TD 1: Classes et objets

Exercice 1. Trace d'un programme

Que fournit le programme suivant ?

```
class Entier
{
        public Entier (int nn) {n = nn;}
        public void incr (int dn) {n += dn;}
        public void imprime () {System.out.println (n);}
        private int n ;
}

public class Ex1
{
        public static void main (String args[])
        {
                Entier n1 = new Entier (2);
                System.out.print ("n1 = "); n1.imprime();
                Entier n2 = new Entier (5);
                System.out.print ("n2 = "); n2.imprime() ;
                n1.incr(3) ;
                System.out.print ("n1 = "); n1.imprime() ;
                System.out.println ("n1 == n2 est " + (n1 == n2));
                n1 = n2; n2.incr(12);
                System.out.print ("n2 = "); n2.imprime() ;
                System.out.print ("n1 = "); n1.imprime() ;
                System.out.println ("n1 == n2 est " + (n1 == n2)) ;
        }
}
```

Exercice 2. Trace d'un programme

a. Détailler les différentes valeurs présentes dans les champs de la classe A lors de l'exécution du programme suivant. Quel est le résultat de l'exécution ?

```
class A
{
        public A (int coeff)
        {
                nbre *= coeff;
                nbre += decal;
        }
        public void affiche ()
        {
                System.out.println ("nbre = " + nbre + " decal = " + decal);
        }
        private int nbre = 20 ;
        private int decal;
}

public class Ex2
{
        public static void main (String args[])
        {
                A a = new A (5);
                a.affiche();
        }
}
```

b. Comment faut-il modifier l'initialisation par des champs de la classe A pour que la même exécution du programme affiche *nbre = 103...* ?

Exercice 3. Géométrie plane

Définir des classes représentant

1. les points de $\mathbb{R}^2$ (on utilisera le type flottant `double` pour représenter une approximation d'un nombre réel). Cette classe comprendra aussi une méthode calculant la distance à un autre point.
2. les droites du plan (représentées par un couple de points). Elle disposera en outre des méthodes suivantes :
   — test d'appartenance
   — test de parallélisme
   — test d'orthogonalité
   — une fonction retournant la droite parallèle à la droite courante passant par un point donné
3. les triangles du plan. La classe disposera en outre des méthodes suivantes :
   — test si le triangle est isocèle
   — test si le triangle est rectangle
   — calcul de chacune des trois hauteurs, et médianes
   — calcul du centre de gravité

Exercice 4. Conception de classes

On souhaite écrire un logiciel de gestion du trafic ferroviaire. Il faut que ce logiciel représente

Les voies, définies par
— deux gares (marquant chaque extrémité)
— une longueur en km

Les gares, définies par
— un nom (chaîne de caractère)
— un ensemble de voies qui lui sont connectées.

On pourra utiliser les collections ArrayList<Voie> ou HashSet<Voie> pour stocker l'ensemble de voies d'une gare.

a. Écrire ces deux classes Voie et Gare

On souhaite maintenant représenter des lignes, c'est à dire un chemin d'une gare à une autre, passant éventuellement par plusieurs gares intermédiaires en suivant des voies.

b. Écrire une telle classe.

c. Écrire une classe statique Outils contenant une méthode distance. Cette méthode devra pouvoir prendre en argument
— une voie
— une ligne
— deux gares
et retourner la (meilleure) distance correspondante.

Exercice 5. Compter les instances d'une classe

Écrivez une classe Point possédant une méthode nbObjets qui renvoie le nombre d'instances de Point créées.

Exercice 6. L'Unique

Écrivez une classe AnneauUnique qui n'autorise la création que d'une seule instance.