Ecole Nationale des Sciences Appliquées de safi

A.U :2014-2015

Devoir Surveillé N°1 -Semestre2-

'La clarté de vos programmes et sa présentation seront prise en compte dans la notation.'

Exercice1 : Déterminer les erreurs dans ces deux programmes et les corriger

1. public Class A{ int x ;
public A(int x){this.x=x;}
}}
public Class B extends A{
int y ;
public A(int x){
this.x=x;
this.y=y}
}}
	2. On définit une classe abstract D qui hérite d'une classe C :**abstract D extends C**
	.Aprés on fait appel à une méthode **affiche** déjà définie dans cette classe.Dans le main on met cet appel
	D d1=new D( ) ;
	d1.affiche() ;

Exercice2 : 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 caractéres), son prénom(chaine de caractéres) et son âge(double).

+*Ces méthodes :* -Un constructeur à trois paramétres permettant d'initialiser une instance de la classe Employe.

+Affiche() :qui permet d'afficher les trois 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:

-Vendeur : 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()

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

01/04/2016 15:21

-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.Si le tableau est complètement remplie, afficher un message indiquant à l'utilisateur cela.

+void **show( )** : qui affiche tous les employés 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.

+void **Tri (int choix)** :Si choix=1 :trier par ordre alphabétique de nom

Si choix=2 : trier par ordre alphabétique de prenom

Si choix=3 :trier par ordre décroissant de l'âge.

Exercice3 : Le but ici est de créer un programme permettant de gérer les notes des étudiants d'une classe.Pour cela on aura :

-**Classe Note :** contient une suite de notes(Vector<Double>) et une méthode :

+ void ajouter (double x) : pour ajouter une note.

-**Classe Etudiant :** contient le nom et le prénom de l'étudiant. Elle est aussi composée d'une référence sur un objet Note. On veut aussi différencier les notes selon les matières, on va donc créer un objet de type Hashtable< String, Note> qui sera faites d'association (une matière, les notes de l'étudiant dans cette matière).car dans une matière l'étudiant pourra avoir plusieurs notes : Note de examen1,examen2…Cette classe a les méthodes suivantes :

+Etudiant (String nom,String prenom) :le constructeur

double **+void ajouterNoteMatiere(String matiere,int note) :** ajoute une note de l'étudiant concerné dans une matière.

+ double moyenDansMatiere(String matiere) :retourne la moyenne de l'etudiant concerné dans la matière passée en paramètre

+void afficheNote(String matiere) :affiche la note moyenne de l'étudiant concerné dans la matière passée en paramètre

-Classe Groupe : On Créer une classe Groupe composée d'une collection des étudiants (Vector<Etudiant>).Cette classe se compose de :

01/04/2016 15:26

+void ajouterEtudiant(Etudiant e) : pour ajouter un étudiant dans le groupe.

+void afficherEtudiants(String matiere) : affiche les notes des étudiants de groupe dans une matière par ordre alphabétique des noms des étudiants. Une exception de type **NullPointerException** doit être traité au cas où un étudiant n'a eu aucune note dans cette matière.

+void AfficherMoyenClasse(String matiere) : Affiche la moyenne de la classe dans une matiere. Une exception de type **NullPointerException** doit être traité au cas où un étudiant n'a eu aucune note dans cette matière.

+double NoteMax(String matiere) :retourne la note maximale dans une matière.

-**TestGroupe :** Ecrivez un exemple d'une classe main pour appeler la classe groupe.

Annexe :

Les méthodes de la classe Vector :

-void addElement(Object obj) : ajoute un nouvel élément au vecteur.

-void remove (Object obj) : supprime l'élément qui passe en paramètres.

-Object elementAt(int index) : retourne l'élément demandé.

-Enumeration elements() : retourne l'ensemble des éléments du vecteur

-int size() : retourne le nombre d'éléments du vecteur.

Les méthodes de la classe HashTable :

-void put(Object key, Object value) : Insère une clé et une valeur dans la table de hachage.

-Object get (Object key) : Renvoie l'objet qui contient la valeur associée à la clé. Si la clé n'est pas dans la table de hachage, un objet null est renvoyé.

-Enumeration elements() : retourne sous forme d'une énumération tous les éléments de la table,

-Enumeration keys() : retourne cette fois ci toutes les clefs,

-int size() : nombre d 'éléments dans la table

Enumération :

-boolean hasMoreElements(): retourne « true » si l'énumération comporte encore des éléments.

-Object nextElement(): récupère l'élément suivant de l'énumération.

Comparable/CompareTo

-Comparable : interface pour définir un ordre de tri, contient la méthode compareTo (Object o).

-Comparator : interface pour définir un ordre de tri quelconque, contient la méthode compare (Object o1, Object o2)

Collections :

-Collections.Sort(Liste list) :Permet de trier une collection des objets.

-Collections.binarySearch(Liste list, élément) : rend la position de élément dans la collection ou une valeur < 0 sinon. La collection doit être auparavant trié(par ordre croissant), la recherche est une recherche dichotomique.

-Collections.max(Liste list): rend la valeur maximale dans une liste