TD de Programmation par objets et Java #3

J. Ferber - Octobre 1999

Les associations entre objets

Exercice 1: variation sur le compte en banque

On donne les classes `Compte` et `Personne` suivantes :

```
class Compte {
   int numero;
   int solde;

   Personne client;

   void deposer(int s){
      solde = solde + s;
   }
   void retirer (int s){
      solde = solde - s;
   }

   int avoirSolde() {
      return(solde);
   }

   void associerPersonne(Personne p){
      client = p;
      p.ajouterCompte(this);
   }

}

class Personne {
   String nom;
   String prenom;

   Compte monCompte;

   void ajouterCompte(Compte c){
      monCompte = c;
   }
   Personne(String m, String p, int a){
      nom = m;
      prenom = p;
      annee = a;
   }
}
```

Définir un constructeur de la forme:

```
Compte(int n, String m, String p, int a)
```

où n représente le numéro du compte, m le nom du client, p son prénom et a son année de naissance.

Ce constructeur doit créer une personne et mettre à jour les liens entre le client du compte et le compte lui-même.

Exercice 2: L'arbre généalogique

On désire réaliser un système permettant de construire et d'afficher un arbre généalogique.

Question a

a) Créer la classe Date qui contient une date sous la forme de 3 entiers, le jour, le mois et l'année.

b) Définir un constructeur de la forme Date(int j, int m, int a) qui sert à construire une telle date.

c) Définir une méthode `versChaine()` qui retourne une chaîne de caractère, représentation de la date sous la forme *jour/mois/année*.

Exemple:

```
Date d = new Date(20, 10, 1999);
System.out.println(d.versChaine());    //    affiche: 20/10/1999
```

Question B

a) Définir la classe Personne qui est telle que toute personne est définie par son nom, son prénom, sa date de naissance et éventuellement par son père, sa mère (qui sont tous deux des personnes) et par sa date de décès.

b) Faire un constructeur qui permet de créer une personne à partir de son nom, son prénom, le jour, le mois et l'année de sa date de naissance.

c) Faire des méthodes d'accès pour la définition du pere, de la mere et de l'année de décès de la personne. Voilà ce que cela donne:

```
Personne jean = new Personne("Dupond", "Jean", 3, 1, 1966);
Personne marie = new Personne("Durand", "Marie", 5, 9, 1942);
Personne marcel = new Personne("Dupond", "Marcel", 15, 10, 1938);
Personne helene = new Personne("Leblanc", "Hélène", 21, 11, 1908);
Personne paul = new Personne("Durand", "Paul", 12, 06, 1905);

helene.deces(10,08,1986);
paul.deces(1,12,1970);

jean.pere(marcel);   // Jean a comme pere Marcel
jean.mere(Marie);
marie.pere(Paul);
marie.mere(Helene);
```

d) Faire une méthode afficher() dans Personne qui donne le résultat suivant:

```
jean.afficher()
→
    Jean Dupond
    né(e) le 3/1/1996

helene.afficher()
→
    Hélène Leblanc
    né(e) le 21/11/1908
    décédé(e) le 10/08/1986
```

e) Faire une méthode afficherGenealogie() dans Personne affiche toute la généalogie d'une personne. Exemple:

```
jean.afficherGenealogie();
→
```

```
Jean Dupond
né(e) le 3/1/1996
a comme père:
   Marcel Dupond
   né(e) le 15/10/1938
a comme mère:
   Marie Durand
   né(e) le 5/9/1942
   a comme père:
       Paul Durand
       né(e) le 5/9/1942
       décédé(e) le 1/12/1970
   a comme mère
       Hélène Leblanc
       né(e) le 21/11/1908
       décédé(e) le 10/08/1986
```

Note: dans un premier temps on ne s'intéressera pas à l'indentation. Mais dans un deuxième temps on se posera la question de l'indentation. Idée: créer une méthode auxiliaire (que l'on appellera afficherGenealogie1(int i) où i est le nombre de blanc qui sont affichées avant les message de sortie, et créer de même une méthode auxiliaire afficher1(int i) qui fait comme afficher mais qui place i blanc avant le message de sortie.