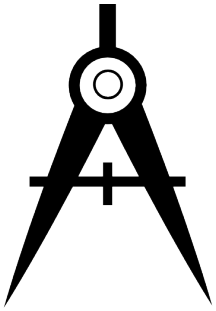

M1 GIL – Mini-projet d'architecture logicielle 2020–2021
Bibliothèque et programmes de manipulation de dessins vectoriels

F. Nicart

À rendre avant le **30 avril 2021**

1 Présentation du mini-projet

Exceptionnellement, compte-tenu du contexte, le projet reprend le sujet des premiers TP plutôt qu'un sujet complètement neuf pour lequel il faut faire toute l'analyse : une bibliothèque de dessin vectoriel. Pour rappel, le code qui était fourni comme base en TP avait grand besoin d'être refactoré. Il vous est donc déconseillé de repartir de là.

Les concepts du dessin vectoriel retenus sont : les lignes, les cercles, les rectangles et les sous-images. Chacun de ces concepts sera décliné en deux types : standard et à main levée (voir la définition données en TP). Pour rappel, l'aléa utilisé pour les dessins à main levée doit être défini au moment de la création de la forme et ne plus changer par la suite. La bibliothèque n'interdira pas de mélanger des formes de types différents mais elle proposera en revanche la notion de « dessin »également décliné en ces deux types qui permettra de travailler facilement soit dans un univers (tracé standard) soit dans l'autre (tracé à main levée). Un dessin pourra disposer de propriétés comme sa dimension, un titre sous forme d'une chaîne arbitraire (indépendante du nom du fichier), le nom de l'auteur...

Le project consiste en quatre programmes séparés à réaliser :

— un éditeur graphique qui permet créer ou de modifier visuellement un dessin vectoriel et de le sauvegarder ;
— un programme en ligne de commande qui produit le dessin miroir (horizontale) d'un dessin de type quelconque ;
— un programme en ligne de commande qui convertit un dessin contenant un type arbitraire de formes en un dessin ne contenant que des formes à tracé standard.
— un programme en ligne de commande qui calcul la surface totale des éléments d'un dessin de type tracé standard.

Un format de sérialisation sera conçu pour la sauvegarde et le chargement des dessins.

1.1 L'éditeur graphique

L'éditeur graphique fonctionnera soit avec des formes standards, soit avec des formes à main levée. Ce choix sera opéré au moment de la création d'un nouveau dessin (choix de l'utilisateur), soit au chargement d'un dessin (information dans le flux).

L'éditeur doit permettre les opérations suivantes :

1. Créer un nouveau dessin vierge d'un type et d'une taille donnée,
2. Charger un dessin existant depuis le disque.
3. Sauvegarder un dessin vers le disque.
4. Ajouter une forme au dessin.
5. Déplacer une forme existante dans le dessin.

6. Supprimer une forme du dessin.
7. Grouper plusieurs formes existantes du dessin, de même niveau, en une seule sous-image.
8. Dégrouper le contenu d'une sous-image.
9. Effectuer le miroir (horizontal) d'une forme quelconque ou du dessin complet.

1.2 Programme miroir en ligne de commande

Ce programme réalise le miroir complet du dessin en transformant les coordonnées de chacun de ses éléments. Ce programme prendra en paramètre le nom d'un fichier contenant un dessin ainsi que le nom du fichier destination dans lequel sera écrit le dessin transformé. Son invocation ressemblera à :

```
> java Mirroir maison.draw retourne.draw
```

ou, en le cachant dans un script *bash*, ressemblera à :

```
> mirroir maison.draw retourne.draw
```

Le programme devra gérer correctement les erreurs en affichant des messages clairs à l'utilisateur.

1.3 Programme de conversion vers dessin standard en ligne de commande

Ce programme prend un dessin quelconque en entrée et remplace toute forme non standard par son équivalent standard. Ce programme prendra en paramètre le nom d'un fichier contenant un dessin ainsi que le nom du fichier destination dans lequel sera écrit le dessin transformé. Son invocation ressemblera à :

```
> java Standard maison.draw maison-std.draw
```

ou, en le cachant dans un script *bash*, ressemblera à :

```
> standard maison.draw maison-std.draw
```

Le programme devra gérer correctement les erreurs en affichant des messages clairs à l'utilisateur.

1.4 Programme de calcul de surface en ligne de commande

Ce programme prend un dessin de formes standards quelconque en entrée et affiche la somme des surfaces de ses formes. Il n'est pas demandé de gérer le cas où les formes se chevauchent. La surface affichée sera donc un maximum.

Ce programme prendra en paramètre le nom d'un fichier contenant un dessin de formes standards :

```
> java Surface maison.draw
```

ou, en le cachant dans un script *bash*, ressemblera à :

```
> surface maison.draw
```

Le programme devra gérer correctement les erreurs en affichant des messages clairs à l'utilisateur.

2 Modalités

— Le travail est à réaliser en binôme maximum [1]
— L'implémentation et le compte-rendu sont à rendre avant le **30 avril 2021**.

3 Travail à rendre

Vous fournirez une archive nommée `projet-al-nom1-nom2.zip` comportant un dossier éponyme. Celui-ci contiendra le code source de votre implémentation, votre rapport au format `PDF`, ainsi que des jeux d'essai de dessins.

Il est important de noter que le rapport aura autant d'importance que l'implémentation. Celui-ci devra contenir :

— Une explication de l'architecture globale illustrée par un ou plusieurs diagrammes UML.

1. Ou en singleton ;-)

— Pour chaque patron mis en oeuvre : la motivation du choix du patron, le diagramme UML de la partie de votre application correspondant au patron instancié (en prenant soin d'y indiquer les acteurs du patron).
— Éventuellement, une liste des patrons que vous avez hésité à utiliser et les raisons pour lesquelles vous les avez écartés.
— Le moins de fautes d'orthographe et de français possible.

Une attention particulière sera portée à la précision sémantique des noms de vos entités (à vos dictionnaires d'anglais !).

Un autre point **important** à garder à l'esprit est que ce mini projet est un prétexte à l'élaboration d'une architecture de taille raisonnable mais respectant les bons principes vus en cours. Par conséquent, le logiciel n'a pas vocation à être abouti ni ergonomique. Par exemple, on pourra renoncer au glisser-lâcher pour invoquer les fonctionnalités dans l'éditeur graphique si cela n'apporte rien d'intéressant à l'architecture. On pourra aussi utiliser des bouchons pour les fonctions pour lesquelles le temps a manqué en le mentionnant dans le rapport et en décrivant le travail à réaliser pour compléter la fonctionnalité, en particulier pour une fonctionnalité pour laquelle une autre fonctionnalité a été développée utilisant le même mécanisme/patron.

4 Quelques indications

— Vous soignerez la modularité de votre application et justifierez dans votre rapport la répartition en paquetages.
— (Comme d'habitude,) vous soignerez la rédaction de vos interfaces. Vous détaillerez leur conception dans votre rapport.
— Vous fournirez une description du format de sauvegarde (par exemple votre DTD) en expliquant sa conception.
— Vous fournirez, en annexe du rapport, la liste complète, par paquetage, des entités de votre application. Vous donnerez en face du nom de chaque entité sa responsabilité dans l'application en une phrase.
— Vous produirez la javadoc de l'ensemble de votre application que vous fournirez, après l'avoir lue, dans un sous-dossier intitulé `javadoc` de votre archive.