
Exercice1: Dans le cadre de l'informatisation d'une entreprise, un directeur souhaite automatiser la gestion des salaires de ses employés pour cela on aura :

-Une classe Employe : Un employé est caractérisé par :

+*ces attributs :* nom(chaine de caractères), son prénom(chaine de caractères) et son âge(entier),son année de recrutement(entier)

+*Ces méthodes :* -Un constructeur a quatre paramètres permettant d'initialiser une instance de la classe Employe.

+Affiche() :qui permet d'afficher les quatre attributs de la classe Employe.

+CalculSalaire() :permet de renvoyer le salaire mensuel d'un employé mais ce calcul dépend du type de l'employé. On distingue les types d'employés suivants qui seront définie dans les classes suivantes:

-Producteur : Leur salaire vaut le nombre d'unités produites mensuellement multipliées par 5 :

Ajouter l'attribut NbUnités de type entier ;Définir la méthode CalculSalaire() ;Redéfinir la méthode Affiche( )

-Commercial : Leur salaire mensuel se calcule en fonction du chiffre d'affaire qu'ils réalisent :

Ajouter l'attribut ChiffreAffaire de type double ; Redéfinir la méthode Affiche() ,aussi empêcher la redéfinition de la méthode Affiche dans les classes dérivées

- La classe Commercial dérive la classe **Vendeur** et la classe **Représentant** tel que leur salaire est le 20 % du chiffre d'affaire qu'ils réalisent mensuellement, plus respectivement 400DH et 800DH : Définir la méthodes CalculSalaire() .

-Ajoutez une classe **Personnel** contenant un tableau d'employés. Il s'agira d'une collection polymorphique d'Employe contenant ces méthodes :

+void **ajouterEmploye(Employe)** : qui ajoute un employé au tableau

+void **show( )** : qui affiche tous les employes du Personnel.

+double **salaireMoyen( )** :qui affiche le salaire moyen des employés de l'entreprise.

+void **NbEmployes()** :qui affiche le nombre de employés instanciées pour chaque d'employées

Exemple d'exécution de cette méthode :Dans votre entreprise vous avez 3 représentant,2vendeur et 2 producteur. Pour cela il faut ajouter les attributs nécessaires dans les classes précédentes.

Exercice2: Créer une version graphique de l'exercice1

La version graphique de l'exercie1:

Effectuer les modifications suivantes:

-Classe Employée:

Affiche():méthode qui retourne une description de l'employée;

-Classe Personnel:

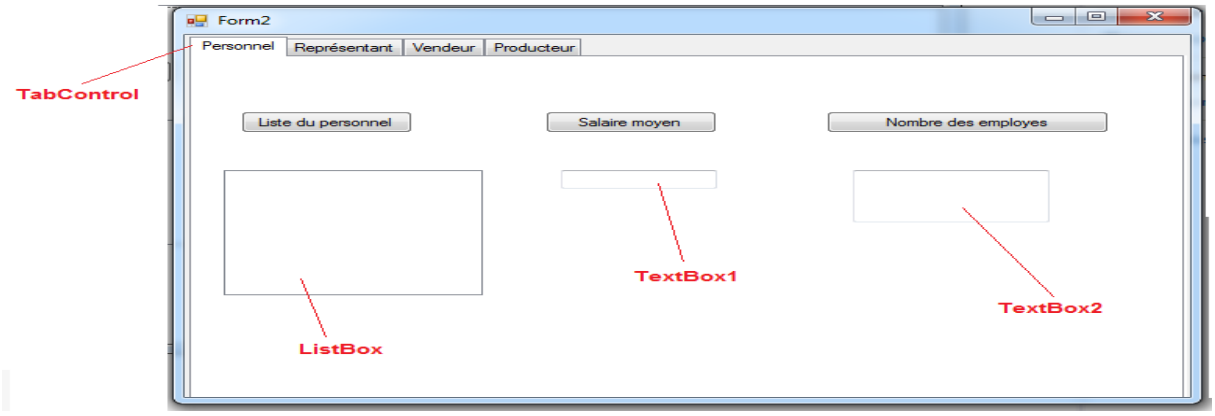
List<string> show():retourne la description de tous les employées

- **Figure1**

-Button "Liste du personnel":affiche dans la liste déroulante une description des personnels de l'entreprise

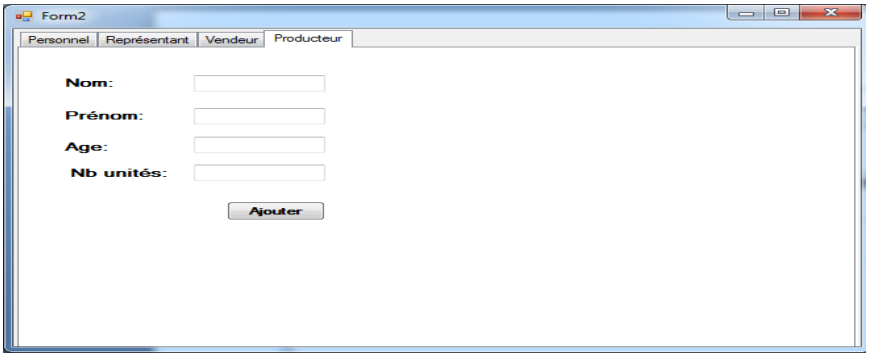
-Button "salaire moyen": affiche dans le TextBox le salaire moyen de tous les employées

-Button "nombre des employés":affiche dans le 2ème TexteBox le nombre de chaque type des employée



- **Figure2:**

Le click sur les onglets du TabControl permet d'ajouter des employées: le formulaire doit contenir les champs selon le type de l'employée



Exercice3: - Ecrire une classe Personne contenant les champs nom, prenom et âge et qui implémente l'interface IComparable. List<T> is equivalent of Vector in C #

-Ecrire une classe TestPersonne1 qui permet de définir un vecteur de personne,Trier ce vecteur par ordre alphabétique de nom ,afficher les éléments de ce vecteur,rechercher un élément dans ce vecteur.

Exercice4: - Ecrire une classe Personne contenant les champs nom, prenom et âge et qui implémente l'interface IComparer. List<T> is equivalent of Vector in C #

-Ecrire une classe TestPersonne2 qui permet de définir un vecteur de personne,Trier ce vecteur par ordre alphabétique de nom et prénom, ou à partir de personne plus jeune ,le choix de critère de tri est choisi par l'utilisateur. afficher les éléments de ce vecteur,rechercher un élément dans ce vecteur et extraire la personne la plus jeune.

Exercie5: Même exercice3 mais cette fois utiliser les query collection

Exercice6: L'objectif est de réaliser un programme permettant de créer un ensemble de villes, sachant que toutes les villes auront des noms différents.Dans un premier temps, on développe deux classes : une classe Ville, une classe EnsembleVille contenant un dictionnaire de Ville dont la clé sera le nom de la ville. Dans un deuxième temps, on crée un programme permettant d'utiliser ces classes.

Classe Ville	Classe EnsembleVille

public class Ville
{
private String nomVille;
private String codeDepartement;
private String codeCommune;
 	
//Constructeur
public Ville(String pNomVille, String pCodeDepartement, String pCodeCommune)
	public class EnsembleVille
	{
	private Dictionary<String, Ville> lesVilles;
	
	//retourne l'objet Ville dont le nom est passé en paramètre, null si le nom n'est pas trouvé
	public Ville getUneVille(String unNomVille)
	
	// si une ville du même nom est déjà présente indiquer cela à l'utilisateur
	public void ajouterVille(Ville uneVille)
	
	//Renvoie le code postal de la ville dont le nom est passé en paramètre, "n'existe pas" si la ville n'a pas été trouvée
	public String getCodePostal(String unNomVille)
	
	//Affiche l'ensemble par ordre croissant des noms de ville
	public void AfficherEnsemble()
	}

Exercice7: Ecrire un programme qui permet de gérer une pile en intégrant deux méthodes:Methode Piler pour ajouter un élément au pile;Dépiler pour supprimer un élément.A savoir cette pile devra

générer n'importe quel type de données. Cette pile peut être représentée par un tableau qui va contenir des éléments.

-Créer la classe générique Pile en intégrant un tableau de taille max=100.Ce tableau pourra être de n'importe quel type(int,string,float ...).Ajouter à la classe un attribut auxiliaire représentant un "pointeur" sur l'index courant;

-Créer la méthode Dépiler qui permet de retirer le dernier élément de notre pile ensuite retourner l'élément sorti de la pile.On lance une exception InvalidOperationException si on essaye de retirer un élément d'une pile vide

-Créer la méthode Piler permettant d'ajouter un élément à notre pile. on lève une exception StackOverflowException si on atteint la taille maximale.

-Définir une méthode Affiche qui permet d'afficher les éléments du tableau.

-Tester les méthodes de la classe Pile.

Exercice8:

1.Créez les quatre méthodes de calcul: (soustraction, addition, division et multiplication) qui prennent en paramètres deux entiers et retournent le résultat.

2.Créez une méthode calculatrice qui affiche le résultat de calcul de deux entiers après l'opération introduite par l'utilisateur(soustraction, division...).La méthode doit être bien optimisé.

3.Ajouter à la méthode précédente la possibilité d'exécuter l'instruction suivante entre deux entiers x,y (x*y*10) sans écrire cela dans une méthode indépendante