# Projet Java — Développement de l'application *Shape Studio*

## Présentation générale

Ce projet a pour objectif la conception et le développement d'une application logicielle intitulée **Shape Studio**. Celle-ci offrira à l'utilisateur la possibilité de créer, manipuler et visualiser différentes figures géométriques (cercles, rectangles, triangles, polygones réguliers, etc.) au moyen d'une interface graphique conviviale. Les étudiant·e·s mobiliseront, dans ce cadre, les principes fondamentaux de la programmation orientée objet, la gestion des exceptions, ainsi que l'usage de bibliothèques graphiques en Java.

### Objectifs pédagogiques

Ce projet vise à :

- Mettre en œuvre les concepts fondamentaux de la programmation orientée objet (encapsulation, héritage, polymorphisme).
- Exploiter des structures de données pour organiser et gérer des collections de formes.
- Développer une interface graphique utilisateur (GUI) en Java.
- Appliquer les bonnes pratiques en matière de développement logiciel (organisation du code, gestion des exceptions, tests unitaires, etc.).

## 1 Structure du projet

L'architecture du projet est organisée comme suit :

- `src/main/java/fr/gema/` : contient le code source principal de l'application.
- `src/main/java/fr/gema/shapes/` : regroupe les classes représentant les différentes figures géométriques. Toutes les classes sont définies dans le package `fr.gema.shapes`.
- `fr.gema.shapes.exceptions` : sous-package contenant les classes d'exceptions personnalisées. **Ces exceptions sont déjà implémentées.**
- `ShapeStudioGUI.java` : classe principale de l'interface graphique.
- `src/test/java/fr/gema/shapes/` : contient les tests unitaires permettant de valider les différentes implémentations. **Ces tests sont fournis.**
- `pom.xml` : fichier de configuration du projet pour Maven.

Les fichiers `.gitignore` et `README.md`, présents à la racine du projet, relèvent respectivement de la gestion de version et de la documentation. Ils ne nécessitent aucune intervention de votre part.

### Compilation et exécution

L'utilisation de Maven est recommandée pour compiler et exécuter le projet.
Pour **compiler le projet**, exécutez la commande suivante dans le terminal à la racine du projet :

```
mvn compile
```

Pour **exécuter l'application**, utilisez la commande suivante :

```
mvn exec:java "-Dexec.mainClass=fr.gema.ShapeStudioGUI"
```

Pour **nettoyer le projet** (supprimer les fichiers compilés et les dépendances), vous pouvez utiliser la commande suivante :

```
mvn clean
```

## 2 Éléments fournis

### Interface `Shape`

L'interface `Shape` constitue le contrat que toutes les formes géométriques doivent respecter. Elle définit les opérations fondamentales (calculs d'aire et de périmètre, translation, rotation, redimensionnement, accès au centre, etc.) devant être implémentées par chaque classe concrète.
**Cette interface est déjà fournie dans le projet.**

### Classe abstraite `GenericShape`

Cette classe abstraite offre une implémentation partielle de l'interface `Shape`, en factorisant les attributs et comportements communs à toutes les formes (étiquette, couleurs, épaisseur du trait, etc.). Les classes représentant des formes concrètes hériteront de `GenericShape` et fourniront l'implémentation des méthodes spécifiques à leur géométrie.
**Cette classe est déjà fournie dans le projet.**

## 3 Travail attendu

Le projet se décline en deux volets :

1. **Implémentation des classes de formes géométriques** — Obligatoire. Ce travail doit impérativement être achevé avant d'entamer la seconde partie.
2. **Extension de l'application via l'interface graphique** — Facultative. Cette partie s'adresse aux étudiant·e·s ayant complété la première phase avant la fin du cours.

*Remarque importante : l'utilisation d'outils d'intelligence artificielle pour générer du code est interdite pour la première partie. Elle est en revanche autorisée pour la seconde.*
**L'évaluation sera centrée sur l'implication en classe.** Le travail personnel en dehors des séances ne sera valorisé que pour les étudiant·e·s ayant activement participé aux activités en classe.

# 4 Modalités de rendu

Deux modalités de dépôt sont possibles :

— Dépôt d'une archive `.zip` contenant l'intégralité du projet sur Classroom.

— Dépôt sur un répertoire GitHub public ; le lien vers ce dépôt devra être soumis via Classroom.

**La date limite de dépôt est fixée au 22 juin 2025, 23h59.**

# 5 Première partie : implémentation des formes géométriques

Les classes représentant les figures géométriques sont fournies sous forme de squelettes à compléter. Chaque classe doit respecter le contrat de l'interface `Shape` et hériter de `GenericShape`.
Pour chaque classe :

— Implémentez les méthodes indiquées par le mot-clé `TODO`.

— Prenez connaissance des commentaires et indications dans le code source.

— Exécutez les tests unitaires correspondants à l'aide de Maven (voir exemple ci-dessous).

```
mvn -Dtest=fr.gema.shapes.PointTest test
```

En cas d'échec d'un test, analysez le message d'erreur et corrigez votre implémentation.

## Classe `Point`

Cette classe modélise un point du plan en deux dimensions.
Voici des formules utiles pour la classe `Point` :

— **Distance** entre deux points $(x_1, y_1)$ et $(x_2, y_2)$ :

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

où $(x_1, y_1)$ et $(x_2, y_2)$ sont les coordonnées des deux points.

— **Translation** : Pour translater un point $(x, y)$ de $(dx, dy)$, on utilise :

$$\begin{cases} x' = x + dx \\ y' = y + dy \end{cases}$$

— **Rotation** : Pour faire tourner un point $(x, y)$ autour d'un centre de rotation $(x_0, y_0)$ d'un angle $\theta$ (en radians), on utilise :

$$\begin{cases} x' = (x - x_0)\cos(\theta) - (y - y_0)\sin(\theta) \\ y' = (x - x_0)\sin(\theta) + (y - y_0)\cos(\theta) \end{cases}$$

## Classe `Rectangle`

Modélise un rectangle par ses deux coins opposés. Soient $w$ la largeur et $h$ la hauteur du rectangle, et $(x_c, y_c)$ les coordonnées du centre du rectangle.

- **Aire** :
$$A = wh$$
- **Périmètre** :
$$P = 2(w + h)$$
- **Translation** : Pour translater un rectangle, on translatera ses deux points (supérieur gauche et inférieur droit) de la même manière que pour un point.
- **Rotation** : Pour faire tourner un rectangle autour de son centre, on peut utiliser la rotation des points qui le définissent.
- **Rédimensionnement d'un rectangle** : Pour redimensionner un rectangle, on peut ajuster les coordonnées de ses deux points (supérieur gauche et inférieur droit) en fonction d'un facteur d'échelle donné. Le rédimensionnement ne doit pas changer la position du centre du rectangle. Soit $k$ le facteur d'échelle et $(x, y)$ un point du rectangle : les nouvelles coordonnées $(x', y')$ après redimensionnement seront données par :
$$\begin{cases} x' = x_c + k \cdot (x - x_c) \\ y' = y_c + k \cdot (y - y_c) \end{cases}$$

## Classe `Circle`

Représente un cercle à partir de son centre et de son rayon $r$.

- **Aire** :
$$A = \pi r^2$$
où $r$ est le rayon du cercle.
- **Périmètre** (circonférence) :
$$C = 2\pi r$$
- **Translation** : Pour translater un cercle, on translatera son centre de la même manière que pour un point.
- **Rotation** : Un cercle est invariant par rotation autour de son centre, donc la rotation n'affecte pas sa forme.
- **Redimensionnement** : Pour redimensionner un cercle, on peut ajuster son rayon. Si on redimensionne par un facteur $k$, le nouveau rayon sera :
$$r' = k \cdot r$$

## Classe `Triangle`

Défini par trois sommets. Soient $a$, $b$, et $c$ les longueurs des côtés du triangle, et $s$ le demi-périmètre.

— **Aire** (formule de Héron) :

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

Pour le calcul des longueurs des côtés, vous pouvez utiliser la classe `Segment` qui est déjà implémentée dans le projet.

— **Périmètre** :

$$P = a + b + c$$

— **Translation** : Pour translater un triangle, on translatera chacun de ses trois points de la même manière que pour un point.

— **Rotation** : Pour faire tourner un triangle autour de son centre, on peut utiliser la rotation des points qui le définissent. Le centre du triangle peut être approximé par le barycentre de ses sommets. Le barycentre $G$ de trois points $(x_1, y_1)$, $(x_2, y_2)$, et $(x_3, y_3)$ est donné par :

$$G = \left( \frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3} \right)$$

— **Redimensionnement** : Pour redimensionner un triangle, on peut ajuster les coordonnées de ses trois points en fonction d'un facteur d'échelle donné. Le rédimensionnement ne doit pas changer la position du barycentre du triangle, donc si on redimensionne par un facteur $k$, les nouvelles coordonnées de chacun des points seront calculées comme suit :

$$\begin{cases} x_i' = x_G + k \cdot (x_i - x_G) \\ y_i' = y_G + k \cdot (y_i - y_G) \end{cases}$$

où $(x_G, y_G)$ sont les coordonnées du barycentre du triangle, et $(x_i, y_i)$ sont les coordonnées du sommet $i$ du triangle.

## Classe `RegularPolygon`

Défini par un centre $(x_c, y_c)$, un rayon et un nombre de côtés.
Voici des formules utiles pour la classe `RegularPolygon` :

— **Calcul des sommets d'un polygone régulier** : Les sommets d'un polygone régulier à $n$ côtés peuvent être calculés en utilisant les formules suivantes pour chaque sommet $i$ (de 0 à $n-1$) :

$$x_i = x_c + r \cdot \cos\left( \frac{2\pi i}{n} + \theta \right)$$

$$y_i = y_c + r \cdot \sin\left( \frac{2\pi i}{n} + \theta \right)$$

où $(x_c, y_c)$ est le centre du polygone, $r$ est le rayon, et $\theta$ est l'angle de rotation du polygone. Pour un polygone régulier à $n$ côtés, les sommets sont répartis uniformément autour du cercle de rayon $r$ centré en $(x_c, y_c)$. Par exemple, pour un triangle équilatéral ($n = 3$), les sommets seront à 120 degrés d'intervalle les uns des autres. Pour un carré ($n = 4$), les sommets seront à 90 degrés d'intervalle, et ainsi de suite pour d'autres polygones réguliers.

— **Aire d'un polygone régulier** :

$$A = \frac{n \times r^2}{2} \times \sin\left(\frac{2\pi}{n}\right)$$

où $n$ est le nombre de côtés et $r$ est le rayon.

— **Périmètre d'un polygone régulier** :

$$P = n \times d$$

où $d$ est la distance entre deux sommets consécutifs (égal au côté du polygone).

— **Translation d'un polygone régulier** : Pour translater un polygone régulier, on translatera son centre de la même manière que pour un point.

— **Rotation d'un polygone régulier** : Il faudra juste mettre à jour l'angle de rotation du polygone.

— **Redimensionnement d'un polygone régulier** : Pour redimensionner un polygone régulier, on peut ajuster son rayon. Si on redimensionne par un facteur $k$, le nouveau rayon sera :

$$r' = k \cdot r$$

où $r$ est le rayon initial et $r'$ est le nouveau rayon. Les sommets du polygone seront recalculés en utilisant le nouveau rayon.

## Classe `GroupedShape`

La classe `GroupedShape` représente un groupe de formes. Elle permet de regrouper plusieurs formes géométriques en une seule entité, facilitant ainsi leur manipulation collective. Voici des formules utiles pour la classe `GroupedShape` :

— **Aire d'un groupe de formes** : L'aire totale d'un groupe de formes est la somme des aires de chaque forme individuelle :

$$A = \sum_{i=1}^{n} A_i$$

où $A_i$ est l'aire de la $i$-ème forme dans le groupe.

— **Périmètre d'un groupe de formes** : Le périmètre total d'un groupe de formes est la somme des périmètres de chaque forme individuelle :

$$P = \sum_{i=1}^{n} P_i$$

où $P_i$ est le périmètre de la $i$-ème forme dans le groupe.

— **Translation d'un groupe de formes** : Pour translater un groupe, on translatera chaque forme du groupe de la même manière que pour un point.

— **Rotation d'un groupe de formes** : Pour faire tourner un groupe, on fera tourner chaque forme du groupe autour du centre du groupe. Le centre du groupe peut être approximé par le barycentre des centres des formes individuelles.

— **Redimensionnement d'un groupe de formes** : Pour redimensionner un groupe, on peut ajuster les dimensions de chaque forme en fonction d'un facteur d'échelle donné. Si on redimensionne par un facteur $k$, les nouvelles dimensions de chaque forme seront calculées en multipliant les dimensions initiales par $k$.

Il vous reviendra d'écrire les tests unitaires pour vérifier le bon fonctionnement de ces méthodes. Vous rajouterez les tests dans le fichier `GroupedShapeTest.java` sous le dossier `src/test/java/fr/gema/shapes/`.

### Conclusion de la première partie

Une fois que vous aurez implémenté toutes les classes de formes géométriques, assurez-vous que tous les tests unitaires passent correctement. Vous pouvez exécuter tous les tests d'un coup en utilisant la commande suivante dans le terminal à la racine du projet :

mvn **`test`**

Si tous les tests passent, cela signifie que votre implémentation des formes géométriques est correcte et que vous pouvez passer à la seconde partie du projet.

## 6 Partie II : Interface graphique

L'interface graphique de l'application `Shape Studio` est implémentée dans la classe `ShapeStudioGUI`. Cette classe utilise la bibliothèque Java Swing pour créer une interface utilisateur permettant de visualiser et manipuler les formes géométriques. Les fonctionnalités de l'interface graphique incluent :

— Affichage des formes géométriques dans une zone de dessin.

— Outils pour créer, déplacer, redimensionner, faire tourner et supprimer des formes.

— Options pour changer les propriétés des formes (couleur, épaisseur du trait, etc.).

— Gestion des groupes de formes pour manipuler plusieurs formes à la fois.

Vous êtes libres de concevoir l'interface graphique comme vous le souhaitez. Le code pouvant être complexe, il est conseillé de le structurer en plusieurs classes pour une meilleure lisibilité et maintenabilité. Ne mettez pas tout le code dans la classe `ShapeStudioGUI`. Il est recommandé de créer des classes pour les différents composants de l'interface graphique, comme les boutons, et les panneaux de dessin.