la méthode **print()** de la classe **Personne** abstract
5. Que va afficher **p1→print()** maintenant dans chacun des cas ?
6. Faire le destructeur de la classe **Personne** virtual

2 Partie 2 : Les classes abstraites

1. Créer la classe abstraite **Shape**

Nom	Type
width	int
height	int

2. Créer les méthodes abstraites **area(), perimeter()** pour calculer la surface et le périmètre

3. Créer la classe **Triangle** qui hérite de la classe **Shape** et implémenter les méthodes **area() et perimeter()**

4. Créer la classe **Rectangle** qui hérite de la classe **Shape** et implémenter les méthodes **area() et perimeter()**

5. Implémenter la fonction **print(Shape* p)** qui affiche la surface et le périmètre.

3 Partie 3 : Héritage multiple

1. Créer la classe **CarteMere**

Nom	Type
Id	int
typeProcesseur	string
RAM	string
typeVideo	string

2. Implémenter un constructeur, et la méthode **getId()**

3. Créer la classe **Moniteur**

Nom	Type
Id	int
resolutionX	int
resolutionY	int
typeCouleur	string

4. Implémenter un constructeur, et la méthode **getId()**

5. Créer la classe **Ordinateur** qui hérite des classes **CarteMere** et **Moniteur**

Nom	Type
disqueDure	int
LecteurDeCartes	bool

6. Dans le main créer un objet de la classe **Ordinateur** et appeler la méthode **getId()**. Qu'est-ce qui est affiché ?

   Pour explicitement appeler la method **getId()** de chaque classe, on peut utiliser le format suivant :

```
Ordinateur ord1(...);
int id1 = ord1.CarteMere::getId();
int id2 = ord1.Moniteur::getId();
```

7. Créer la classe **Materiel**

Nom	Type
id	int

8. Implémenter la méthode **getId** pour lire **id**.

9. Modifier les classes **CarteMere** et **Moniteur** pour les faire hériter de la classe **Materiel** et utiliser l'**id** de cette classe

10. Dans le main, appeler la méthode **getId()** à partir d'un objet de la classe **Ordinateur**. Qu'est-ce qui est affiché ?

4 Partie 4 : Diamond problem

Pour résoudre le problème d'accès aux champs de la classe de base (Diamond problem), on peut utiliser **virtual inheritance**

1. Modifier le type d'héritage à **public virtual** pour les classes **CarteMere** et **Moniteur**

2. Modifier le constructeur de la classe **Ordinateur** pour appeler directement le constructeur de la classe de base

3. Dans le main, appeler la méthode **getId()** à partir d'un objet de la classe **Ordinateur**. Qu'est-ce qui est affiché ?

4. Quel est l'ordre d'appel des constructeurs (destructeurs) ?