Formateur : LAMOURI Najib

Les collections :

Exercice 1:

a. Définir une classe Employé caractérisée par le nom et l'année de naissance, l'année de recrutement et le salaire, définir le constructeur d'initialisation pour cette classe, et les accesseurs des propriétés, définir les méthodes ToString et Equals pour cette classe.

b. Définir une classe Program (en mode Console) dans lequel vous créez une collection d'employés à l'aide de la classe ArrayList, le module propose à l'utilisateur le menu suivant :

1. Pour ajouter un employé
2. Pour rechercher un employé
3. Pour supprimer un employé
4. Pour effacer tous les employés
5. Pour afficher la liste des employés
6. Pour quitter le programme

Exercice 2:

a. Définir une classe Elève caractérisée par le nom, l'âge et trois notes rangées dans un tableau. Cette classe implémente les interfaces IComparable. Définir le constructeur d'initialisation pour cette classe, et les accesseurs des propriétés, définir une méthode moyenne qui retourne la moyenne de l'objet en cours, définir les méthodes ToString, Equals pour cette classe, ainsi que la méthode CompareTo qui implémente IComparable.CompareTo et qui compare les élèves en terme de moyenne.

b. Définir une classe CElève qui contient un objet col de type List<Elève> et qui implémente l'interface IEnumerable. Définir un constructeur par défaut de cette classe. Ainsi que :

1. La méthode **public void Add(string nom, int age, double[] note)** qui permet de créer un objet de type Elève avec les valeurs passées en argument et de l'ajouter à la collection **col.**
2. La méthode **public void Clear()** qui permet d'effacer tous les éléments de la collection col.
3. La propriété **public int Count** permettant d'obtenir le nombre d'éléments stockés dans la collection col.
4. L'indexeur **public Elève this[int index]** qui permet d'obtenir et de définir l'élément qui se trouve à l'indice **index** de la collection.
5. La méthode **public int IndexOf(Elève e)** qui retourne l'indice de l'élève passé en paramètre dans la collection ou -1 si l'élève n'existe pas dans la collection.
6. La méthode **public void Delete(int index)** qui supprime l'élève qui se trouve à l'indice **index**.
7. La méthode **public void Sort()** qui réalise le tri des élèves par ordre de mérite (par moyenne décroissante).
8. La méthode **public IEumerator GetEnumerator()** qui implémente IEnumerable.GetEnumerator et qui retourne l'énumerateur de la collection des élèves.

c. Créer une classe Program permettant d'afficher un menu à l'utilisateur proposant de réaliser les opérations suivantes :

1. Pour ajouter un élève.
2. Pour rechercher un élève par son nom
3. Pour supprimer un élève par son nom
4. Pour effacer tous les élèves
5. Pour afficher la liste des élèves
6. Pour quitter le programme

Exercice 3: On désire réaliser un annuaire téléphonique. Pour cela on utilise une collection de type SortedList dont les clés sont des objets de type Personne, et les valeurs sont des chaines de caractères contenant les numéros de téléphone.

a. Définir la classe Personne caractérisée par le nom et le prénom, définir le constructeur d'initialisation pour cette classe, et les accesseurs des propriétés, définir les méthodes ToString et Equals pour cette classe. Implémenter la classe IComparable et définir la méthode CompareTo qui compare deux personnes en termes de nom.

b. Définir une classe Annuaire qui contient un objet col de type SortedList et qui implémente l'interface IEnumerable. Définir un constructeur par défaut de cette classe. Ainsi que :

1. La méthode **public void Ajouter(string nom, string prénom, string Tel)** qui permet de créer un objet de type Personne avec les valeurs passées en argument et de l'ajouter à la collection **col** avec le numéro de téléphone Tel**.**
2. La méthode **public void Vider()** qui permet d'effacer tous les éléments de la collection col.
3. La propriété **public int Nombre** permettant d'obtenir le nombre d'éléments stockés dans la collection col.
4. La méthode **public bool ContientClé(string nom, string prénom)** qui détermine si la clé constituée du nom et prénom passés en argument existe dans la collection.
5. La méthode **public bool ContientValeur(string Tel)** qui détermine si le numéro de téléphone passé en argument existe dans la collection.
6. L'indexeur **public string this[Personne p]** qui permet d'obtenir et de définir le numéro de téléphone correspondant à la clé passés en argument.
7. La méthode **public void Supprimer(string nom, string prénom)** qui supprime l'élément correspondant à la clé constituée du nom et prénom passés en argument

c. Créer une classe Program permettant d'afficher un menu à l'utilisateur proposant de réaliser les opérations suivantes :

1. Pour ajouter un contact.
2. Pour rechercher un contact par son nom et prénom
3. Pour supprimer un contact par son nom et prénom
4. Pour effacer tous les contacts
5. Pour modifier un contact
6. Pour quitter le programme

Exercice 4: On désire réaliser une application de gestion d'un service de salariés. Pour cela on utilise une collection de type dictionnaire fortement typé dont les clés sont les matricules de type string, et les valeurs sont les salariés.

a. Définir la classe Salarié caractérisée par le nom le salarié et l'année de recrutement et le salaire, définir le constructeur d'initialisation pour cette classe, et les accesseurs des propriétés, définir les méthodes ToString et Equals pour cette classe.

b. Définir une classe Service qui contient un objet col de type Dictionary<string, Salarié>. Définir un constructeur par défaut de cette classe. Ainsi que :

1. La méthode **public void Ajouter(string matricule, string nom, int annéeRecrutement, double salaire)** qui permet de créer un objet de type Salarié avec les valeurs passées en argument et de l'ajouter à la collection **col** avec le matricule associé**.**
2. La méthode **public void Vider()** qui permet d'effacer tous les éléments de la collection col.
3. La propriété **public int Nombre** permettant d'obtenir le nombre d'éléments stockés dans la collection col.
4. La méthode **public bool ContientClé(string matricule)** qui détermine si la clé constituée du matricule passé en argument existe dans la collection.
5. L'indexeur **public Salarié this[string matricule]** qui retourne et modifie l'employé identifié par le matricule passé en paramètre.
6. La méthode **public void Supprimer(string matricule)** qui supprime l'élément correspondant à la clé constituée du matricule passé en paramètre.

c. Créer une classe Program permettant d'afficher un menu à l'utilisateur proposant de réaliser les opérations suivantes :

1. Pour ajouter un salarié.
2. Pour rechercher un salarié par son matricule
3. Pour supprimer un salarié par son matricule
4. Pour effacer tous les salariés
5. Pour afficher la liste des salariés et leur matricule
6. Pour quitter le programme