Programmation orientée objet (C++) – ING2-GSI

TP1 : Les bases de l'orienté objet

Objectifs

- Comprendre la notion de classe et d'objet ;
- Savoir utiliser une classe ;
- Avoir un premier exemple de classe C++.

1 – Fraction

Le but de ces exercices est d'écrire une classe Fraction simplifiée et de l'enrichir progressivement.

Exercice 1.1 : La classe Fraction

Définir une première version de la classe Fraction possédant deux attributs « numerateur » et « denominateur » qui correspondent respectivement au numérateur et au dénominateur de la fraction. On définira également deux fonctions. La première s'appelle « afficher » et permet d'afficher la fraction à l'écran. La seconde s'appelle « set » et permet d'initialiser la fraction à partir de deux entiers correspondant respectivement au numérateur et au dénominateur.

Toutes les caractéristiques (attributs et fonctions) de cette première classe seront définies publiques.

Remarque : Aucune contrainte n'est imposée sur le numérateur et le dénominateur de la fraction.

Il n'est en particulier pas demandé de normaliser la fraction.

Exercice 1.2 : Programme de test

Écrire un programme de test qui permet de vérifier la classe Fraction ainsi écrite.

Remarque : Il est conseillé de construire le programme de test en même temps que la classe. Ceci permet de tester les fonctions au fur et à mesure de leur écriture. Les erreurs sont alors détectées plus tôt et sont donc plus faciles à localiser et à corriger.

Exercice 1.3 : Appliquer le principe de protection en écriture des attributs

Il est recommandé de toujours déclarer les attributs en privé. Le premier intérêt est pour le programmeur de la classe. Il peut ainsi garantir que l'état des objets de la classe sera toujours

cohérent car les attributs ne pourront pas être modifiés sans passer par une fonction qu'il aura définie (fonction d'altération ou de modification).
Le second intérêt est pour les utilisateurs de la classe. Ne pouvant pas accéder aux attributs de la classe, ils doivent donc passer par les fonctions d'accès et sont ainsi moins dépendants du choix d'implantation de la classe.
Modifier la classe Fraction et le programme de test pour que les attributs soient privés.

Exercice 1.4 : Compléter la classe Fraction
Ajouter les deux fonctions suivantes sur la classe Fraction :

- opposer est une fonction qui change le signe de la fraction ;
- opposée est une fonction qui retourne une nouvelle fraction qui est l'opposée de la fraction sur laquelle elle est appliquée. Cette dernière fraction reste inchangée.

Exercice 1.5 : Normaliser les fractions
Compléter la classe Fraction de manière à ce que les fractions soient conservées sous forme normalisée : le numérateur et le dénominateur sont sous forme réduite, le signe de la fraction est porté par le numérateur et la fraction nulle est représentée par 0/1.

2 - Le compteur

Un compteur est défini par les propriétés suivantes :

- Il possède une valeur entière, positive ou nulle, à sa création.
- Il ne peut varier que par pas de 1 (incrémentation ou décrémentation). On admettra que décrémenter un compteur nul laisse ce dernier à 0.

Ecrire une classe Compteur sur ce cahier des charges simple.

Ensuite, écrire un programme permettant de tester ce compteur comme suit :

- D'abord, le programme crée un compteur et affiche sa valeur.
- Ensuite, il incrémente 10 fois ce compteur avant d'afficher à nouveau sa valeur.
- Enfin, il décrémente 20 fois ce compteur, et il affiche une dernière fois sa valeur.

Que doit rendre le programme en sortie ?

3 – Les cercles

Ecrire une classe qui permet de gérer des points. Cette classe doit contenir :

- Une abscisse et une ordonnée réelle.
- Des méthodes permettant de retourner ou de renvoyer cette abscisse et cette ordonnée.

Ensuite, écrire une classe qui permet de gérer des cercles. Cette classe doit savoir :

- créer des cercles de centre et de rayon donnés (le centre doit être un objet de la classe Point définie précédemment),
- renvoyer une copie de son rayon et de son centre,
- calculer son périmètre,
- calculer son aire,
- déterminer s'il chevauche un autre cercle.

Ecrire un programme qui permet de tester les fonctionnalités spécifiées.