

TP 2 Optionnel

## Mini-turtle, fractales



### 1 – Classe Turtle

#### Implémentation

Cette partie consiste à coder un « mini-turtle » en Java, c'est-à-dire une classe implémentant une partie des fonctionnalités du module Python « turtle », qui consiste à dessiner avec une tortue en lui donnant des ordres de déplacement. La tortue ne dessine que si son stylo est baissé. A tout instant, elle possède une orientation, selon laquelle elle se déplace quand on lui demande d'avancer ou de reculer. Plus d'informations [ici](#).

Implémentez une classe Turtle héritant de Paint et possédant les méthodes suivantes :

Méthode	Description
<code>void penUp()</code>	Lève le stylo
<code>void penDown()</code>	Baisse le stylo
<code>void forward(int distance)</code>	Avance la tortue de la distance spécifiée en pixels, en suivant l'orientation courante. Dessine un trait si le stylo est baissé.
<code>void backward(int distance)</code>	Reculé la tortue de la distance spécifiée en pixels, en suivant l'orientation courante. Dessine un trait si le stylo est baissé.
<code>void moveTo(int x, int y)</code>	Déplace la tortue au point spécifié, et dessine un trait si le stylo est baissé. Ne change pas l'orientation de la tortue.
<code>void right(int degrees)</code>	Tourne la tortue du nombre de degrés spécifiés vers la droite
<code>void left(int degrees)</code>	Tourne la tortue du nombre de degrés spécifiés vers la gauche
<code>int getHeading()</code>	Renvoie l'orientation courante de la tortue, en degrés
<code>Point getPosition()</code>	Renvoie la position courante de la tortue (coordonnées (x,y) en pixels)
<code>bool isDown()</code>	Renvoie true si le stylo est baissé

#### Indications :

- On supposera que la tortue est dans l'état initial suivant : stylo baissé, orientation = 0°, position = (0,0).
- Quel(s) paramètre(s) possède(nt) le(s) constructeur(s) de Turtle ?
- Quel(s) attribut(s) devez-vous ajouter à la classe Turtle ?
- Quelle(s) interface(s) implémente la classe Turtle ?

Pour plus de confort, et par cohérence avec le module Python turtle, on peut :

- Placer l'origine du repère au centre de la fenêtre (plutôt que dans le coin inférieur gauche, de base en Swing)
- Diriger l'axe y vers le haut (plutôt que vers le bas, de base en Swing)

#### Test

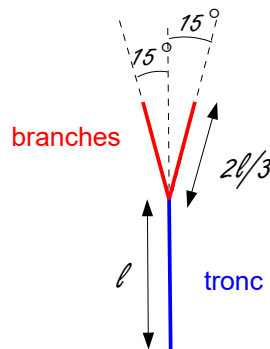
Vérifiez le bon fonctionnement de votre classe en dessinant :

- Un carré
- Un triangle équilatéral

## 2 – Dessin récursif d'arbres

### Présentation

L'objectif de cette partie est de dessiner récursivement une fractale dont l'ordre 1 est représentée ci-dessous :


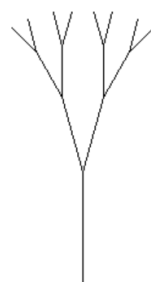
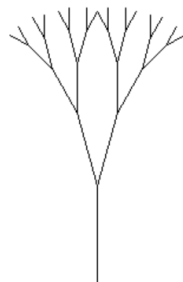


Il est à noter que :

- L'angle formé par les branches par rapport au tronc est  $\pm 15^\circ$
- La longueur des branches est égale à  $2/3$  la longueur du tronc
- Les couleurs sont là pour rendre le dessin plus clair, mais tout sera dessiné dans la même couleur.

L'ordre  $n+1$  consiste à répéter ce motif sur les branches de l'ordre  $n$ .

A titre d'illustration, les figures ci-après montrent les ordres 2, 3 et 4:

Ordre 2	Ordre 3	Ordre 4
		

### Implémentation de base

Dans une classe `Tree`, implémentez une méthode récursive :

```
public static void drawTree(Turtle turtle, int l, int n)
```

qui dessine la fractale décrite ci-dessus à l'ordre  $n$  en utilisant la tortue `turtle`, en partant d'un tronc de longueur `l`.

Ajoutez une méthode `main()` à la classe `Tree` pour tester `drawTree()`.

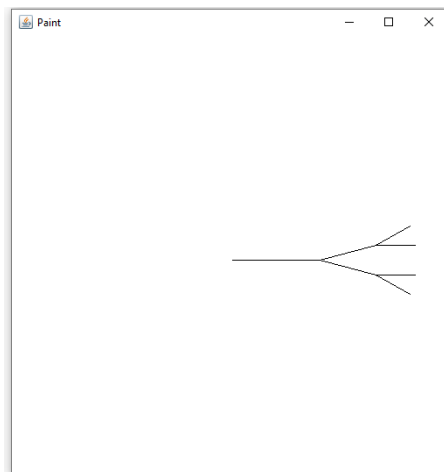
## Exemples d'utilisation

### Exemple minimal (ordre 2)

Le code suivant :

```
public static void main(String[] args){  
    Turtle turtle = new Turtle(500, 500);  
    drawTree(turtle, 100, 2);  
  
}
```

doit aboutir à l'affichage :



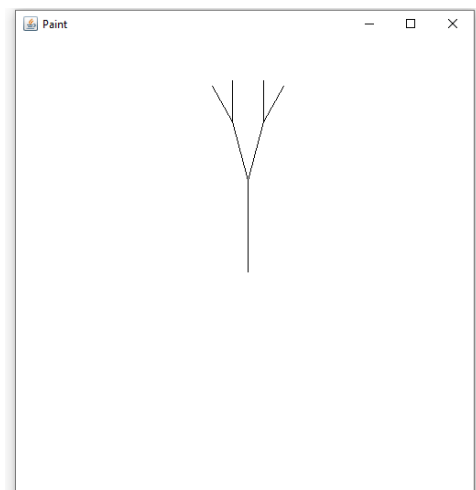
A noter :

- L'état final de la tortue peut être différent de ce qui est représenté ici, il dépend de votre implémentation.
- L'arbre doit être dessiné selon l'orientation initiale de la tortue (qui pointe vers l'est par défaut). Il est donc normal que l'arbre soit horizontal dans cet exemple. Ne pas changer le code de `drawTree()` pour le rendre vertical.

### Arbre vertical

Dans le code fourni ci-dessous, on ajoute l'appel `turtle.left(90)` *avant* l'appel à `drawTree()` pour dessiner un arbre vertical.

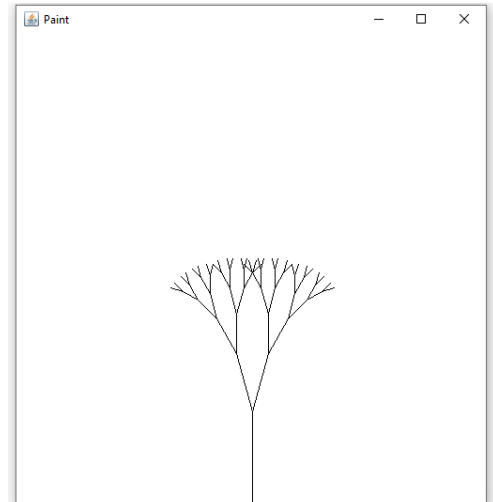
```
public static void main(String[] args){  
    Turtle turtle = new Turtle(500, 500);  
    turtle.left(90);  
    drawTree(turtle, 100, 2);  
  
}
```



### Test final (ordre 5)

Le code suivant contient des lignes supplémentaires qui permettent de faire démarrer la tortue en bas de la fenêtre, et donc d'exploiter toute sa hauteur. On peut ainsi plus aisément tester des ordres plus élevés, qui requièrent plus d'espace. Exemple à l'ordre 5 :

```
public static void main(String[] args){
    Turtle turtle = new Turtle(500, 500);
    turtle.penUp();
    turtle.goto(0,-250);
    turtle.penDown();
    turtle.left(90);
    drawTree(turtle, 100, 5);
}
```



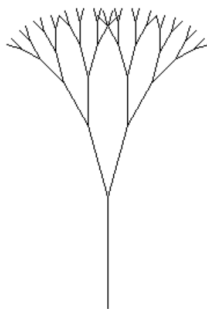
### Personnalisation

Ajoutez des paramètres suivants à `drawTree()` afin que l'on puisse personnaliser :

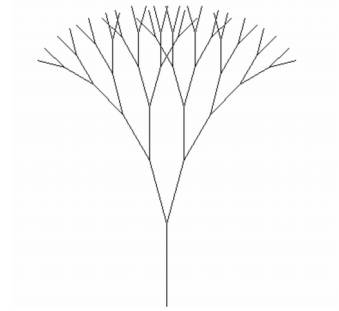
- le ratio de longueur `lratio` entre les branches et le tronc (2/3 par défaut)
- l'orientation relative `langle` de la branche gauche par rapport au tronc (15° par défaut)
- l'orientation relative `rangle` de la branche droite par rapport au tronc (15° par défaut)

**Contrainte** : le test final donné dans la partie « exemples d'utilisation » (sans aucun paramètre de personnalisation) devra toujours fonctionner et donner le rendu demandé.

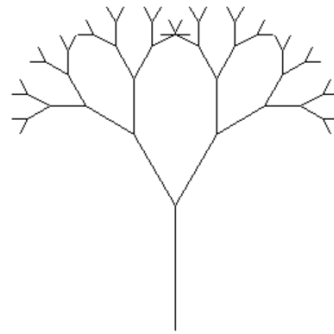
Exemples à l'ordre 5:

Aucune personnalisation	
-------------------------	---

lratio = 4/5



langle = rangle = 30°



langle = -5°  
rangle = 30°

