



Systemes Embarqués Communicants
Spécification et modélisation de programme

TP7 - Classes

Auteur :
M. Jean-Marc KERVIL M.
Aymeric ZIMINSKI

Version du
29 janvier 2026

Table des matières

1	Introduction	2
2	Création d'une classe point	3
2.1	Constructeurs	3
2.2	Getter et Setter	3
2.3	Méthode translater	3
2.4	Surcharge des opérateurs	3
2.5	Tests	4
3	Classe Forme	4
4	Les formes géométriques	5
5	Tests des classes Forme, Cercle, Rectangle et Carré	5
6	Liste de forme	5

1 Introduction

Nous souhaitons dans ce TP expérimenter l'implémentations des classes en C++. Pour cela, nous créons une classe `Forme` qui est parente de la classe `Cercle` et `Rectangle`. Nous implémenterons des fonctions de calcul d'aire et de périmètre, ainsi que des fonctions d'affichage.

2 Création d'une classe point

Nous choisissons de créer une classe `Point` qui contient deux attributs `x` et `y` de type `float`.

Nous souhaitons implémenter un constructeur, une méthode `translater`,...

2.1 Constructeurs

Pour les constructeurs, nous souhaitons avoir différents moyens de créer un point :

- Sans paramètres : le point est à l'origine (0,0)
- Avec deux paramètres : le point est créé avec les coordonnées données
- A partir d'un autre point : recopie des coordonnées

Pour l'implémentation, nous récupérons les paramètres et nous les assignons aux attributs `x` et `y`.

2.2 Getter et Setter

Nous implémentons des getters et setters pour les attributs `x` et `y` : `getX`, `getY`, `setX`, `setY`.

Ces méthodes permettent de récupérer et modifier un attribut privé de la classe.

2.3 Méthode `translater`

La méthode `translater` permet de déplacer un point en ajoutant des valeurs aux coordonnées `x` et `y`.

Nous définissons deux méthodes `translater` : une qui prend deux paramètres (`x`, `y`) et une qui prend un autre point.

2.4 Surchage des opérateurs

Pour la surcharge des opérateurs, nous implémentons le « pour afficher un point dans le format (`x`,`y`). Nous le définissons comme `friend` de la classe pour pouvoir accéder aux attributs privés. Nous surchargeons aussi l'opérateur `+=` pour additionner deux points (addition des coordonnées).

2.5 Tests

Je demande à Copilote de m'aider à écrire des tests pour la classe Point. Je lui donne les instructions précises et il me génère le code.

Le résultat des tests est le suivant :

```
=====
TEST DE LA CLASSE POINT
=====

--- TEST POINT - CONSTRUCTEURS ---
Point par défaut: (0; 0)
Point(3.5, 7.2): (3.5; 7.2)
Point copié à partir de p2: (3.5; 7.2)

--- TEST POINT - GETTERS/SETTERS ---
p2.getX() = 3.5
p2.getY() = 7.2
Après setX(10.0) et setY(20.0), p1 = (10; 20)

--- TEST POINT - TRANSLATER ---
p4 avant translation: (5; 5)
Après tradlater(2.0, 3.0): (7; 8)
p5 avant translation: (1; 1)
Après tradlater par Point(4.0, 2.0): (5; 3)

--- TEST POINT - OPÉRATEUR += ---
p6 = (1; 1), p7 = (2; 3)
Après p6 += p7: p6 = (3; 4)
```

3 Classe Forme

Nous créons maintenant la classe Forme, qui est une classe abstraite. Elle contient deux méthodes virtuelles pures : aire() et perimetre().

Nous créons comme varibale protégée un Point centre permettant ainsi au sous-classes d'y accéder, et pouvoir définir le centre de leur forme.

4 Les formes géométriques

Nous créons deux classes filles de Forme : Cercle et Rectangle. Elles implémentent les méthodes aire() et perimetre().

Nous utilisons les formules mathématiques pour calculer l'aire et le périmètre de chaque forme.

Nous créons une classe fille de Rectangle : Carré. Elle hérite des attributs et méthodes de Rectangle. En effet, un carré est un rectangle avec des côtés égaux.

5 Tests des classes Forme, Cercle, Rectangle et Carré

Nous écrivons des tests pour vérifier le bon fonctionnement des classes Forme, Cercle, Rectangle et Carré.

```
=====
TEST DES DIFFÉRENTES FORMES
=====

--- TEST CERCLE ---
Cercle par défaut (rayon=0, centre=(0,0))
(Cercle, (0; 0), 0)
Cercle avec centre=(5,5), rayon=0
(Cercle, (5; 5), 0)
Cercle avec centre=(10,10), rayon=7.5
(Cercle, (10; 10), 7.5)
Surface: 176.715
Périmètre: 47.1239

--- TEST RECTANGLE ---
Rectangle par défaut (0,0,0,0)
(Rectangle, (0; 0), 0, 0)
Rectangle centre=(0,0), longueur=10, largeur=5
(Rectangle, (0; 0), 5, 10)
Surface: 50
Périmètre: 30
Rectangle centre=(15,15), longueur=8, largeur=6
(Rectangle, (15; 15), 6, 8)
Surface: 48
Périmètre: 28
```

```
--- TEST CARRÉ ---
Carré par défaut (0,0,0)
(Carree, (0; 0), 0)
Carré centre=(0,0), côté=5
(Carree, (0; 0), 5)
Surface: 25
Périmètre: 20
Carré centre=(20,20), côté=8
(Carree, (20; 20), 8)
Surface: 64
Périmètre: 32

=====
                        COMPARAISONS D'AIRES
=====
Cercle (rayon=7.5): 176.715
Rectangle (10x5): 50
Carré (côté=8): 64
```

Nous obtenons les résultats attendus, confirmant le bon fonctionnement des classes.

6 Liste de forme

Je n'ai pas réussi à implémenter cette partie. J'ai cependant créé la structure dans le .h. Nous savons que chaque forme a comme classe mère *Forme*, ainsi j'ai créé un vecteur de *Forme*.