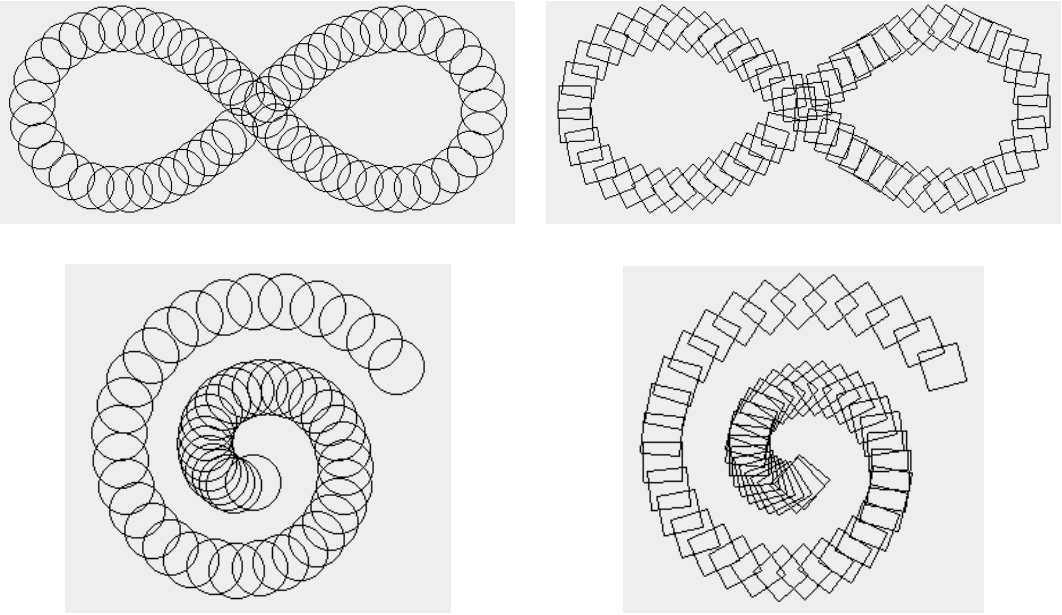
Projet

1 Contexte

On veut dessiner des figures géométriques en rotation sur elle-mêmes se déplaçant sur des chemins dans le plan cartésien. Par exemple, on voit ci-dessous les traces laissées par un cercle et un carré le long d'une lemniscate de Bernoulli et d'une spirale d'Archimède.



L'objectif du projet est de programmer un ensemble de classes et interfaces en Java pour réaliser ces déplacements dans une fenêtre graphique. On attend une solution extensible permettant d'ajouter aisément de nouvelles formes et de nouveaux chemins sans modifier le code existant.

2 Travail

Le travail sera décomposé en étapes comme suit.

1. Définir les classes et interfaces principales : attributs, méthodes et relations entre ces types.
2. Implémenter une classe de forme et une classe de chemin, par exemple pour déplacer le cercle sur la lemniscate. Faire des tests pour vérifier que la forme se déplace bien le long du chemin.
3. Implémenter une interface graphique permettant (1) de voir une forme qui se déplace le long d'un chemin et (2) de produire une trace (comme les dessins ci-dessus). On veut des mouvements lisses et non saccadés ainsi qu'une vitesse de déplacement régulière.
4. Ajouter de nouvelles formes et de nouveaux chemins, par exemple le carré et la spirale. La solution est extensible s'il n'est pas nécessaire de modifier les classes existantes.
5. Gérer des interactions.
   — Un clic souris dans une forme change sa couleur.
   — Les touches + et − du clavier permettent d'augmenter ou de diminuer la vitesse de déplacement.
   — Les formes et chemins peuvent être choisis via l'interface, par exemple via des boutons radio.

Si le temps le permet : ajouter un bouton dans l'interface permettant d'inverser le déplacement (la forme refait le chemin parcouru dans le sens inverse) ; gérer une vitesse de rotation de la forme sur elle-même et une vitesse de déplacement le long du chemin au moyen de *sliders* ; générer une documentation au format Javadoc (voir la classe Question fournie au TP n°5).

3 Évaluation

Le projet sera réalisé impérativement en **binômes au sein d'un même groupe de TP**. Il pourra y avoir au plus un trinôme en cas de nombre impair. Les binômes et trinômes devront être déclarés au plus tard en novembre auprès de votre chargé de TP.

Les évaluations auront lieu en deux temps comme décrit ci-dessous. Il faudra à chaque fois déposer un fichier dans l'espace dédié à son groupe de TP sur madoc dans la section Devoirs. Dans chaque binôme ou trinôme, un seul membre doit effectuer le dépôt.

2 décembre 2020

Un rapport de conception après les étapes 1 et 2 décrites ci-avant dans la section « Travail » sera rendu au plus tard le 2 décembre 2020. Ce rapport au format **pdf** décrira les classes et interfaces principales. Par exemple, une classe Derived héritant de Base sera décrite comme suit :

Classe Derived

- Rôle : Cette classe représente *bla bla bla* et sert à *bla bla bla*
- Super-classe : Base
- Attributs

int val	*bla bla bla*
AClass obj	*bla bla bla*

- Méthodes

Derived(int val)	constructeur tel que *bla bla bla*
void action(int n)	réalise l'action de *bla bla bla* n fois

Un diagramme de classes pourra également être fourni.

20 décembre 2020

Un fichier archive contenant les sources Java (seulement les fichiers **.java**) et un rapport final au format **pdf** sera rendu au plus tard le 20 décembre 2020. Le rapport final sera constitué uniquement des réponses aux questions suivantes. Les réponses devront être claires, concises et argumentées.

1. Copier une image de l'interface graphique et décrire les fonctionnalités de l'application.
2. Encapsulation. L'encapsulation est-elle respectée ?
3. Héritage. Justifier chaque relation d'héritage entre classes et chaque relation entre une interface et une classe d'implémentation.
4. Polymorphisme. Comment le polymorphisme est-il mis en œuvre ? Pourquoi est-ce utile ?
5. Extensibilité. Votre solution est-elle extensible ?
6. Interface. Comment fonctionne l'interface graphique, selon MVC ou un autre modèle ?
7. Interactions. Quels types d'interactions sont gérées ? Comment ?
8. Expérimentations. Copier les traces produites dans l'interface graphique pour montrer l'étendue des expérimentations.
9. Regard critique. Quels sont les limites de votre programme ? Quels sont les points positifs ou négatifs ? Qu'avez-vous appris ?