Durée : 1h30

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

Section A-ACAD

**Rattrapage du module POO**

**Exercice 1**

**Supposition**: on suppose qu’on dispose de la définition suivante pour la classe **Date**

class **Date** { int j, m, a ;

**Date** (int, int, int);

void **SAISIR**()

void **AFFICHER** ();

Date **AJOUTER** (int nbjours) ; // retourne une nouvelle date après ajout d’un nombre de jours

}

Pour chaque classe manipulée (sauf la classe **abonnement**) dans l’exercice, on dispose d’une méthode **void SAISIR()** qui permet de saisir les données au clavier. Il suffit de l’utiliser sans l’écrire.

Dans cet exercice, on s’intéresse à la gestion des abonnements de clients à la connexion internet. Un abonnement se fait en choisissant une catégorie de connexion. Une catégorie de connexion est décrite par un *débit* (256K, 512K, 1M, 2M), une *durée* en nombre de mois, un *type* (Ethernet, USB, WIFI) et un *tarif*.

1. Donner une implémentation de la classe connexion en déclarant ***private***, l’attribut *tarif* et en déclarant les attributs *débit* et *type* comme étant des *types énumérés*. La classe doit contenir une méthode **AFFICHER** et les **accesseurs** (les méthodes get et set) nécessaires.

Un abonné (client) peut être une personne *physique* (un individu) ou une personne *morale* (une société). Une personne physique est décrite par un *numéro de pièce d’identité* (***private*** NPI), un *nom*, un *prénom*, une *date de naissance*, un *numéro de téléphone* et une *adresse* (numéro de rue, nom de rue, code postal et ville). Une personne morale possède une *raison sociale* (société par actions, SARL, …), un *numéro de registre de commerce* (***private*** NRC), un *numéro de tel* et une *adresse*.

1. Donner une implémentation de la classe Adresse. Prévoir une méthode **AFFICHER**.
2. Donner une implémentation des abonnés en utilisant ***l’héritage***. Les classes doivent comporter chacune, une méthode **AFFICHER** et les **accesseurs** nécessaires.

Un abonnement peut être *résidentiel* (si abonné physique) ou *professionnel* (si abonné moral). En plus des informations de l’abonné, pour chaque abonnement on doit enregistrer la *date de début de connexion*, la *date de fin de connexion* (en cas de résiliation), la *date du dernier paiement* (***private***) et aussi la *catégorie de connexion*. Un abonné peut demander le changement de sa connexion.

1. Donner une implémentation des abonnements en utilisant ***l’héritage***. Les classes doivent comporter une méthode **AFFICHER**, une méthode **SAISIR,** une méthode **RESILIE** qui vérifie si l’abonnement est résiliéet les accesseurs nécessaires.
2. Ecrire un programme qui crée une collection d’abonnements professionnels à l’aide d’informations entrées au clavier (méthode SAISIR) et affiche tous les abonnements en retard de paiement dans une ville v donnée.
3. On voudrait déclarer la classe abonnement comme étant abstraite (**abstract**), proposer une solution pour son implémentation.

**Exercice 2**

On considère l’entité **Intervalle** dans l’ensemble des nombres réels. Un intervalle est défini par deux valeurs **BI** et **BS** correspondant respectivement, à la borne inférieure et la borne supérieure de l’intervalle.

Donner l’implémentation en java, de la classe Intervalle comportant les méthodes :

**AFFICHER** : affiche l’objet intervalle [BI, BS]

**MILIEU** : calcule le point milieu de l’intervalle

**DISJOINT** : vérifie si deux intervalles sont disjoints (aucune intersection entre eux)

**CONTENU**: vérifie si un intervalle est complètement inclus dans un autre

**INTERSECTION** : calcule l’intervalle intersection de deux autres intervalles

**AVANT** : vérifie si un intervalle est avant un autre sur l’ensemble des réels

**VERIF\_FUSION** : vérifie s’il est possible de fusionner deux intervalles en un seul

**FUSION :** calcule l’intervalle fusion de deux autres intervalles.

**Indications  pour les deux exercices**

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

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

**Java.util.ArrayList** : classe pour la création d’une liste d’objets de taille variable

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

**get(i)** : donne l’élément d’indice i dans la liste de type ArrayList

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

***Bon Courage***