Aix*Marseille université
Initiative d'excellence
Année 2018-19

Domaine Sciences et Technologies
Licence informatique

Programmation 2 : TP 2
Code UE : SIN3U02

Classes et instances

1 Objets et dessins

L'objectif de cet exercice est de vous permettre de vous familiariser avec les classes et les objets Java. Vous allez devoir définir des classes et les utiliser de façon à écrire un code le plus lisible et évolutif possible. Il est très important de bien choisir les noms des variables, méthodes et propriétés, d'introduire des méthodes privées pour rendre votre code le plus lisible possible et de pensez à définir correctement la visibilité des différentes méthodes et propriétés des classes (mots-clés `public` et `private`).

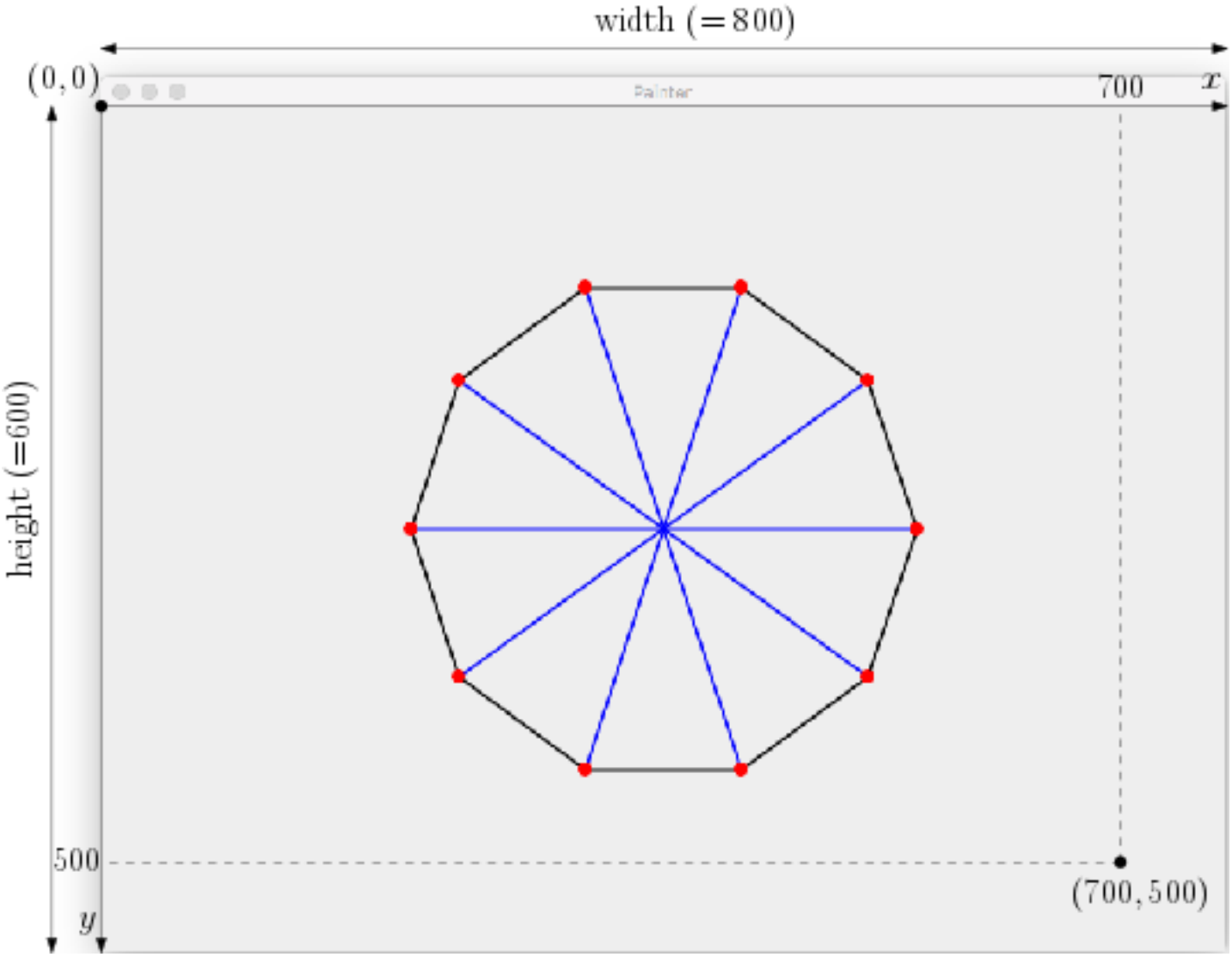
1.1 Géométrie

1.1.1 Ajout d'un fichier `.jar`

Dans ce TP, vous allez devoir dessiner des lignes et des points à l'écran. Pour ce faire, vous allez utiliser la classe `Painter` du paquet `tp2.lib` contenu dans le fichier tp2.jar. Cette classe possède :

- un constructeur qui prend respectivement la largeur et la longueur en pixels de la zone de dessin.
- une méthode `void addPoint(double x, double y, Color color)` qui ajoute au dessin un point (x,y) de la couleur demandée.
- une méthode `void addLine(double x1, double y1, double x2, double y2, Color color)` qui ajoute au dessin une ligne entre le point $(x1, y1)$ et $(x2, y2)$ de la couleur donnée en argument.

L'origine du système de coordonnées se trouve dans le coin supérieur gauche de la fenêtre et l'axe des y est inversé : les points ayant une coordonnée positive en y se trouve en dessous de l'axe x.



Créer un projet `drawing` sous IntelliJ IDEA comme vous l'avez fait au TP précédent pour la classe `Student`. Ajoutez la bibliothèque `tp2.jar` à votre projet. Pour se faire sous IntelliJ IDEA, il faut aller au menu `File`, puis `Project Structure`. Il faut ensuite sélectionner `Libraries`, cliquer sur le `+` puis java et enfin sélectionner le fichier tp2.jar que vous avez téléchargé au préalable.

Mettez le code ci-dessous dans la classe `Main` de votre projet pour tester le bon fonctionnement du paquet `tp2.lib` :

```
import tp2.lib.Painter;
import java.awt.Color;

public class Main {
    public static void main(String[] args){
        Painter painter = new Painter(400, 400);
        painter.addLine(100, 100, 300, 100, Color.black);
        painter.addLine(300, 100, 300, 300, Color.black);
        painter.addLine(300, 300, 100, 300, Color.black);
        painter.addLine(100, 300, 100, 100, Color.black);
        painter.addPoint(100, 100, Color.red);
        painter.addPoint(300, 100, Color.red);
```

```
        painter.addPoint(100, 300, Color.red);
        painter.addPoint(300, 300, Color.red);
    }
}
```

Les deux premières lignes du code servent à importer les classes `Painter` et `Color`.

1.1.2 Classe `Point`

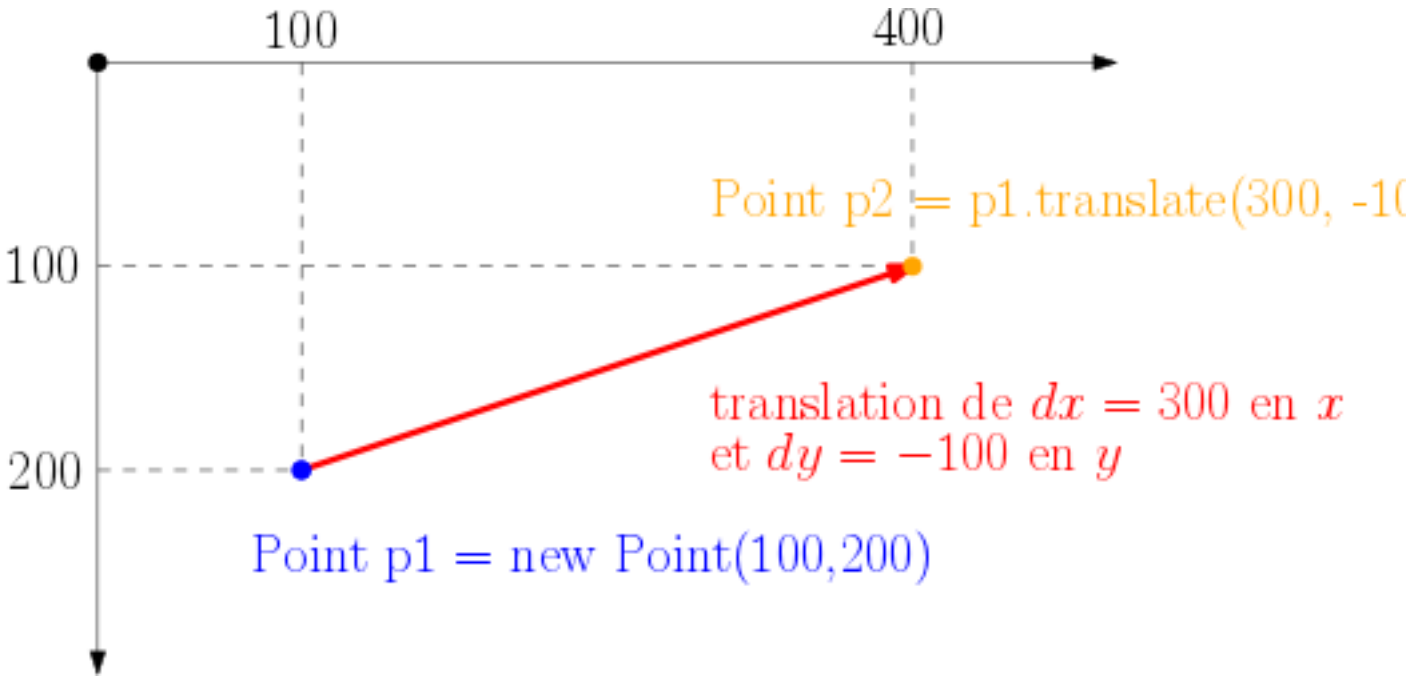
1. Écrivez une classe Point possédant :
   — deux attributs `x` et `y` de type double ;
   — un constructeur `Point(double x, double y)` ;
   — une méthode `void draw(Painter painter, Color color)` qui dessine un point `(this.x,this.y)` de couleur `color` dans le `painter` ;
   — une méthode `void drawLine(Point p, Painter painter, Color color)` qui dessine une ligne de couleur `color` entre le point `(this.x, this.y)` et le point `(p.x, p.y)` dans le `painter` ;

   Le code suivant doit avoir le même comportement que celui de la question 1 :

```
import tp2.lib.Painter;
import java.awt.Color;

public class Main {
    public static void main(String[] args){
        Painter painter = new Painter(400, 400);
        Point p1 = new Point(100,100);
        Point p2 = new Point(300,100);
        Point p3 = new Point(300,300);
        Point p4 = new Point(100,300);
        p1.drawLine(p2, painter, Color.black);
        p2.drawLine(p3, painter, Color.black);
        p3.drawLine(p4, painter, Color.black);
        p4.drawLine(p1, painter, Color.black);
        p1.draw(painter, Color.red);
        p2.draw(painter, Color.red);
        p3.draw(painter, Color.red);
        p4.draw(painter, Color.red);
    }
}
```

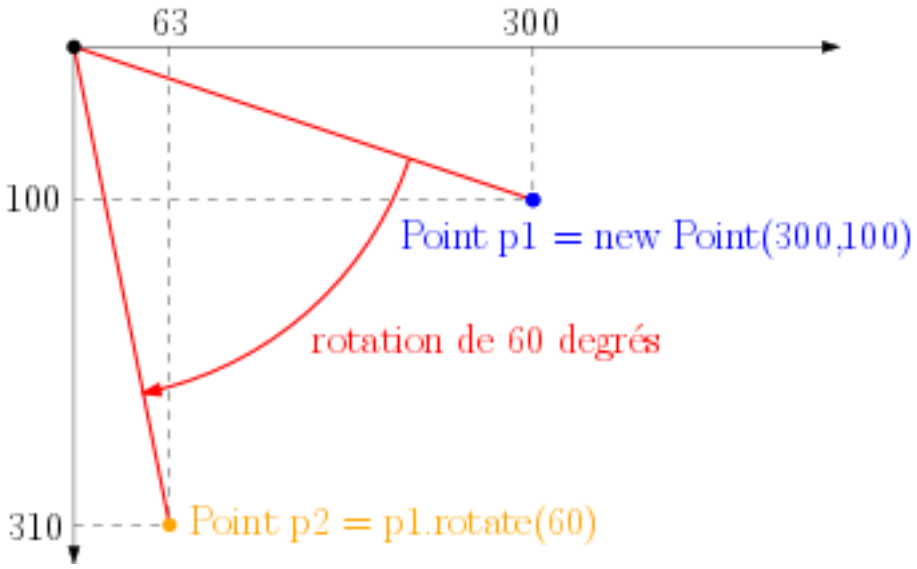
2. Réécrivez le code ci-dessus en utilisant des boucles pour éviter la répétition de code.
3. Ajoutez la méthode suivante à la classe `Point` :
   — `Point translate(double dx, double dy)` qui renvoie un nouveau point issu de la translation du point de `dx` sur l'axe x et de `dy` sur l'axe y (voir image ci-dessous).



Le code suivant doit avoir le même comportement que celui de la question 1 :

```
Painter painter = new Painter(400, 400);
Point p1 = new Point(100,100);
Point p2 = p1.translate(200, 0);
Point p3 = p2.translate(0, 200);
Point p4 = p3.translate(-200,0);
p1.drawLine(p2, painter, Color.black);
p2.drawLine(p3, painter, Color.black);
p3.drawLine(p4, painter, Color.black);
p4.drawLine(p1, painter, Color.black);
p1.draw(painter, Color.red);
p2.draw(painter, Color.red);
p3.draw(painter, Color.red);
p4.draw(painter, Color.red);
```

4. Réécrivez le code ci-dessus en utilisant des boucles pour éviter la répétition de code.
5. Ajoutez une méthode `rotate(double angle)` qui crée un nouveau point en effectuant une rotation d'angle `angle` (exprimé en degrés) dans le sens des aiguilles d'une montre et de centre $(0, 0)$.



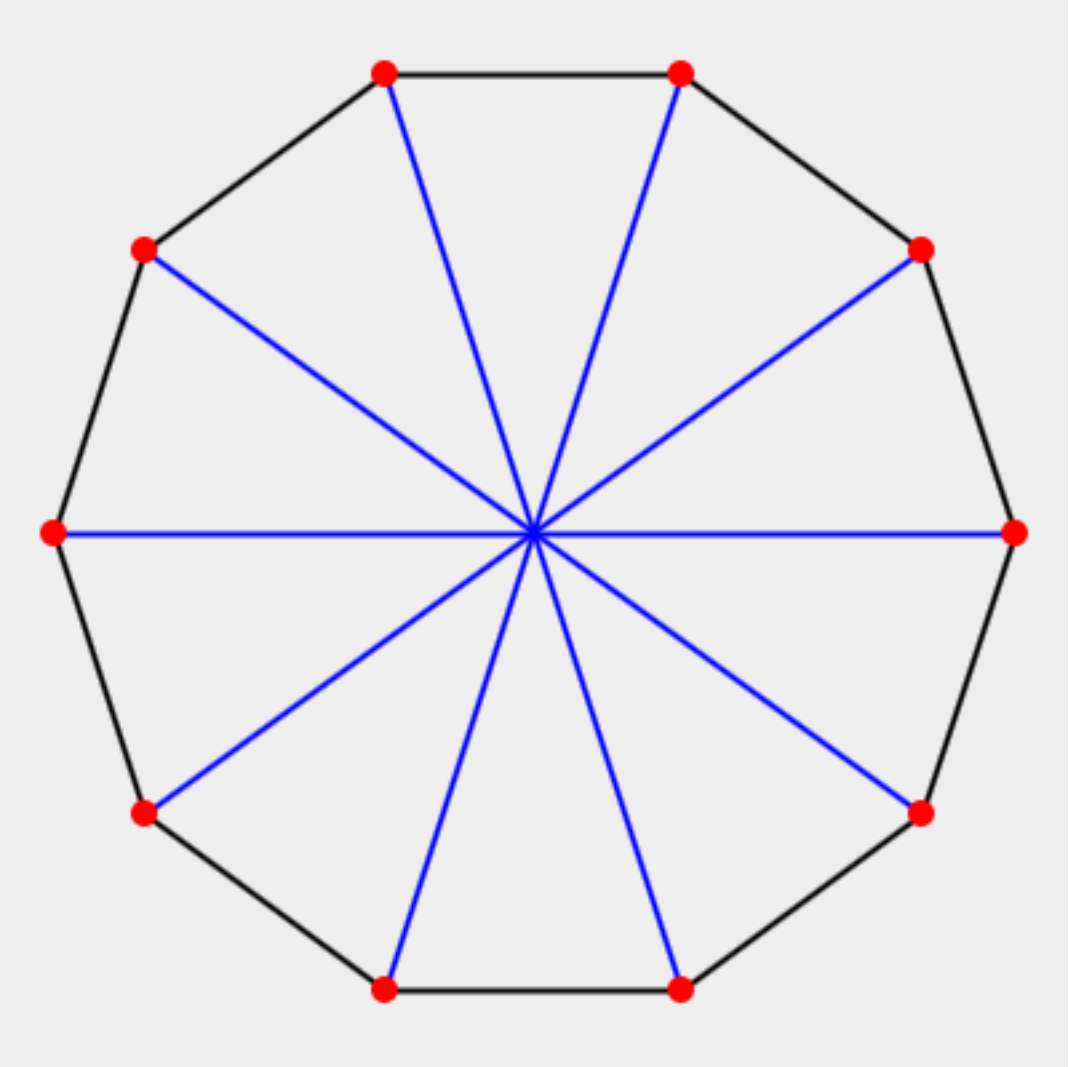
Vous devez utiliser les méthodes et propriétés statiques suivantes de la classe `Math` :

— `Math.PI` : valeur de π ;
— `Math.cos(x)` le cosinus de `x` (avec `x` exprimé en radians) ;
— `Math.sin(x)` le sinus de `x` (avec `x` exprimé en radians).

Rappel : pour effectuer une rotation, vous pouvez utiliser les formules suivantes où $\alpha = (\pi \times angle)/180$ pou obtenir les coordonnées (x', y') d'un point obtenu par rotation du point de coordonnées (x, y) :

— $x' = x\cos(\alpha) - y\sin(\alpha)$
— $y' = x\sin(\alpha) + y\cos(\alpha)$

6. Dessinez la roue suivante :



1.1.3 Tortue

Vous allez écrire une classe `Turtle` permettant de simuler les déplacement d'une tortue. Une tortue est un robot qui se situe au centre de la fenêtre au début du programme et qui regarde vers le haut. Elle possède un crayon qui lui permet de dessiner une ligne lorsqu'elle avance. Il est de possible de lui donner les ordres suivants :

- **[forward 30]** fait avancer la tortue de 30 pas.
- **[turnLeft 20]** fait tourner la tortue de 20 degrés sur sa gauche.
- **[turnRight 20]** fait tourner la tortue de 20 degrés sur sa droite.
- **[penUp]** fait en sorte que la tortue lève le crayon.
- **[penDown]** fait en sorte que la tortue descende le crayon.
- **[setColor red]** fait en sorte que la tortue prenne un crayon rouge.

La suite suivante d'ordres permet de dessiner un carré rouge de 40 pas de côté :

```
[setColor red]
[penDown]
[forward 40]
[turnLeft 90]
[forward 40]
```

```
[turnLeft 90]
[forward 40]
[turnLeft 90]
[forward 40]
```

La suite d'ordres précédente doit pouvoir être simulée par la classe `Turtle` en écrivant le code suivant :

```java
public class Main {
  public static void main(String[] args) {
    Turtle turtle = new Turtle();
    turtle.setColor(Color.red);
    turtle.setPenDown();
    turtle.moveForward(40);
    turtle.turnLeft(90);
    turtle.moveForward(40);
    turtle.turnLeft(90);
    turtle.moveForward(40);
    turtle.turnLeft(90);
    turtle.moveForward(40);
  }
}
```

La classe tortue contient :

- les attributs suivants :
  - `penColor` de type `Color` qui correspond à la couleur du crayon de la tortue.
  - `angleDirection` de type `double` qui correspond à l'angle entre l'axe x et la direction courante de la tortue (compté dans le sens des aiguilles d'une montre).
  - `position` de type `Point` qui correspond à la position courante de la tortue.
  - `penIsDown` de type `boolean` qui est à vrai si le stylo est posé (la tortue dessine un trait correspondant à sa trajectoire lorsqu'elle avance) et faux sinon (la tortue ne dessine rien lorsqu'elle avance).
  - `painter` : l'objet de la classe `Painter` dans laquelle la tortue dessine.
- Un constructeur :
  - `public Turtle(int width, int height)` : crée une tortue en initialisant les attributs de la manière suivante :
    - `penColor` : initialisé à `null`
    - `angleDirection` : initialisé de sorte à ce que la tortue pointe vers le Nord.
    - `painter` : initialisé en créant un nouveau `Painter` dont la taille (`width` et `height`) correspond aux attributs du constructeur.
    - `penIsDown` : initialisé à `false`.
    - `position` : initialisé en créant un nouveau `Point` au centre de la fenêtre.
- des méthodes :
  - `public void moveForward(double distance)` : attend 500 millisecondes (en utilisant `sleep` expliqué ci-dessous) et fait avancer la tortue dans sa direction courante de la `distance` donnée (traçant un trait si le crayon est posé).
  - `public void setColor(Color color)` : change la couleur du crayon par `color`.
  - `public void turnLeft(double angle)` : change la direction courante de la tortue en la faisant tourner de `angle` degrés dans le sens **inverse** des aiguilles d'une montre.
  - `public void turnRight(double angle)` : change la direction courante de la tortue en la faisant tourner de `angle` degrés dans le sens des aiguilles d'une montre.
  - `public void setPenDown()` : pose le stylo (met `penIsDown` à `true`)
  - `public void setPenUp()` : lève le stylo (met `penIsDown` à `false`)

Pour dessiner à l'écran, votre tortue doit déléguer le dessin des lignes à la classe `Point` de la première partie. Notez que vous pouvez également faire dormir le programme pendant `n` millisecondes en utilisant la fonction statique `sleep(n)` de la classe `Tools` du paquet `tp2.lib`.

Une fois votre classe écrite, testez-là avec le code suivant :

```
public class Main {

  public static void drawSquare(Turtle turtle, int size) {
    for (int i = 0; i < 4; i++) {
      turtle.forward(size);
      turtle.turnLeft(90);
    }
  }

  public static void main(String[] args) {
    Turtle turtle = new Turtle(800,600);
    turtle.setColor(Color.black);
    turtle.setPenDown();
    int n = 20;
    for (int i = 0; i < n; i++) {
      turtle.turnRight(360.0/n);
      drawSquare(turtle, 100);
    }
  }
}
```

1.1.4 Fractales

1. Ajouter à la classe `Turtle` une méthode `void drawString(String sequence, double length, double angle)` permettant de faire exécuter à la tortue la séquence d'ordres codée dans une chaîne de caractères `sequence`. La chaîne de caractères doit être lue de la gauche vers la droite et chacun des caractères doit être interprété de la façon suivante :
   - le caractère **'A'** fait avancer la tortue de `length` pas ;
   - le caractère **'+'** fait tourner la tortue de angle degrés à droite ;
   - le caractère **'-'** fait tourner la tortue de angle degrés à gauche ;
   - les autres caractères doivent être ignorés.

   Pour tester votre méthode, dessinez un carré avec la séquence `"A+A+A+A"`, une longueur de `100` et un angle de 90°.
2. Nous allons appliquer plusieurs fois un ensemble de règles de réécriture sur une séquence de caractères afin d'obtenir une séquence d'ordres qui dessine une fractale à l'écran. Une règle de réécriture est un couple $(c \to S)$. Écrivez une classe `Rule` modélisant une règle de réécriture. Cette classe possède :
   - deux attributs privés `symbol` (de type `char`) et `sequence` (de type `String`) ;
   - un constructeur qui permet d'initialiser les deux attributs ;
   - deux accesseurs `char getSymbol()` et `String getSequence()` permettant d'obtenir les valeurs des deux attributs.
3. Écrivez la classe `SetOfRules` modélisant un ensemble de règles. Cette classe possède :
   - un attribut privé `rules` de type `Rule[]` ;
   - un constructeur permettant d'initialiser l'attribut `rules` ;
   - une méthode publique `String apply(String sequence)` qui applique les règles du tableau `rules` à la chaîne de caractères sequence et qui retourne le résultat. Appliquer un ensemble de règles

$\{(c_1 \to S_1), \ldots, (c_n \to S_n)\}$ sur la séquence `sequence` consiste à remplacer chaque caractère c_i de la chaîne `sequence` par la chaîne `S_i`. Par exemple, l'application des règles $\{('A' \to' AB'), ('B' \to' BA')\}$ sur `"AB"` doit produire `"ABBA"`.

4. Testez votre programme en exécutant la fonction suivante :

```
 private static void drawFractale(){
   int width = 590, height = 580, nbIterations = 4;
   Turtle turtle = new Turtle(width, height);

   // Déplacement de la tortue en bas à gauche.
   turtle.setPenUp();
   turtle.turnLeft(90); turtle.moveForward(width/2-10);
   turtle.turnLeft(90); turtle.moveForward(height/2-10);
   turtle.turnLeft(180);
   turtle.setPenDown();

   // Définition des règles
   Rule[] rules = { new Rule('X', "XAYAX+A+YAXAY-A-XAYAX"),
           new Rule('Y', "YAXAY-A-XAYAX+A+YAXAY") };
   SetOfRules setOfRules = new SetOfRules(rules);

   // Application des règles nbIterations fois
   String sequence = "X";
   for (int i = 0; i < nbIterations; i++)
       sequence = setOfRules.apply(sequence);

   // Dessin de la séquence par la tortue
   turtle.drawString(sequence, 7, 90);
}
```