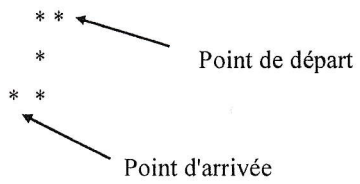
TP3

I. Introduction

Ce TP s'inspire d'un petit langage de programmation appelé "Logo" qui utilise un ensemble de commandes pour effectuer des dessins en déplaçant un pointeur appelé "tortue". Les commandes sont des ordres donnés à la tortue. Initialement la tortue se situe à une position définie (coordonnées x, y) ; elle est orientée par défaut vers le haut (nord). Elle peut avancer d'une unité (commande 'A') ou tourner à gauche sur elle-même (commande 'G') ou enfin tourner à droite sur elle-même (commande 'D').

Par exemple : la séquence de commandes `{'G','A','G','A','A','D','A'}` engendre :



II. La classe Tortue

Une tortue va se déplacer selon un cycle de commandes. Par exemple, le cycle {'G', 'A', 'D', 'A'} engendre un déplacement de la tortue avec une séquence de mouvements correspondant à :

'G', 'A', 'D', 'A', 'G', 'A', 'D', 'A', ..., 'G', 'A', 'D', 'A' ...

On peut définir une classe **Tortue** à travers les attributs suivants :

- char[] cycle; // cycle de déplacement de la tortue
- int _x, _y; // sa position courante
- int indice; // indice courant dans le cycle
- int dir; // direction courante

La classe **Tortue** doit offrir une méthode **mouvement()** qui à chaque appel engendre un mouvement de la tortue en respectant le cycle de déplacement (cf. indice).

Exemple d'utilisation ; création d'une **Tortue** et déplacement :

```
char[] c1 = {'G','A','G','A','D','D'}; // définition d'un cycle de déplacement
Tortue t1=new Tortue(c1,10,10); // constructeur d'une Tortue avec Cycle et Coordonnées de départ
t1.mouvement(); t1.mouvement(); t1.mouvement(); t1.mouvement();
t1.mouvement(); t1.mouvement(); t1.mouvement(); t1.mouvement();
// 8 mouvements ont été commandés
```

la **Tortue** t1 s'est déplacée de la façon suivante (affichage du résultat) :

```
G : 10 10
A : 9 10
```

```
G : 9 10
A : 9 11
D : 9 11
D : 9 11
G : 9 11
A : 8 11
```

Exo 1 Compléter la définition de la **classe Tortue** à partir du squelette contenu dans le fichier : **Tortue.java**

Les différentes méthodes proposées sont données à titre indicatif pour faciliter la conception de la classe **Tortue**. Elles peuvent être modifiées. D'autres méthodes peuvent être ajoutées.

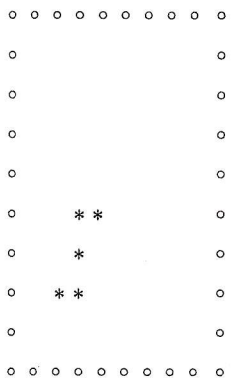
```
public class Tortue {

 private char[] cycle;  // cycle de déplacement de la tortue
 private int _x,_y;     // sa position courante
 private int indice;    // indice courant dans le cycle
 private int dir;       // direction courante  : codage interne suggéré : nord(0); ouest(1) ; sud(2) ; est(3);

 // constructeur d'une Tortue avec Cycle et Coordonnées de départ
 public Tortue(char[] c, int x, int y) {
        // à compléter
 }
 // codage interne suggéré : nord(0); ouest(1) ; sud(2) ; est(3);
 // déplacement dans une direction donnée (d)
 private void deplace(int d) {
        // à compléter
 }
 // exécution d'un (1 seul) mouvement selon le cycle
 // rend vrai si le déplacement est effectif (avance)
 public boolean mouvement() {
        // à compléter
 }
 // test unitaire
 // on crée une tortue et on la déplace
 public static void main (String args[]) {
        // à compléter
 }
}
```

Exo 2 Tester votre classe **Tortue** en reproduisant l'exemple donné précédemment avec affichage à l'écran des commandes de déplacement associées aux coordonnées de la tortue.

III. La classe Grille

Nous allons maintenant définir une grille de déplacement pour les tortues afin d'afficher leur trace.

Pour ce faire, on attribue à chaque tortue une marque particulière (un caractère). La grille est définie par une matrice de caractères. Cette matrice est remplie au départ par des caractères blanc (' '), sauf pour les bords où l'on met un caractère prédéfini ('o' par exemple). La grille sera alors mise à jour au fur et à mesure des déplacements des tortues en mémorisant dans la grille leur marque respective.



Exemple d'affichage d'une grille (10,10)
où s'est déplacé une tortue (marque '*').

La classe grille doit notamment offrir les deux méthodes suivantes :

```
// affichage de la grille
// c-a-d, de la matrice de caractères (qui devra contenir les traces des tortues)
public void affiche() {
  // .... à compléter ...
}
```

```
// déplacement sur la grille d'une tortue
// jusqu'a   collision de la tortue contre un bord
//           ou nombre de mouvements maximum (nb) atteint
public void deplace (int nb, Tortue t) {
// .... à compléter ...
}
```

Exo 3 Modifier la classe **Tortue** pour permettre d'attribuer une marque (un caractère) à chaque Tortue créée. Compléter la définition de la **classe Grille** à partir du squelette contenu dans le fichier : **Grille.java**

Les différentes méthodes proposées sont données à titre indicatif pour faciliter la conception de la classe **Grille**. Elles peuvent être modifiées. D'autres méthodes peuvent être ajoutées.

```
public class Grille {

 // matrice de déplacement des tortues
 private char [][] G;

 // construction de la grille de déplacement
 // délimitée par une bordure
 public Grille(int x, int y) {
        // ... à compléter ...
 }

 // affichage de la grille
 // c-a-d, de la matrice de caractères (qui devra contenir les traces des tortues)
 public void affiche() {
        // ... à compléter ...
 }
```

```
// déplacement sur la grille d'une tortue
 // jusqu'à    collision de la tortue contre un bord
 //          ou nombre de mouvements maximum (nb) atteint

 public void deplace (int nb, Tortue t) {
          // ... à compléter ...
 }

public static void main (String args[]) { ... }
}
```

Exo 4 Tester votre classe **Grille** en définissant une Grille **g1** et une Tortue **t1**. Afficher les déplacements de la tortue dans la Grille.

On pourra afficher l'état de la grille après chaque mouvement de la **Tortue** afin de vérifier et de valider précisément les classes mises en œuvre. Vérifier la gestion des collisions sur les bords.

```
char[] c1 = {'G','A','G','A','A','D','A'};      // définition d'une tortue
Tortue t1=new Tortue(c1,5,5,'*');

Grille g1= new Grille(10,10);                   // définition d'une grille de déplacement
g1.deplace(7, t1);                              // 7 mouvements maxi sur la tortue
g1.affiche();
```

Exo 5 Enrichir la classe **Grille** pour permettre la gestion de plusieurs **Tortues** différentes (cycles et positions de départ différents) se déplaçant en même temps sur la même grille. Les tortues à déplacer sur une même grille seront stockées dans un tableau. Pour ce faire, créer une nouvelle méthode **deplace**(int nb, Tortue[] tabTortue) et la tester :

```
// déplacement sur la grille des tortues contenues dans un tableau
// jusqu'à    collision de l'une des tortues contre un bord
//          ou nombre de mouvements maximum (nb) atteint

public void deplace (int nb, Tortue[] tabTortue) {
          // ... à compléter ...
}
```