UIC GI 2

M.K. GUENNOUN

**TP JAVA et Systèmes distribués Interfaces et Héritage**

On souhaite développer une application de gestion de comptes bancaires comprenant les entités suivantes :

* Une entité personne qui est définie par les éléments suivants
  + Nom (chaine de caractères)
  + Prenom (chaine de caractères)
  + Adresse (chaine de caractères)
  + Pour implanter cela, nous définirons
    - L’interface personneInt offrant les méthodes
      * void Afficher()
      * void setNom(String n)
      * void setPrenom(String p)
      * void setAdresse(String Add)
    - la classe personne implémentant l’interface personneInt
* Une entité Employe qui reprend les éléments définis pour l’entité personne en plus des éléments :
  + Fonction (chaine de caractères)
  + Salaire (réel)
  + Pour implanter cela, nous définirons
    - L’interface EmployeInt offrant les méthodes de l’interface personneInt en plus des méthodes :
      * void setFonction(String F)
      * void setSalaire(double s)
    - la classe Employe héritant de personne et implémentant l’interface EmployeInt.
* Une entité Client qui reprend les éléments définis pour l’entité personne en plus des éléments
  + CB (objet de la classe CompteBancaire définie ci-dessous)
  + Pour implanter cela, nous définirons
    - L’interface clientInt offrant les méthodes de l’interface personneInt en plus des méthodes :
      * void setCompte(CompteBancaire C)
* Une entité EmployeClient correspondant à un employé qui est en même temps un client. Proposer une classe correspondante
* Une entité CompteBancaire qui est définie par les éléments suivants :
  + NumeroCompte (entier)
  + Solde (double)
  + Pour implanter cela, nous définirons
    - La classe CompteBancaire reprenant les attributs nécessaires et définissant les méthodes:
      * void afficher()
      * void setNumeroCompte(int n)
      * void setSolde(double s)
      * float getSolde()
      * void depot(double montant)
      * boolean retrait(double montant)
      * boolean virement(CompteBancaire destinataire, double montant)
* Une entité CompteBancaireRemunere qui reprend les éléments des attributs de la classe CompteBancaire en plus de
  + tauxRemuneration (double)
  + Implanter cette dernière classe en héritant de CompteBancaire et en introduisant les méthodes
    - Void setTaux(double t)
    - void AppliquerRemuneration()

Une fois que toutes les classes et interfaces définies, implanter une classe GestionBancaire qui définira la méthode main avec le comportement suivant :

* déclaration de
  + Cinq tableaux d’une taille maximale de 100 éléments où chacun stockera un ensemble de
    - Clients
    - Employés
    - Clients-employés
    - Comptes Bancaires
    - Comptes Bancaires rémunérés
  + Cinq variables entières comptabilisant les nombre d’entités dans chaque tableau
* Une boucle comprenant des switch imbriqués
  + Le switch de plus haut niveau permettra de choisir l’entité sur laquelle en souhaite agir
  + Les switch internes permettant de
    - traiter les différentes méthodes relatives à chaque entité
    - transformer un employé ou un client en client-employé et vice-versa
    - transformer un compte bancaire en compte bancaire rémunéré et vice versa
    - afficher l’ensemble des entités du système