Niveau 3

TP
S4.6. - Programmation orientée objet
Modèle canonique de Coplien
Surcharges d'opérateurs

Formes

Table des matières

1 - PRÉSENTATION

Le but de ce premier TP est la maîtrise des classes au format Coplien, ainsi que l'apprentissage des liaisons d'utilisation et finalement la définition de méthodes virtuelles.

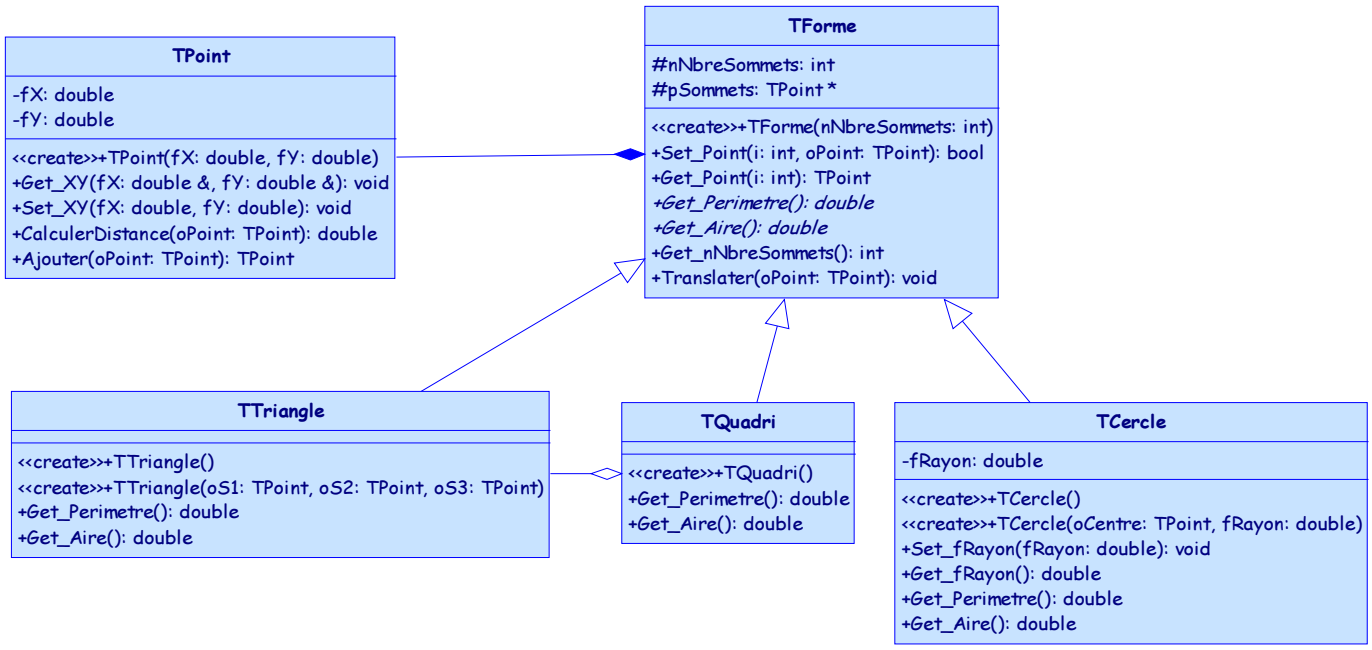
Cette étude porte sur la définition de formes géométriques quelconques.

Au final des deux TPs successifs, nous saurons développer des classes héritées et/ou utilisatrices d'autres classes.

De plus, nous allons redéfinir l'opérateur d'affichage de `ostream` (`cout`) de façon à ce que notre classe personnelle soit affichable plus facilement !

2 - CLASSES ÉTUDIÉES

Voici le diagramme de classes global de notre étude. Les méthodes Colpien ne sont pas indiquées, mais elles seront à coder finalement.



Dans ce premier TP, nous mettrons au point les classes `TPoint` et `TForme`, qui n'ont pas de difficultés algorithmiques.

REMARQUE IMPORTANTE
Vous testerez, grâce au débogueur, chaque méthode que vous développez par quelques lignes de code dans le `main()` AVANT de passer à la question suivante !

3 - TPOINT

Vous ferez bien attention au type de retour des fonctions, et donc de correctement utiliser l'instruction `return` !

Il est possible de reprendre cette classe d'un TP précédent, en l'adaptant aux besoins de ce TP-ci !

3.1 - COPLIEN ET CONSTRUCTEURS

1) Sur un nouveau projet QT console, ajoutez une nouvelle classe `TPoint`.

2) Complétez les fichiers entête et application conformément au diagramme de classe ci-dessus.

3) Ajoutez les méthodes nécessaires (et qui ne sont pas dans le diagramme de classe) afin de définir la classe selon la **forme canonique de Coplien.**

4) Codez le constructeur par défaut de la classe `TPoint`. Le point est défini par défaut aux coordonnées `[0.0 ; 0.0]`.

5) Codez le constructeur de recopie, qui initialise les valeurs de nos attributs avec les valeurs correspondantes de l'objet de même type passé en paramètre.

6) Codez le destructeur qui permet de réinitialiser les valeurs de tous les attributs à des valeurs par défaut !

7) Codez l'opérateur de recopie, qui réinitialise et détruit les attributs de l'objet en cours puis les initialise avec les valeurs des attributs correspondants de l'objet passé en paramètre.

8) Codez le constructeur paramétré.

3.2 - AFFICHAGE

9) Redéfinissons l'opérateur d'affichage de la classe `ostream`, permettant l'affichage du contenu de notre classe au moyen du flux `cout` standard, selon le format suivant :

`Le point (`***`adresse de l'objet courant`***`) est aux coordonnees [ `***`X.X`***` ; `***`Y.Y`***` ]`

3.3 - ACCESSEURS ET MUTATEURS

10) Codez les accesseur et mutateur indirects, `Get_` et `Set_`, des attributs `fX` et `fY` de la classe.

Notez que l'accesseur doit retourner deux valeurs... ce qui n'est pas possible, mais on va contourner cet impératif grâce à des passages de paramètres par référence !

3.4 - MÉTHODES MÉTIER

11) Codez la méthode `CalculerDistance()`, qui calcule la distance entre le point courant (`this->…`) et le `oPoint` passé en paramètre, et qui la retourne.

12) Programmez la méthode `Ajouter()`, qui additionne les coordonnées du point courant avec celles du point passé en paramètre.

4 - TFORME

13) Reprenez toutes les questions précédentes pour coder la classe `TForme` !

4.1 - PARTICULARITÉ DES MÉTHODES DE CETTE CLASSE

14) Bien que la liaison soit une composition, les constructeurs doivent **créer dynamiquement** un tableau de `TPoint`, appelé `pSommets`, afin de représenter les sommets de la forme. Les sommets sont ordonnés dans ce tableau par suite logique du parcours du périmètre : le premier est n'importe lequel de la forme, le second est l'un qui lui est relié par un côté, le troisième est relié au second par un autre côté, etc.
Le constructeur par défaut crée un triangle (forme n'ayant que 3 sommets).
Les autres constructeurs suivent le nombre de sommets qui leur est fourni.

15) Le destructeur doit **désallouer** le tableau des sommets correctement.

16) L'opérateur d'affectation doit aussi désallouer le tableau `pSommets` existant correctement, puis recréer un autre tableau de taille adaptée.

17) L'opérateur d'affichage dans le flux doit aussi afficher les informations de tous les points constituant la forme.

18) La méthode `Set_Point()` positionne le `TPoint` passé en paramètre dans la case `i` du tableau `pSommets` (sauf si l'indice indiqué n'est pas valide !),

19) La méthode `Get_Point()` retourne le `TPoint` qui est positionné dans la case adéquate du tableau `pSommets` (sauf si l'indice indiqué n'est pas valide !).

20) La méthode `Translater()` va décaler tous les points de la forme selon une certaine translation indiquée par le point passé en paramètre !