TD n°2

Rappels Java et introduction à la POO (suite)

Exercice 1 *Devinettes*

On considère la classe suivante :

```
public class Integer{
    public static int n;
    private int m;

    public Integer(int i){
      n = i;
      this.m = i;
    }

    public void afficher(){
      System.out.println("n = " + n + ", m = " + this.m);
    }
}
```

1. Ecrire les accesseurs/modifieurs pour `m`. Quelles sont les différences entre `n` et `m` ?
2. On considère la classe suivante :
   ```
   public class TD2{
       public static void main(String[] args){
         Integer un = new Integer(1);
         un.afficher();
         Integer deux = new Integer(2);
         deux.afficher();
         un.afficher();
       }
   }
   ```
   Qu'obtient-on dans le terminal lorsqu'on exécute `TD2` ?
3. Dans la méthode `main` précédente, on souhaite modifier la valeur de la propriété de classe `n`. Comment procéder ?
4. On rajoute maintenant les lignes suivantes :
   ```
   int x = 1;
   int y = x;
   y = 3;
   System.out.println(x);
   Integer trois = un; trois.setM(3);
   un.afficher();
   ```

Qu'obtient-on dans le terminal ?

Exercice 2 *Manipulations de matrices*

1. Ecrire une classe `Matrice` comportant deux champs de type `private` : sa `taille` et un `tableau` de `double` à deux dimensions.
2. Ecrire un constructeur ne prenant qu'un seul argument, un tableau de `double` et qui recopie ce dernier. Est-ce une bonne idée d'écrire simplement `m.tableau = argument` ?
3. Ecrire l'accesseur à la propriété `taille`.
4. Ecrire les accesseurs/modifieurs `getCoeff, setCoeff` servant à lire et modifier les coefficients de la matrice.
5. Ecrire une méthode `multiplier`, prenant un argument `double` et qui permet de multiplier la matrice par ce coefficient.
6. Ecrire une méthode `multiplier`, prenant en argument une matrice et permettant de multiplier la matrice par celle passée en argument.
7. Ecrire une méthode `multiplier` prenant deux arguments, un entier (la ligne) et un `double` (le coefficient) qui multiplie la ligne correspondante de la matrice par le coefficient passé en second argument.

   Surcharge : *Une méthode est dite surchargée lorsqu'on désigne par le même nom plusieures méthodes différentes.* java *fait la différence entre ces différentes méthodes grâce au type des arguments.*
8. Ecrire une méthode `permuter` prenant deux arguments entiers et qui permute la première ligne avec la seconde.
9. Ecrire une méthode `additionner` prenant deux arguments entiers et qui ajoute la première ligne à la seconde.
10. Ecrire une méthode de classe `comparer` prenant deux arguments de type `Matrice` et retournant un booléen. Elle retournera `true` si les deux matrices ont les mêmes coefficients (et la même dimension !). Est-on obligé de faire deux boucles imbriquées ?