TP 2

Exercice 1

On va reprendre les classes *Point*, *Segment* et *Triangle* codées précédemment. Si les champs représentant l'abcisse et l'ordonnée d'un *Point* n'étaient pas de type flotant, faire le nécessaire dans votre code pour que ce soit le cas.

Dans la classe Segment, ajouter les méthodes :

- **milieu** : Retourne le milieu du segment courant.

```
public Point milieu ();
```

- **longueur** : Retourne la longueur du segment courant.

```
public float longueur ();
```

- **projection** : Retourne le *Point* résultat de la projection du *Point* passé en argument sur le *Segment* courant.

```
public Point projection (Point p);
```

Dans la classe *Triangle*, ajouter les méthodes :

- **perimetre** : Retourne le périmètre du triangle courant.

```
public float perimetre ();
```

- **aire** : Retourne l'aire du triangle courant.

```
public float aire ();
```

Exercice 2

Notre but est de travailler avec les matrices de nombres entiers. Nous représenterons le contenu de ces matrices avec des variables de type *int*[][].

Une matrice peut donc être vue comme un objet qui a comme paramètres un nombre de lignes, un nombre de colonnes et un tableau à deux dimensions.

Compléter la classe *Matrice* suivante :

```
public class Matrice{
  private int nbLignes;
  private int nbCol;
  private int [][] coefficients;

  public Matrice (){}
  public Matrice (int nbL, int nbC){
    //A vous de jouer
    //Ce constructeur cree une matrice contenant nbL lignes et nbC colonnes et que des 0.
  }
  public Matrice (int [][] coef){
    //A vous de jouer
    //Cette matrice sera initialisee avec le tableau coef.
  }
  public int getNbLignes () {//A vous de jouer}
  public int getNbCol () {//A vous de jouer}

  public void setNbLignes (int nbL) {//A vous de jouer}
  public void setNbCol (int nbC) {//A vous de jouer}
}
```

Exercice 3

Ajouter à la classe Matrice les méthodes suivantes :

- **litMatrice** : demande à l'utilisateur les dimensions d'une matrice ainsi que ses coéfficients, crée la matrice correspondante et la renvoie.
- **afficheMatrice** : affiche une matrice ligne par ligne.

- **addition** : retourne la matrice résultat de l'addition de la matrice courante avec la matrice passée en argument.

```
public Martrice addition (Matrice m);
```

- **produit** : retourne la matrice résultat du produit de la matrice courante avec la matrice passée en argument.

```
public Martrice produit (Matrice m);
```

- **transpose** : retourne la transposée de la matrice courante.

```
public Martrice transpose ();
```