OFPPT

مكتب التكوين المهني وإنعاش الشغل

Office de la Formation Professionnelle
et de la Promotion du Travail

Examen de Passage

Session Juin 2007

Filière : TSDI

Epreuve : Pratique

Niveau : Technicien Spécialisé

Durée : 4 h 30

Barème : 40 Pts

Variante n° 9

Partie 1 – JAVA (23 Pts)

I - Créez une classe Livre pour représenter un livre qui possède les propriétés suivantes :

- `Titre de type texte`
- `Auteur de type texte`
- `nbPages de type numérique`
- `prix de type numérique`

`Créez la méthode toString, les Modificateurs, les Accesseurs`

II - Créez une classe `Etagere` pour représenter une étagère qui peut contenir un certain nombre de livres (fixe pour chaque étagère).
Vous utiliserez pour cela un tableau.

- Le constructeur prendra en paramètre le nombre de livres que pourra contenir l'étagère.
- Vous ajouterez des méthodes pour (lisez l'énoncé jusqu'au bout avant de commencer à coder)
  - Donner le nombre de livres que peut contenir l'étagère, et le nombre de livres qu'elle contient.
  - Ajouter des livres ("`void ajouter(Livre)`"). Vous ajouterez les livres "à la fin" de l'étagère. Il devra être impossible d'ajouter des livres dans une étagère pleine.

- Récupérer un livre dont on donne la position sur l'étagère (le livre reste sur l'étagère, on récupère simplement une référence sur le livre). La méthode renverra une instance de Livre. La position du premier livre d'une étagère devra être 1 (et pas 0, bien que le livre soit rangé dans la première position du tableau, qui est d'indice 0). La signature de la méthode sera "`Livre getLivre(int)`".
- Chercher sur l'étagère un livre repéré par son titre et son auteur. La méthode renverra la position du livre dans l'étagère (ou 0 si le livre n'y est pas). Le profil de la méthode sera "`int chercher(String, String)`". S'il y a plusieurs livres avec le même titre et le même auteur, la méthode renvoie celui qui a le plus petit indice.
- Avoir une fonctionnalité semblable à la précédente, mais la méthode renvoie un tableau de positions s'il y a plusieurs livres qui ont ce titre et cet auteur. On aimerait appeler cette méthode "chercher" ; est-ce possible ? Le tableau aura pour taille le nombre de livres trouvés (0 si aucun livre n'a été trouvé). Si vous avez besoin de faire une copie de tableau, utilisez la méthode `arraycopy` pour voir... Ecrivez aussi 2 méthodes pour rechercher tous les livres d'un auteur, ou tous les livres qui ont un certain titre. Cette fois-ci, les méthodes renvoient un tableau de livres.
- Enlever des livres. Vous "tasserez" les livres vers le début quand vous enlèverez des livres. Vous écrirez 2 méthodes de même nom pour enlever un livre (on appelle ça une surcharge) :
  - une méthode qui repèrera le livre par sa position (1 pour le premier livre) dans l'étagère (de profil "`Livre enleverLivre(int)`"),
  - une méthode qui repèrera le livre par son auteur et son titre, et qui utilisera la méthode `chercherLivre` (de profil "`Livre enleverLivre(String, String)`").

  Les 2 méthodes renverront le livre supprimé (ou `null` si le livre n'a pas été trouvé).

- S'il y a beaucoup de livres, cet algorithme ("tasser" les livres, et donc faire un grand nombre de décalages) n'est pas bon pour enlever un livre. Réécrivez les méthodes `enleverLivre` pour placer le dernier livre à la place du livre enlevé ; vous nommerez les nouvelles versions "`enlever`".
- Renvoyer une description d'une étagère (la fameuse, et bien utile, méthode `toString()`). Utilisez-la dès le début, par exemple pour tester la méthode `ajouter`.
- Et tout ce qu'il vous semblera utile d'ajouter....

Dans la méthode `main()`, vous créerez des livres, 2 étagères et ajouterez les livres dans les étagères. Vous chercherez un des livres dans une étagère ; s'il y est, vous ferez afficher son nombre de pages. Vous rechercherez tous les livres d'un auteur et les ferez afficher. Vous supprimerez un des livres d'une étagère. Vous ferez afficher à chaque fois que nécessaire les étagères modifiées (en utilisant la méthode `toString()`).

`Méthode main`	**4 pts**
`class Livre`	**6 pts**
`class Etagere`	**8 pts**
`class Librairie`	**5 pts**

Partie 2 – Sql Server (8 Pts)

Soit le modèle logique suivant :

Poste (id_poste, lb_poste)
Entreprise (id_entreprise, lb_entreprise)
Contact (Ncontact, nom, prenom, date_n, id_poste, id_entreprise)

Chaque employé a sa propre gestion des contacts, de cette manière ils sont indépendants

A – Créer la base de données **Contact** en respectant le **MLD** ci-dessus **(2 pts)**

B – Les requête SQL :

1 - Afficher les personnes qui travaillent travaille dans la même entreprise « DIORH ». **(0.5 pt)**

2 – Trier les contacts par Nom **(0.5 pt)**

3 - Les employés veulent envoyer un joyeux anniversaire, afficher les contacts avec date de naissance et l'age de chaque personne. **(1 pt)**

4 - Afficher les noms avec le poste « chef de service » dans les entreprises « DIORH » et « OMNIDATA ». **(1 pt)**

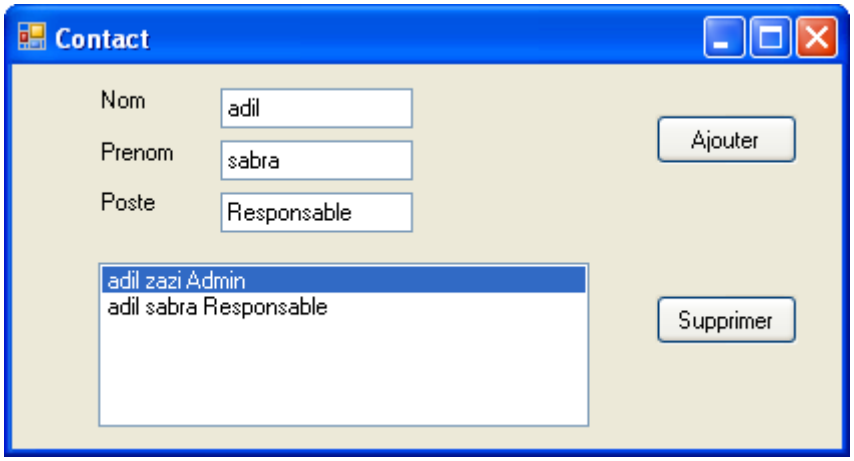
5 - Afficher le nombre des contacts part entreprise. **(1 pt)**

6 – Supprimer les contacts dont le nom commence par S **(1 pt)**

7 – Mettre en majuscule les noms des contacts et en minuscule les prénoms **(1 pt)**

Partie 3 – VB.Net (9 Pts)

Soit à développer l'application ci-dessous



1) Créer la fenêtre. **(1 pt)**
2) Créer la structure sur la quel vous représentez les données. **(2 pts)**
3) Créer la fonction affiche() pour afficher la liste sur la listBox. **(1 pt)**
4) Réaliser le code du bouton Ajouter et Supprimer du listbox. **(2 pts)**
5) Ajouter une option de tri par nom **(1 pt)**
6) Ajouter une fonction qui calcule le nombre des enregistrements **(1 pt)**
7) Sécuriser le lancement de l'application par (Respecter la casse) : **(1 pt)**

Compte : AdminCom

Mot de Passe : CFMOTI