# **Mini Projet Dessin Vectoriel**

# Introduction

Le projet sur le dessin vectoriel a pour objectif de définir un langage de dessin vectoriel à partir du langage de programmation Java. Pour ce faire, on se donne trois phases essentielles :

* Définition du langage
* Définition de l’interprétation du langage
* Modularité de ces définitions

# Le langage

Le langage a été défini en prenant en compte cet aspect modulaire du projet. Nous devions avoir dans notre projet des classes représentant un ensemble de « formes » géométriques prédéfinis que l’utilisateur pourra utiliser en paramétrant des arguments. Par exemple, pour une forme géométrique simple comme une droite ou un rectangle, l’utilisateur utilisera notre implémentation en fournissant respectivement deux points pour la droite et quatre points pour le rectangle.

Ces formes pourraient néanmoins être regroupées dans une entité abstraite et définissant globalement certains aspects et propriétés qu’elles ont en commun. Cette représentation pourrait être très efficace à long terme quand il s’agira de faire évoluer le langage et à court terme pour éviter de la duplication de code. D’où l’aspect modulaire de ce projet encore justifié.

# Modularité

Un patron de conception était alors nécessaire. Devant la multitude de classes faisant état de « formes géométriques », une structure adaptée était nécessaire. Structure d’autant plus justifiée que l’écriture de scripts par l’utilisateur et l’interprétation du langage devraient être affranchies le plus possible du langage Java.

On a donc jugé opportun d’utiliser un patron composite pour l’architecture de notre projet. Une première raison pourrait être que la similitude de plusieurs de nos classes représentant des formes au sens des méthodes implémentées, du paramétrage et de l’utilisation pourrait faciliter l’interprétation du langage. Plus concrètement, on aurait comme dans toute structure composite :

* Un **composant** qui sera ici notre objet permettant de faire l’abstraction pour tous les composants. Ce serait donc une classe nommée « Chemin » dans notre projet qui fixe l’utilisation et certaines propriétés (comme *dessiner*) pour toutes les formes. Techniquement, on pourrait le représenter comme une interface ou encore comme une classe abstraite.
* La **feuille** qui ici n’a pas trop d’intérêt puisque notre hiérarchie n’est pas très riche. Mais on pourrait s’imaginer, dans l’enrichissement du langage, des formes géométriques étant des sous-éléments d’autres formes géométriques. Par exemple, un arc de cercle comme sous élément d’un cercle
* Le **composite** qui ici sera représenté par toutes nos classes étendant « Chemin ». Les comportements spécifiques d’une forme géométrique particulière seront définis dans ces classes.

# Interprétation