Université Internationale de Casablanca LAUREATE INTERNATIONAL UNIVERSITIES Nous innovons pour votre réussite !	**Ecole d'Ingénieurs** **Filières :** MIAGE **Classe :** 2ème année - S3	**Cours : POO, C#** **Professeur** : MOUJAHID Abdallah

Travaux Pratiques N° 4

La POO avec le langage C#

Exercice N° 1 – Les collections

Soit à développer une application pour la gestion des employés d'une entreprise.

Un employé est caractérisé par son code, son nom, son prénom, son salaire brut, son grade et son Taux IGR avec :

- Grade est un entier compris entre 16 et 22
- TauxIGR ne peut prendre que les valeurs 0,42 ou 0,35

1. Créer la classe employé (constructeurs, Accesseurs et la méthode Tostring())
2. Créer les exceptions GradeException et TauxIgrException
3. Créer la méthode double CalculerSalaireNet avec: Salaire net= salaire brut * (1-TauxIGR)

L'entreprise se caractérise par sa raison sociale, son capital et des employés

4. Ecrire la classe Entreprise (attributs, Accesseurs, constructeurs, la méthode ToString())
5. Ajouter les méthodes suivantes:
    a. Ajouter un nouveau employé
    b. Afficher tous les employés de l'entreprise
    c. Rechercher un employé et afficher ses informations ou un message de non existence
    d. Supprimer un employé par son code.
    e. Modifier (nom, prénom, salaire) d'un employé donné
    f. MasseSalariale : retourne la masse salariale de l'entreprise (somme des salaires net)
6. Ecrire un programme de test qui lance le menu suivant pour gérer les employés
    a. Ajouter Employé
    b. Modifier les informations d'un employé
    c. Supprimer un employé
    d. Rechercher les informations d'un employés
    e. Afficher tous les employés avec leurs nombre
    f. Afficher la masse salariale de l'entreprise
    g. Afficher l'employé qui a le plus grand salaire

Exercice N° 2 – Héritage, Polymorphisme, abstraction

Un garage vend des voitures; chaque voiture possède un numéro d'immatriculation ainsi qu'un prix de vente de base. Deux types de voitures sont proposés:

- Des voitures d'occasion qui ont un certain kilométrage et une ancienneté en années. Le prix de base est modifié selon la règle suivante : le kilométrage normal est de 20000 km par an ; le prix est majoré ou minoré à raison de 0.05 euros par km inférieur ou supérieur.
- Des voitures neuves qui peuvent être vendues avec des options - climatisation, airbag, Toit ouvrant, bleutooth... – on enregistre le prix de l'option. Le prix des options se rajoute au prix de base.

  NB : prix=-1 explique que l'option n'existe pas

Le garage propose ainsi un parc d'automobiles à la vente, et des éventuelles options pour les voitures neuves.

L'objectif de l'application est de simuler des ventes de voitures.

1. Sachant qu'une voiture sera obligatoirement neuve ou d'occasion (on ne peut pas créer d'objet « Voiture » , on ne peut créer que des objets « Voiture neuve » ou « Voiture d'occasion »), implémentez la classe abstraite « Voiture » avec son constructeur par défaut et d'initialisation et l' accesseur « Numero ».
2. Essayez de créer une voiture. Que se passe-t-il ?
3. Ajoutez à votre classe « Voiture » une méthode « getPrix » qui retourne le prix du véhicule. Compte tenu de la différence de règles de calcul entre les voitures d'occasion et les voitures neuves, nous ne pouvons pas calculer le prix dans la classe « Voiture ».

   La méthode « getPrix » de la classe « Voiture » doit donc être une méthode abstraite.
4. Créer la classe « Occasion » qui hérite de voiture avec son constructeur par défaut et d'initialisation et 2 accesseurs permettant de retourner le nombre de kilomètres et l'age de la voiture en années (lecture seule)
5. Essayez de créer une voiture d'occasion. Que se passe-t-il ?
6. Redéfinir la méthode « getPrix » dans votre classe « Occasion » en tenant compte de la règle énoncée au début du TP.
7. Ajouter une méthode « toString » à votre classe « Voiture » qui retourne le numéro d'immatriculation
8. Ajouter une méthode « toString » dans la classe « Occasion » qui retourne toutes les informations d'une voiture occasion séparées par des espaces.
9. Créer la classe «Neuve » qui hérite de voiture avec son avec son constructeur par défaut et d'initialisation et accesseurs
10. Redéfinir la méthode « getPrix » qui retourne le prix de la voiture en tenant compte de toutes ses options.

11. Ajoutez une méthode « toString » dans la classe « Neuve » qui retourne toutes les informations d'une voiture neuve séparées par des espaces.
12. testez en créant une voiture neuve et une voiture occasion en affichant les informations et le prix (Pour une voiture neuve afficher la liste des options disponibles)

Exercice N° 3 – La gestion des Exceptions

Un centre de formation professionnelle souhaite mettre en place un système de gestion de parrainage des stagiaires.
Soit la classe **Formateur** qui comporte les attributs privés suivants :

- codeFormateur int
- nom String
- prénom String
- sexe char ('M','F')
- âge int
- spécialité String

1. Ecrire la classe Formateur.
2. Ajouter un compteur qui permet de compter le nombre des objets créés de cette classe
3. Ajouter un constructeur sans argument qui initialise l'attribut codeFormateur de la classe Formateur. (le code doit avoir la valeur du compteur).
4. Ajouter un constructeur qui initialise tous les attributs de la classe Formateur
5. Ajouter une méthode Affichage() qui afficher toutes les informations d'un formateur.
6. Ajouter une Exception nommée CodeException qui s'exécute si le code formateur est négatif
7. Ajouter une Exception nommée SexeException qui s'exécute si le sexe formateur n'est pas M ou F
8. Ajouter une Exception nommée NomException qui s'exécute si le nombre de caractères du nom formateur est inférieur à 3

Exercice N° 4 – La gestion des Exceptions (suite de l'exercice 5)

Soit la classe **Vacataire** qui modélise un formateur vacataire qui comporte, en plus des attributs de la classe Formateur, les attributs suivants :

- NbHeure int
- SalaireHoraire Réel

1. Ecrire la classe Vacataire.
2. Ajouter un compteur qui permet de compter le nombre des objets créés de la classe Vacataire
3. Ajouter un constructeur sans argument qui initialise tous les attributs de cette classe.
4. Modifier la méthode Affichage() pour prendre en compte les attributs supplémentaires de cette classe.
5. Ajouter une Exception nommée SalaireException qui s'exécute si le salaireHoraire est < à 200DH
6. Ajouter une Exception nommée NbHeureException qui s'exécute si le nombre d'heures est négatif ou supérieur à 8