SMP - TP7

L'objectif de ce TP est de manipuler des classes, de travailler sur la surcharge d'opérateurs et de mettre en œuvre le polymorphisme.

1 CRÉATION D'UNE CLASSE POINT

Écrire une classe pour un point du plan. Outre les attributs d'abscisse et d'ordonnée (de type réel), elle devra comporter :

- des méthodes `translater()` qui prennent en argument respectivement un autre `Point` ou une paire de réels
- un constructeur sans paramètres qui initialise un point à l'origine
- un constructeur qui prend deux paramètres réels
- un constructeur de recopie qui devra prendre en paramètre une référence constante à un Point, c'est-à-dire un `Point const &`. La référence est en réalité un pointeur, mais s'utilise comme si on avait affaire à un objet passé par valeur
- des accesseurs et des mutateurs donnant accès aux valeurs des attributs et permettant de changer ces valeurs

Vous expliciterez la classe `Point` sous forme de diagramme de classes dans votre rapport. Vous écrirez les fichiers `Point.h` et `Point.cpp` ainsi d'un programme principal permettant de tester vos différentes méthodes. Vous présenterez vos tests sous forme de jeux d'essais dans votre compte rendu.

2 SURCHARGE D'OPÉRATEURS

- Surcharger l'opérateur `<<` de façon à permettre l'utilisation de `cout` avec des Points
- Surcharger l'opérateur `+=` pour simplifier l'écriture de la translation d'un point (attention, ne peut s'appliquer quentre Points)

Illustrez le bon fonctionnement de vos surcharges d'opérateurs à travers des jeux d'essais.
Remarque : l'opérateur `+=` sera défini comme membre de la classe Point et l'opérateur `«` sera défini comme externe à la classe.

3 Formes géométriques abstraites

On considère des formes géométriques fermées centrées sur un point particulier.

1. Écrire une classe `Forme` qui aura ce point comme attribut avec le constructeur approprié.
2. Surcharger l'opérateur `+=` pour permettre la translation de la forme
3. Surcharger l'opérateur `«` de façon à permettre l'utilisation de `cout` avec des Formes
4. Déclarer deux méthodes abstraites `perimetre()` et `surface()`

4 Formes géométriques concrètes

Écrire au moins les classes Cercle, Rectangle et Carré (Carré héritant elle-même de Rectangle) classes de formes géométriques concrètes dérivant de la classe Forme. Les Rectangles et les Carrés auront dans un premier temps leurs côtés alignés avec la verticale et l'horizontale.
Elles devront être écrites dans des fichiers séparés et définir les méthodes abstraites.
Surchargez l'opérateur `«` pour chacune d'entre elles pour afficher son type et les valeurs des attributs.
Vous montrerez le bon fonctionnement de ces classes à travers des jeux d'essais et représenterez les des dépendances entre ces classes sous forme de diagramme de classes dans votre rapport.

5 Liste de formes

À l'aide de la structure de données `vector<>` `https://www.cplusplus.com/reference/vector/vector/`, définissez une classe `ListeFormes` qui outre les opérations classiques de consultation ou d'ajout de formes dans la liste permet de :

— calculer la surface totale de la liste
— calculer la boîte englobante de la liste de formes (le plus petit rectangle aligné sur les axes qui contient l'ensemble des formes de la liste)

Vous montrerez le bon fonctionnement de ces classes à travers des jeux d'essais.
Vous déposerez vos comptes rendus sur hippocampus et vos codes sur github classroom.