TP II. CLASSE, CONSTRUCTEURS, DESTRUCTEURS, SURCHARGE DES OPÉRATEURS

Exercice 1

Définir une classe point permettant de manipuler un point d'un plan. On prévoira :

- un constructeur recevant en arguments les coordonnées (float) d'un point ;
- deux fonctions membres : getAbscisse et getOrdonnee fournissant respectivement l'abscisse et l'ordonnée d'un point
- une fonction membre translater effectuant une translation définie par ses deux arguments (float) ;
- une fonction membre afficher se contentant d'afficher les coordonnées d'un point.
- une fonction distance qui permet de retourner la distance séparant le point courant à un second point qui est indiqué par ces coordonnées à la fonction.

La déclaration de la classe doit se faire dans un fichier d'entête et sa définition dans un fichier source.
Écrire dans un autre fichier source un petit programme de test (main) déclarant un point, l'affichant, le déplaçant et l'affichant à nouveau.

Exercice 2

Écrivez un programme dans lequel vous définissez une classe Cercle ayant comme attributs un rayon et les coordonnées d'un point qui représente le centre du cercle. Déclarez ensuite les méthodes «get» et «set» correspondantes.
Ajoutez ensuite les méthodes :

- double surface() const : qui calcule et retourne la surface du cercle (pi fois le carré du rayon);
- bool estInterieur() const : qui teste si le point de coordonnées (x,y) passé en paramètre fait partie ou non du cercle (frontière comprise : disque fermé).

Dans le main(), instanciez deux objets de la classe Cercle, affectez des valeurs de votre choix à leur attributs et testez vos méthodes surface et estInterieur.

Exercice 3

Créer une classe appelée Article qui possède une donnée membre: numSerie contenant le numéro de série de chaque article. En effet, le premier article créé doit avoir le numéro de série égale à 1, le second 2, etc. Pour cela, vous aurez besoin d'une autre donnée membre nombreArticle qui enregistre le nombre d'articles qui ont été créés et celle ci doit être commune à tous les objets de la classe. Tester la classe Article.

Exercice 4

soit la classe Complexe définie par le code suivant:

```
class Complexe {
public:
Complexe(const double re = 0.0, const double im = 0.0);
Complexe(const Complexe&);
~Complexe();
double getReel() const;
void setReel(double);
double getImag() const;
void setImag(double);
Complexe& operator=(const Complexe &);
```

```
Complexe& operator+=(const Complexe&);
bool operator==(const Complexe &) const;
private:
double* re;
double* im;
};
```

Définir les 2 fonctions non membres de la classe Complexes:

✦ Surcharger les opérateurs << et >> qui permettent respectivement d'afficher et de lire un complexe.

✦ Surcharger l'opérateur + qui retourne la somme de deux complexes.

Exercice 5

Le but de cet exercice est de simuler de façon très basique la gestion d'une bibliothèque. La bibliothèque contient des exemplaires d'œuvres écrites par des auteurs. Il s'agira de modéliser chacun de ces éléments dans votre programme.

Le code fourni crée des auteurs, des œuvres de ces auteurs, stocke dans la bibliothèque des exemplaires de ces œuvres, liste tous les exemplaires de la bibliothèque, affiche le nom de tous les auteurs ayant écrit une œuvre dont la bibliothèque stocke un exemplaire et affiche le nombre d'exemplaires d'une œuvre donnée. Pour cela, vous aurez besoin de définir les 4 classes: Auteur, Oeuvre, Exemplaire et Bibliotheque.

1. **La classe Auteur-** Un auteur est caractérisé par son nom(string) ainsi qu'une indication permettant de savoir s'il a été primé. Les méthodes qui sont spécifiques à cette classe et font partie de son interface d'utilisation sont :

✦ des constructeurs avec l'ordre suivant pour les paramètres : le nom et l'indication permettant de savoir si l'auteur a été primé. Par défaut un auteur n'est pas primé ;

✦ une méthode getNom retournant le nom de l'auteur ;

✦ une méthode getPrix retournant true si l'auteur a été primé.

2. **La classe Oeuvre-** Une Oeuvre est caractérisée par son titre (string), son auteur(référence constante à un auteur) qui l'a rédigée et la langue dans laquelle elle a été rédigée (string). Les méthodes qui sont spécifiques à cette classe et font partie de son interface d'utilisation sont :

✦ des constructeurs avec l'ordre suivant pour les paramètres : le titre, la référence à l'auteur et la langue;

✦ une méthode getTitre retournant le titre de l'œuvre ;

✦ une méthode getAuteur retournant une référence constante sur l'auteur ;

✦ une méthode getLangue retournant la langue de l'œuvre ;

✦ et une méthode affiche affichant les caractéristiques de l'œuvre en respectant strictement le format suivant : <titre>, <nom de l'auteur>, en <langue>

✦ un destructeur affichant un message respectant strictement le format suivant :
L'oeuvre "<titre>, <nom de l'auteur>, en <langue>" n'est plus disponible.

3. **La classe Exemplaire-** La classe Exemplaire modélise les exemplaires d'une œuvre. Une instance de cette classe est caractérisée par une référence à l'œuvre dont elle constitue un exemplaire. Les méthodes spécifiques à la classe Exemplaire et qui doivent faire partie de son interface d'utilisation sont :

✦ un constructeur prenant en argument une référence à une œuvre et affichant un message dont le format : Nouvel exemplaire : <titre>, <nom de l'auteur>, en <langue> ;

✦ un constructeur de copie affichant un message respectant le format suivant : Copie d'un exemplaire de : <titre>, <nom de l'auteur> en <langue>

✦ d'un destructeur affichant un message respectant strictement le format suivant : Un exemplaire de "<titre>, <nom de l'auteur>, en <langue>" a été jeté !
✦ une méthode getOeuvre retournant une référence constante à l'œuvre ;
✦ et une méthode affiche affichant une description de l'exemplaire respectant strictement le format suivant : Exemplaire de : <titre>, <nom de l'auteur>, en <langue> sans saut de ligne.

4. **La classe Bibliotheque -** Une bibliothèque est caractérisée par un nom et contient un ensemble de pointeurs sur des exemplaires. L'ensemble sera modélisé au moyen d'un tableau. Les méthodes spécifiques à la classe Bibliotheque et qui font partie de son interface d'utilisation sont :

✦ un constructeur conforme affichant le message : La bibliothèque <nom> est ouverte !suivi d'un saut de ligne, où <nom> est à remplacer par le nom de la bibliothèque ;
✦ une méthode getNom retournant le nom de la bibliothèque ;
✦ une méthode stocker permettant d'ajouter un ou plusieurs exemplaires d'une œuvre dans la bibliothèque ; elle doit respecter l'ordre suivant des paramètres : la référence à une œuvre et le nombre n d'exemplaires à ajouter, dont la valeur par défaut est 1 ; cette méthode va ajouter à l'ensemble d'exemplaires de la bibliothèque n exemplaires de l'œuvre fournie, dynamiquement alloués ; les nouveaux exemplaires devront impérativement être ajoutés à la fin du tableau dynamique ;
✦ une méthode lister_exemplaires affichant tous les exemplaires d'une œuvre écrite dans une langue donnée ; si aucune langue n'est donnée (chaîne vide), tous les exemplaires de la bibliothèque seront affichés ; les exemplaires devront être affichés au moyen de la méthode d'affichage qui leur est spécifique et il y aura un saut de ligne à la fin de l'affichage de chaque exemplaire;
✦ une méthode compter_exemplaires retournant le nombre d'exemplaires d'une œuvre donnée passée en paramètre ;
✦ une méthode afficher_auteurs prenant en paramètre un booléen (valant par défaut false) indiquant si l'on veut afficher uniquement les auteurs à prix ;
✦ un destructeur qui affiche le message suivant : La bibliothèque <nom> ferme ses portes, et détruit ses exemplaires :

Exemple d'exécution:

```
La bibliothèque nationale est ouverte !
Nouvel exemplaire de : Les Misérables, Victor Hugo, en français
Nouvel exemplaire de : Les Misérables, Victor Hugo, en français
Nouvel exemplaire de : L'Homme qui rit, Victor Hugo, en français
Nouvel exemplaire de : Le Comte de Monte-Cristo, Alexandre Dumas, en français
Nouvel exemplaire de : Le Comte de Monte-Cristo, Alexandre Dumas, en français
Nouvel exemplaire de : Le Comte de Monte-Cristo, Alexandre Dumas, en français
Nouvel exemplaire de : The Count of Monte-Cristo, Alexandre Dumas, en anglais
```