MOOC Intro POO C++
Tutoriels semaine 1 : classes et objets

Les tutoriels sont des exercices qui reprennent des exemples similaires à ceux du cours et dont le corrigé est donné progressivement au fur et à mesure de la donnée de l'exercice lui-même.

Ils sont conseillés comme un premier exercice sur un sujet que l'étudiant ne pense pas encore assez maîtriser pour aborder par lui-même un exercice «classique».

Les solutions sont fournies au fur et à mesure sur les pages paires.

> Cet exercice correspond à l'exercice «*pas à pas*» page 106 de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Le but de cet exercice est de reprendre l'exemple du cours illustrant les notions d'objet et de classe.

Le but est de définir une classe `Rectangle` représentant une abstraction (informatique) de ce qu'est un rectangle : une largeur, une longueur et sa surface.

Commencez par créer un fichier `Rectangle.cc` et définissez y la classe.

Essayez de le faire par vous même sans regarder la solution (page suivante).

Solution :

```
class Rectangle {
};
```

Bien ! Prévoyons tout de suite d'utiliser notre toute première classe en créant une instance dans le `main()`.

(solution page suivante)

Solution :

Cela se fait comme n'importe quelle autre variable utilisée jusque maintenant, la classe `Rectangle` étant simplement un nouveau type :

```
class Rectangle {
};
int main() {
  Rectangle rect;
  return 0;
}
```

Bon, jusque là pas grand chose de bien intéressant...
Implémentons maintenant l'abstraction « rectangle » en commençant par ajouter deux attributs : largeur et longueur.

(solution page suivante)

Solution :

Cela se fait aussi simplement que pour des champs d'une structure.

```
class Rectangle {
  double largeur;
  double longueur;
};
...
```

Passons maintenant à la surface, que nous implémentons sous la forme d'une méthode, c'est-à-dire une fonction propre à la classe.

Essayez de le faire par vous même sans regarder la solution (page suivante).

Solution :

Les méthodes se déclarent comme les fonctions usuelles, sauf :

- qu'on les met à l'intérieur de la classe (et non pas à l'extérieur) ;
- que l'on a pas besoin de passer comme argument les attributs de la classe. Ces attributs ont en effet toute la classe comme portée (i.e. ce sont des variables globales à la classe, donc tout élément de la classe les connaît).

Dans notre cas, on a donc pas besoin de passer `largeur` et `longueur` à la méthode `surface`.

```
class Rectangle {
  double largeur;
  double longueur;
  double surface () {
    return largeur * longueur;
  }
};
```

On peut maintenant essayer de tester notre classe : dans le `main()`, ajouter une affectation à chacun des champs et un appel à la méthode `surface()`.

(solution page suivante).

Solution :

```
#include <iostream>
using namespace std;

class Rectangle {
  double largeur;
  double longueur;
  double surface () {
    return largeur * longueur;
  }
};

int main() {
  Rectangle rect;
  rect.largeur = 1.5;
  rect.longueur = 12.8;
  cout << "Surface : " << rect.surface() << endl;
  return 0;
}
```

Essayez de compiler votre programme. Que se passe-t-il ?

Le message est assez clair : tous les attributs sont privés. En effet, lorsqu'aucun droit (`private`/`public`) n'est précisé, les membres (attributs/méthodes) sont considérés commes privés par défaut. Cela veut dire qu'on ne peut pas les utiliser hors de l'objet (ils ne font pas partie de l'interface de l'objet).

En particulier, on n'a pas le droit d'utiliser `rect.largeur` ! Il n'est pas défini à l'extérieur de la classe.

Pour rendre un attribut accessible à l'extérieur de la classe, il faut le déclarer comme `public:` :

(solution page suivante)

Solution :

```
#include <iostream>
using namespace std;

class Rectangle {
public:
  double largeur;
  double longueur;
  double surface () {
    return largeur * longueur;
  }
};

int main() {
  Rectangle rect;
  rect.largeur = 1.5;
  rect.longueur = 12.8;
  cout << "Surface : " << rect.surface() << endl;
  return 0;
}
```

Recompilez. Cela devrait fonctionner cette fois-ci.

Essayons maintenant d'affiner notre « style objet » en rendant privés les attributs et ne laissant publiques que des méthodes.

Essayez de le faire par vous même sans regarder la solution. (solution page suivante)

Solution :

```
#include <iostream>
using namespace std;

class Rectangle {
private:
  double largeur;
  double longueur;
public:
  double surface () {
    return largeur * longueur;
  }
};

int main() {
  Rectangle rect;
  rect.largeur = 1.5;
  rect.longueur = 12.8;
  cout << "Surface : " << rect.surface() << endl;
  return 0;
}
```

Il est clair qu'en l'état le programme ne compile plus, puisque `largeur` et `longueur` sont redevenus privés (donc `rect.largeur` est de nouveau interdit).

Il faut pour accéder aux attributs de la classe mettre en place les méthodes « get » et « set » correspondantes (elles sont évidement publiques !).

Cela se fait simplement, de façon similaire à ce que nous avons fait jusqu'ici :

```
#include <iostream>
using namespace std;

class Rectangle {
private:
  double largeur;
  double longueur;
public:
  double surface () {
    return largeur * longueur;
  }
  double getLongueur() { return longueur; }
  double getLargeur()  { return largeur;  }
  void setLargeur(double x)  { largeur  = x; }
  void setLongueur(double x) { longueur = x; }
};

int main() {
```

```
  Rectangle rect;
  rect.setLargeur(1.5);
  rect.setLongueur(12.8);
  cout << "Surface : " << rect.surface() << endl;
  return 0;
}
```

Pour finir et être parfaitement « propre », on peut marquer explicitement les méthodes qui ne modifient pas les attributs de l'objet (on parle de méthodes « prédicat ») par l'ajout du mot `const` derrière la déclaration des arguments de ces méthodes :

```
#include <iostream>
using namespace std;

class Rectangle {
private:
  double largeur;
  double longueur;
public:
  double surface () const {
    return largeur * longueur;
  }
  double getLongueur() const { return longueur; }
  double getLargeur()  const { return largeur;  }
  void setLargeur(double x)  { largeur  = x; }
  void setLongueur(double x) { longueur = x; }
};

int main() {
  Rectangle rect;
  rect.setLargeur(1.5);
  rect.setLongueur(12.8);
  cout << "Surface : " << rect.surface() << endl;
  return 0;
}
```
