

TP 4 — classes, première partie

Master première année, option EEA

<http://magphy.ujf-grenoble.fr/ratchov/eea/>

14 septembre 2004

1 Les classes en tant que agrégat de données

On se propose de réaliser une classe `point` permettant de manipuler des points de coordonnées entières sur un plan. Traditionnellement, par soucis de clarté, lorsqu'on construit une classe on crée deux fichiers portant le même nom de base que la classe : l'un contenant les déclarations (d'extension `.h`) et l'autre les définitions (d'extension `.cxx` ou `.cc` ou autre). Dans la suite, dans tous les TPs, on utilisera cette convention.

Exercice 1 *Construire une classe `point` ayant comme champs deux entiers `x` et `y`. Munissez-la d'un constructeur qui prend en arguments les coordonnées du point à créer et qui initialise les champs correspondants. Faites de sorte que le constructeur signale sur la console qu'il a été appelé. Faites de même pour le destructeur. Ajoutez aussi une méthode `print` qui affiche les coordonnées du point.*

Exercice 2 *Est-ce que les champs sont publics ou privés ? Sont-ils accessibles depuis `main()` ? Faites de sorte que les champs soient privés et munissez la classe d'un jeu de getters et setters, méthodes publiques qui permettent aux non-membres de la classe d'accéder aux champs, par exemple :*

```
- int  getx();  
- void setx(int);
```

Dans quels cas est-il plus avantageux d'utiliser les getters et les setters plutôt que d'accéder directement aux champs de l'objet.

Exercice 3 *Ajoutez un constructeur par copie, qui, lui aussi, signale qu'il est appelé*

Exercice 4 *Surchargez l'opérateur "=". Essayez de le faire de deux façons différentes : en passant son argument par valeur puis par référence.*

```
- operator =(const point  s);  
- operator =(const point &s);
```

Essayez cet opérateur dans un programme. Quelle est la différence entre ces deux façons de faire, expliquer exactement ce qui s'est passé dans les deux cas.

Exercice 5 *Ajoutez à la classe une méthode `translate` qui prend en argument deux entiers `dx` et `dy` et qui les ajoute à `x` et `y`. Essayez la. Ensuite ajoutez l'opérateur "+" à la classe ; vous pouvez par exemple utiliser la méthode `translate`. Comme pour l'opérateur "=", essayez le passage de l'argument par valeur et puis par référence. Dans un programme, essayez un appel tel que "`a = b + c`". Combien il y a-t-il de copies de points dans chaque cas ? Combien de copies sont réellement nécessaires ?*

Exercice 6 *Est-ce qu'il y a un avantage à utiliser l'opérateur `+` ou `=` plutôt que de définir des méthodes `somme` et `copie` qui feraient la même chose ?*

Exercice 7 *(À faire si le temps le permet) Refaites tous les exercices précédents en C, donc sans utiliser de classes. Comment :*

- définir les méthodes
- adresser les champs dans les méthodes
- avoir des champs et des méthodes privés
- éviter la recopie

2 Les classes en tant que composant logiciel

Dans cette section, le but sera de poser les structures de base permettant de manipuler des figures géométriques. Pour cela, on va définir une classe générique `figure` dont vont dériver les classes `segment`, `cercle`, `rectangle`, *etc...*

2.1 Héritage

Définissez une classe `figure` ne contenant qu'un constructeur, un destructeur et une méthode `print` affichant "`figure`". C'est la classe de base qui va servir de point de départ pour les autres classes ; elle va contenir toutes les caractéristiques communes aux classes qui en dérivent.

Exercice 8 *Définissez la classe `segment` qui dérive de `figure` et contenant deux instances de la classe `point`. Les deux points correspondent aux extrémités du segment. Cette classe aura besoin d'un constructeur qui initialise les deux points, définissez-le comme bon vous semble. Surchargez la méthode `print` pour qu'elle affiche "`segment`" et les coordonnées des extrémités.*

Exercice 9 *De même, définissez la classe `cercle` qui dérive de `figure` et contenant deux instances de la classe `point` (le centre et un point sur le cercle). Comme précédemment, surchargez la méthode `print`.*

Considérez l'extrait de programme suivant :

```
int main() {
    figure *a, *b;
    a = new figure();
    b = new cercle();
    a->print();
    b->print();
    return 0;
}
```

Pourquoi le compilateur nous laisse-t-il référencer un objet de type `cercle` par un pointeur de type `figure` ? Est-ce que c'est sans risque ?

Exercice 10 *Exécutez le programme ci-dessus et notez ce qu'il affiche. Ensuite, déclarez les méthodes `print` en tant que méthodes virtuelles. Exécutez le programme et comparez avec le résultat précédent, expliquer ce qui c'est passé.*