esprit Se former autrement	Programmation Orientée Objet C++
	TD 7 : Surcharge des opérateurs

Objectifs :

- Utilité et mécanisme de surcharge des opérateurs en C++
- Surcharge de quelques opérateurs binaires (+=, +, -=, -, <, >, ...)
- Techniques de surcharge d'un opérateur donné : méthode d'instance / fonction amie
- Surcharge de l'opérateur d'affectation
- Surcharge des opérateurs d'entrée/sortie standards

Exercice 1 : Gestion de stock

Définir la classe Produit pour l'application de gestion de stock, ayant les attributs privés suivants :

— Ref : entier
— Libellé : chaîne de caractères
— Quantité en stock : entier
— Prix unitaire : réel

Et les opérations publiques suivantes :

— ChangerQuantité
— ChangerPrix

L'opérateur :

— + : fusionne deux produits en un seul ayant la référence du premier, le libellé concaténation de deux libellés, la quantité en stock la somme des deux quantités et le prix pondéré par les deux quantités :

Nouveau prix = (prix1*quantité1+prix2*quantité2)/(quantité1+quantité2)

Les constructeurs :

— Ayant comme paramètre la référence et le libellé du produit
— Sans paramètres : référence 0 et libelle inconnu

Et une fonction amie permettant l'affichage du produit.

Ecrire un programme qui :

— Déclare 3 produits :

le 1er :120, verre V225, 12000, 0.125D
le 2e :125, verre V220, 50000, 0.100D

— Additionne les 2 produits dans le 3ème produit
— Affiche les 3 produits

Exercice 2 Gestion de Zoo

Un Zoo désire réaliser une application afin de mieux s'organiser et gérer l'ensemble des animaux et l'ensemble des médecins vétérinaires.

Un animal est caractérisé par son espèce, sa matricule **unique** et une collection contenant les dates de ses vaccins (string).

Un lion est un animal caractérisé en plus par la quantité de viande consommée.

Un médecin vétérinaire est caractérisé par un identifiant **unique**, un nom, un prénom et une collection contenant les matricules des animaux qu'il prend en charge.

La classe «Zoo» est caractérisé par une adresse, un ensemble de médecins vétérinaires et un ensemble d'animaux. Elle répond aux besoins suivants :

1. Ajouter un animal ou un lion.
2. Ajouter une date de vaccination pour un animal donné par sa matricule (vérifier l'unicité de la date).
3. Ajouter un médecin vétérinaire.
4. Affecter la matricule d'un animal à un médecin vétérinaire. Lever une exception si :
   - L'animal n'existe pas.
   - Le médecin n'existe pas.
   - L'animal est déjà affecté à ce médecin vétérinaire.
5. Afficher le nombre total des animaux pris en charge par les médecins vétérinaires (surcharger l'opérateur += dans la classe médecin vétérinaire en retournant un objet contenant la concaténation des deux collections des matricules des animaux).
6. Enregistrer dans un fichier texte la liste des lions qui consomment une quantité de viande supérieure à un seuil bien déterminé.

On vous demande d'implémenter un programme principal (main) qui permet de :

7. Instancier des objets de toutes les classes implémentées.
8. Tester l'ensemble des fonctionnalités demandées.

Exercice 3 Gestion de parkings

On désire réaliser une application pour la gestion des parkings d'une entreprise, via une application C++.

L'accès à un parking nécessite une carte magnétique caractériser par : identifiant unique, nombre d'heures de stationnement, prix de l'heure.

Un abonnement, est une carte magnétique caractérisée en plus par la date d'expédition et une gratuité d'accès pour les 10 premières heures.

Un parking est caractérisé par une référence unique, le nombre de places totales, l'ensemble des identifiants des cartes magnétique autorisées (sans redondance) et l'ensemble des identifiants des cartes magnétique utilisés pour accéder au parking (sans redondance).

L'application gère l'ensemble des parkings et des cartes et abonnements. Elle répond aux besoins suivants :

1. Ajouter un parking, une carte d'accès ou un abonnement.
2. Ajouter l'identifiant d'une carte à un parking pour y accéder. Vérifier que la carte est autorisé, qu'il y a encore de places libres et que la carte n'a pas été utilisé pour accéder au parking.
3. Trouver le parking le moins rentable : celui dont le taux de remplissage est le plus faible. Utiliser la surcharge de l'opérateur <.
4. Supprimer une carte magnétique sachant son identifiant. Il est à rappeler que la carte doit être aussi retirée de tous les parkings.

On vous demande d'implémenter un programme principal (main) qui permet de :

1. Instancier des objets de toutes les classes implémentées.
2. Tester l'ensemble des fonctionnalités demandées.