Durée : 1h30

USTHB- Faculté d’Electronique et d’Informatique Module : POO- java Dpt d’Informatique S4 – 2009/ 2010

Section A-ACAD

**Examen du module POO**

**Exercice 1 (14pts)**

**Partie I**

1. Donner l’implémentation d’une classe **Date,** prévoir une méthode **Afficher** qui affiche la date et une méthode **Inferieur** qui vérifie si une date est inférieure à une autre.
2. Donner l’implémentation d’une classe **Heure,** prévoir une méthode **Afficher** qui affiche l’heure.

*On considère une entité de type Message électronique (email) décrite par un numéro (int), un objet (titre du message- String), un expéditeur (****private*** *String), un destinataire (****private*** *String) et une taille (****private*** *float en Ko), une date (****private****) et heure d’envoi et un contenu (texte du message).*

1. Donner l’implémentation de la classe **Message**, spécifier tous les accesseurs (méthodes get et set) pour les attributs déclarés **« private »,** une méthode **Afficher** qui affiche la description d’un message et une autre méthode **AfficherContenu** qui affiche le contenu d’un message.
2. A partir de la classe Message, définir une classe **MessageReçu** qui contient l’attribut **Etat** (**private**) en plus, cet attribut nous renseigne si le *message est lu ou non encore lu (initialement non lu)*. Ajouter les méthodes nécessaires pour la manipulation et L’affichage.

**Partie II**

*On désire implémenter une entité* ***Boite\_Messages*** *décrite par une adresse (String), un nombre de messages reçus (int nbReçu), un nombre de messages non lus (int nbNonLu), un nombre de messages envoyés (int nbEnvoi) et un espace de stockage disponible en Ko (float Espace) ; la boîte doit aussi contenir* ***tous les messages reçus et tous les messages envoyés*** *(on devra utiliser deux structures de stockage de ces messages).*

1. Donner l’implémentation de la classe **Boite\_Messages** en écrivant les méthodes **EnvoyerMessage** (m), **RecevoirMessage** (m), **LireMessage** (num), **AfficherReçus**  et **AfficherEnvoyés** qui affichent respectivement les messages reçus et les messages envoyés de la boite.

*A ce niveau, on voudrait utiliser dans un même programme, les classes précédemment définies.*

1. Ecrire un programme qui :
   * Crée une boite de messages **B** initialement vide avec un espace de stockage = 1024 (Ko).
   * Crée 100 messages reçus (non encore lus) et 20 messages envoyés devant être contenus dans la boite créée (mettre à jour les différents attributs de l’objet **Boite\_Messages** à l’aide des méthodes de la classe).
   * Affiche la description de tous les messages reçus après une date **D** donnée.
   * Affiche le contenu d’un message (non lu) sélectionné à travers son numéro **n**.

**Exercice 2 (6pts)**

1. On considère une classe **Chaine** qui manipule un seul attribut **ch** de type String. Donner l’implémentation de la classe Chaine, contenant les méthodes **Identique** et **Miroir**, vérifiant respectivement si deux chaines sont identiques et si deux chaines sont miroir l’une de l’autre.
2. Ecrire un programme qui crée une structure de type **Vector** qui :

* Stocke un nombre indéfini de chaines de caractères lues au clavier, l’arrêt de la saisie se fait lorsqu’une chaîne saisie existe déjà dans la structure.
* Affiche ensuite, toutes les chaines qui sont miroir les unes des autres dans la structure.

**Indications  pour les deux exercices**

Pour chaque classe, prévoir un **constructeur**.

**charAt(i) :** renvoie le caractère se trouvant à l’indice i d’un objet de type String

**length()** : renvoie la longueur d’une chaîne de caractères

**Java.util.Scanner**: classe pour la lecture de données au clavier

**Java.util.Vector** : classe pour la création d’un vecteur de taille variable

**add(object obj) :** pour ajouter un élément obj à une structure de type Vector

**elementAt (i)** : donne l’élément d’indice i dans l’objet de type Vector

**Size() :** donne le nombre d’éléments d’un objet de type Vector

***Bon Courage***