ensiie
école nationale supérieure d'informatique
pour l'industrie et l'entreprise

ILO
TP n° 5

Introduction

Le but de ce TP qui s'étalera sur l'ensemble des séances restantes est de vous faire travailler en binôme sur une véritable application : Un mini éditeur de formes géométriques.
L'éditeur est composé :

- D'un modèle de dessin gérant le dessin de plusieurs formes géométriques (Cercle, Ellipse, Rectangle, Rectangle arrondi, Polygone, Polygone régulier et Etoile).
- D'une interface graphique permettant
  - De commander les paramètres du modèle de dessin (type de forme, couleurs de trait et de remplissage, type de trait (continu, pointillé, sans trait) et épaisseur du trait.
  - De dessiner les formes gérées par le modèle de dessin.
  - D'éditer les formes géométriques sélectionnées
    - Déplacement, rotation, facteur d'échelle
    - Changement de style
    - Réordonnancement des figures
  - D'afficher la liste des figures sous la forme d'un arbre qui permettra de sélectionner/désélectionner les figures et pouvant être ordonné suivant
    - Le type de figure, la couleur de remplissage, la couleur de trait et le type de trait.
  - D'afficher dans un panneau d'informations les informations relatives à la figure située sous le curseur de la souris dans la zone de dessin.
  - De filtrer l'affichage des figures suivant plusieurs critères (type de figure, couleur de remplissage ou de trait, types de trait).

Vous pourrez trouver une ébauche du code à réaliser dans l'archive : /pub/ILO/TP5.zip
Documentation Java : http://docs.oracle.com/javase/8/docs/api/

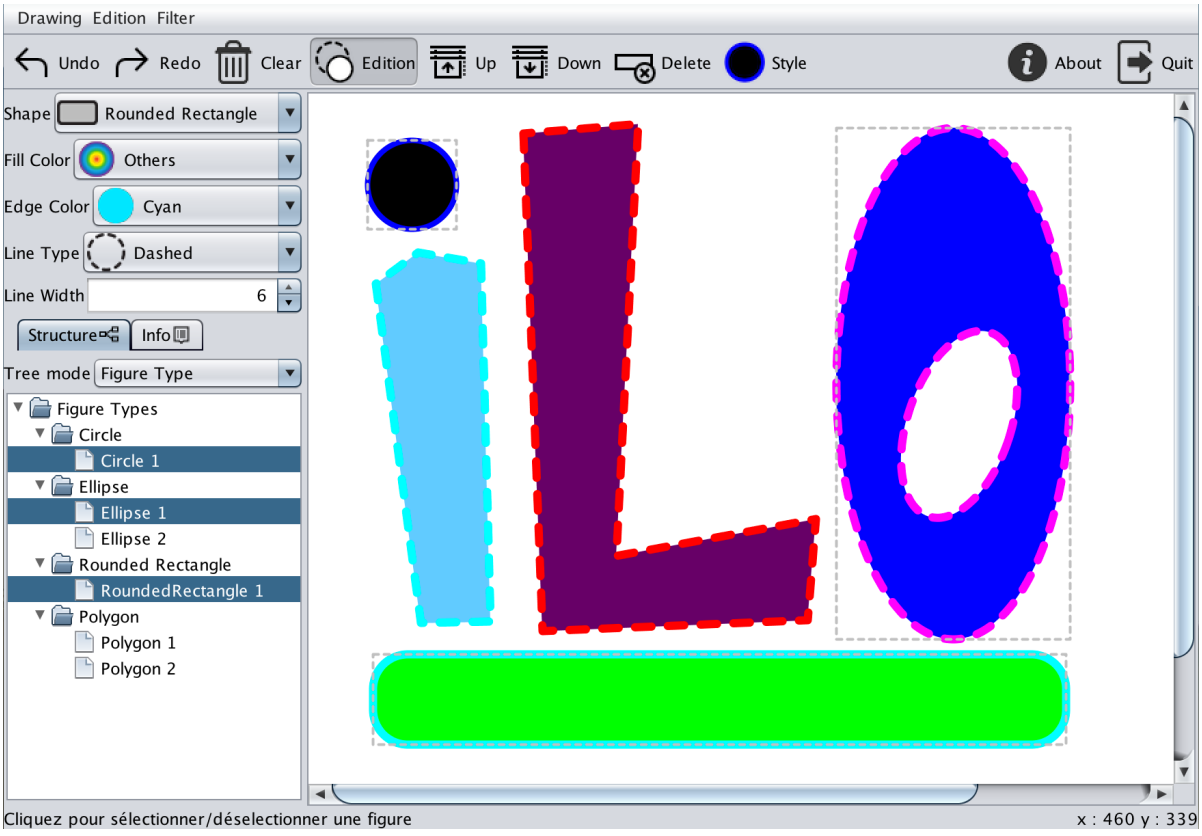

Figure 1 : Editeur de formes géométriques

Figures

La classe Figure qui vous est fournie sera la classe mère de toutes les figures que vous aurez à construire. Les figures sont caractérisées par :

- Shape shape : une forme géométrique à dessiner (au sens de Java)
- Paint edge/fill : les couleurs de remplissage et de trait des figures à dessiner, une figure peut ne pas avoir de remplissage si « fill » est null
- BasicStroke stroke : le style du trait (continu, pointillé), une figure peut ne pas voir de trait si stroke est null.
- Un nom (le type de la figure) et un numéro d'instance unique pour chaque figure du même type qui permettra de distinguer les figures d'un même type entre elles et d'afficher leur nom dans l'arbre des figures ou les informations les concernant dans un panneau d'information dans l'interface graphique.
- 3 transformations affines « translation », « rotation » et « scale » qui peuvent être utilisées pour transformer une figure. Elles sont donc combinées avant le dessin effectif de la figure dans son contexte graphique pour effectivement transformer la figure.
- Un état de sélection indiquant si la figure est sélectionnée ou non.

Les figures doivent pouvoir répondre aux commandes suivantes :

- Création d'une nouvelle figure.
- Déplacement du dernier point de la figure (ce qui permet de créer des figures de tailles non nulles à la souris). Cette méthode pourra être assortie de toutes les méthodes nécessaires à la création d'une nouvelle figure à l'aide d'évènements souris.
- Dessin de la figure dans un contexte graphique (Graphics2D) en utilisant les shape, edge, fill et stroke ce qui permettra au modèle de dessin de dessiner l'ensemble des figures dans un JPanel de dessin. On rajoutera le dessin de la boite englobante de la figure si celle-ci est sélectionnée.
- Point contenu : détermine si un point donné est contenu à l'intérieur de la figure. Ceci permettra plus tard de remplir un panneau d'informations relatives à une figure située sous le pointeur de la souris dans la zone de dessin ou de sélectionner / désélectionner une figure.
- Transformation de la figure : translation, rotation, facteur d'échelle.
- Plus des accesseurs pour les différents attributs qui peuvent être utilisés et/ou modifiés depuis l'interface graphique ou le modèle de dessin.

Modèle de dessin

Le modèle de dessin (la classe Drawing) contient une collection de figures (Figure : classe mère de toutes les figures que vous aurez à construire), ainsi que des méthodes pour :

- Mettre en place les styles de la prochaine figure à créer (le type de figure, les couleurs de trait et de remplissage et le style du trait). Les couleurs et les types de trait seront fournis par des FlyweightFactories (dans le package utils) qui permettent de ne fournir qu'une seule référence d'une couleur ou d'un style de trait donné afin qu'ils puissent être partagés par plusieurs figures.
- Initier une nouvelle figure à la position d'un point p (qui sera fourni par le pointeur de la souris) et l'ajouter à la collection des figures.
- Récupérer la dernière figure de la collection (celle en cours de construction).
- Récupérer la dernière figure de la collection qui contient un point p (pour afficher les caractéristiques de cette figure dans un panneau d'informations, ou pour sélectionner / désélectionner une figure en mode édition).
- Supprimer l'ensemble des figures.
- Fournir un flux de figures afin qu'elles puissent être parcourues (en vue de les dessiner par exemple), mais aussi filtrées par des prédicats afin de n'afficher que les figures correspondant aux prédicats.
    - Mettre en place différents filtres sur ce flux de figures (par type de figure, par couleur de remplissage ou de trait et par type de traits.
- Réaliser diverses opérations sur les figures sélectionnée comme
    - Supprimer les figures sélectionnées
    - Remonter ou descendre les figures sélectionnées dans la liste des figures

- Etc.…

Le modèle de dessin communique suivant deux modes :

- Le modèle de dessin est piloté par les différents contrôleurs (xxxListener) associés aux divers widgets de l'interface graphique.
- Le modèle de dessin est un Observable qui met à jour son ou ses Observers lorsque son état interne change (lorsqu'une figure est ajoutée, modifiée, supprimée ou lorsque les filtres changent). Le JPanel de dessin (DrawingPanel) est donc un Observer du modèle de dessin.

Infrastructure de communication

Certains widgets comme les boutons (JButton), les listes déroulantes (JComboBox) ou les items de menus (JMenuItem/JCheckBoxMenuItem) peuvent modifier le modèle de dessin si on leur attache un contrôleur adéquat : un *ActionListener* pour un bouton ou un item de menu et un *ItemListener* pour une liste déroulante. Ces widgets produisent des événements de haut niveau (*ActionEvent* pour un bouton ou un item de menu, *ItemEvent* pour une liste déroulante). Lorsque l'on attache un contrôleur à un widget, celui-ci peut être unique (à savoir uniquement attaché à CE widget), auquel cas il pourra être implémenté en tant que classe anonyme (ou lambda) directement dans l'ajout du contrôleur (dans l'appel de la méthode addXXXListener). Mais si l'on souhaite réutiliser un même contrôleur sur plusieurs widgets ou à plusieurs endroits de l'application on les implémentera en tant que classes internes (dans la fenêtre principale par exemple).
Une extension de « ActionEvent » est fournie par l'interface « Action » qui combine une description de l'action associée à une icône et la gestion des ActionEvents dans une seule classe afin que cette action puisse être attachée à plusieurs widgets comme un bouton ET un item de menu (voir Figure 4, page 4). Plusieurs « Actions » (certaines à compléter) vous sont fournies et pourront être associées à vos boutons et items de menu comme par exemple :

- QuitAction : action à réaliser pour quitter l'application
- UndoAction : action à réaliser pour annuler le dessin de la dernière figure
- ClearAction : action à réaliser pour effacer l'ensemble des figures
- AboutAction : action à réaliser pour afficher une boite de dialogue « A propos … » de l'application.
- ShapeFilterAction : pour mettre en place le filtrage sur les différents types de figures.
- LineFilterAction : pour mettre en place le filtrage des figures sur la base des différents types de trait.
- …

D'autres widgets comme les JPanel ou les JLabel permettent d'afficher des informations (respectivement du dessin ou du texte). Par ailleurs tous les widgets émettent des événements de bas niveau dont font partie les événement souris (MouseEvent) que nous souhaitons utiliser pour dessiner nos figures. La construction graphique des figures dans le panel de dessin fera donc appel à un MouseListener /MouseMotionListener. Néanmoins, chaque type de figure peut nécessiter un contrôle des évènements souris différent. La classe « figures.listeners.creation.AbstractCreationListener » fournit un squelette de base des listeners que vous aurez à créer pour construire graphiquement les différents types de figures.

Plus généralement, le package « figures.listeners » contient les différents Mouse[Motion]Listeners utilisés pour gérer les évènement souris pour créer ou modifier des figures.

Ces différents éléments permettent de mettre en place une architecture Model / View / Controller vue en cours, pour réaliser la construction, le dessin et la modification des figures :

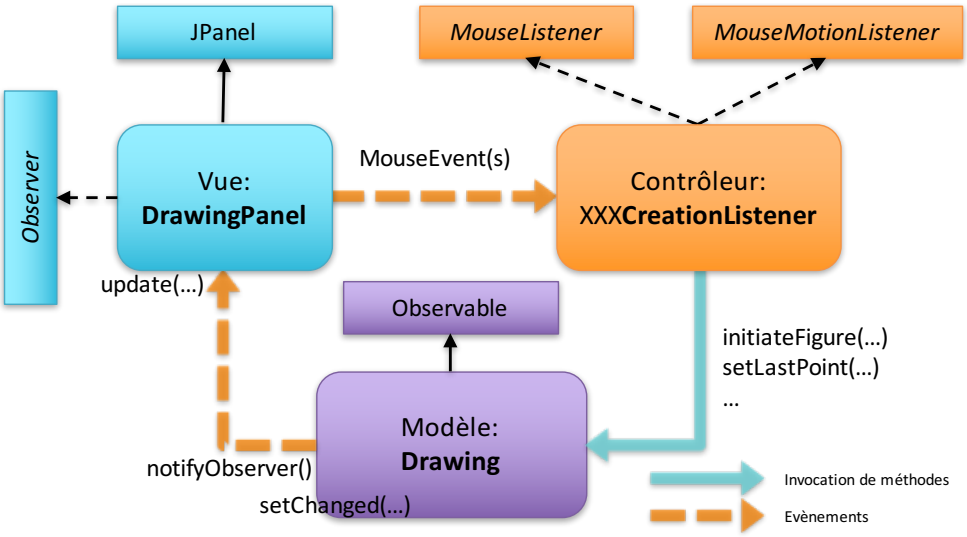

Figure 2 : le MVC entre les modèle de dessin et l'interface graphique

De manière générale les évènements émis par les widgets sont interceptés par des xxxListeners qui servent de contrôleur pour modifier le modèle de dessin, ou plus directement divers widgets.

Architecture de l'interface graphique

L'interface graphique à construire dans la classe EditorFrame se structure de la manière suivante :

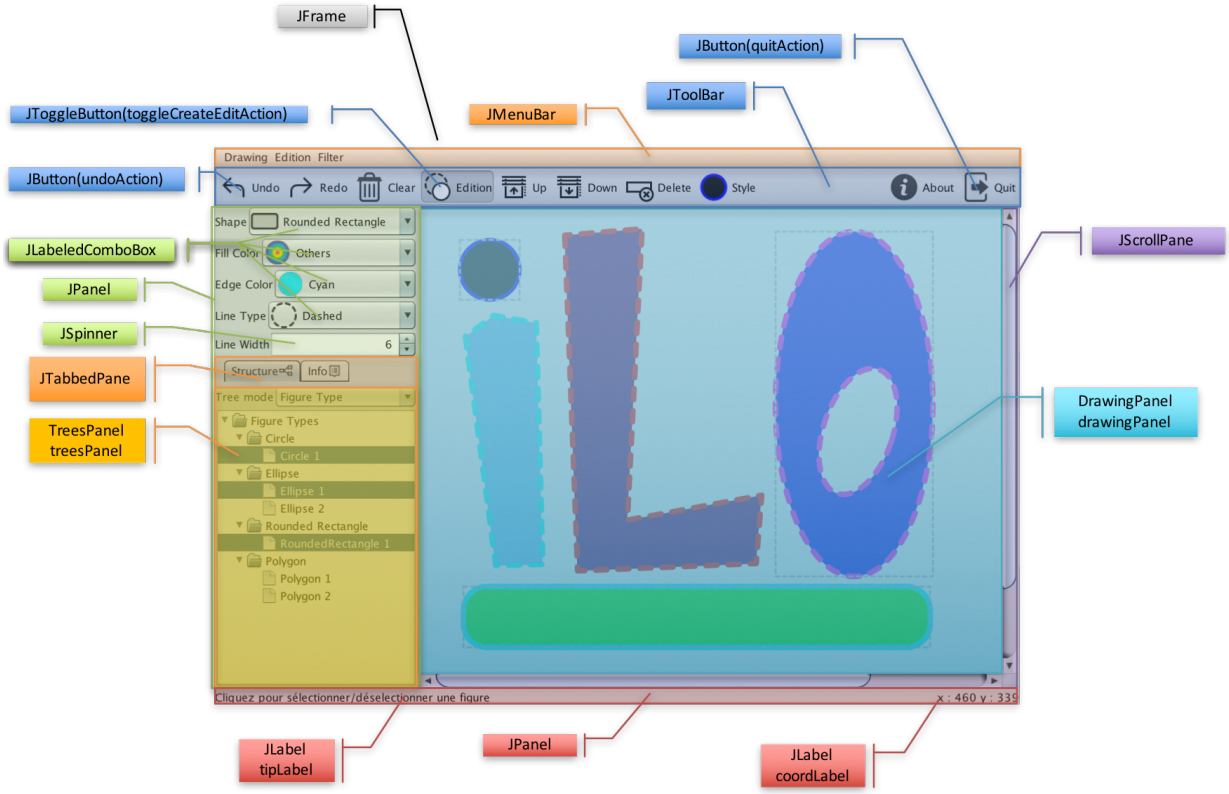

Figure 3 : Structure de l'interface graphique

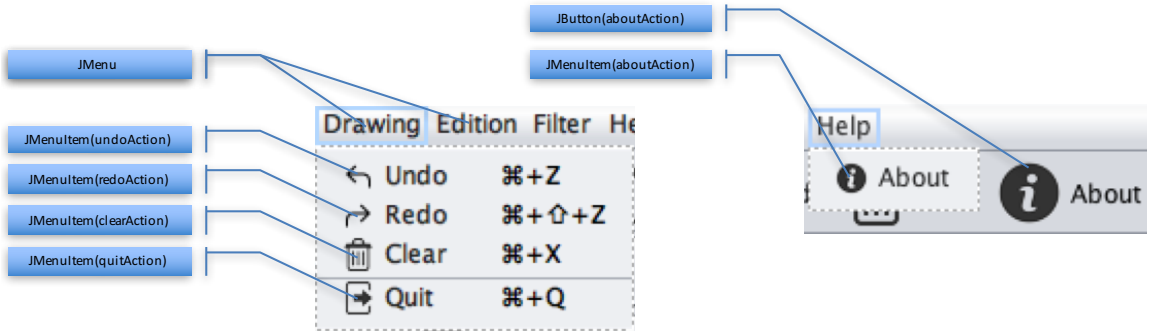

Figure 4 : Actions dans les menus et boutons

La fenêtre principale (EditorFrame héritière de JFrame)

La fenêtre principale contient les différents widgets qui composent l'interface graphique. Ces widgets sont organisés (placés et dimensionnés) en utilisant un LayoutManager, en l'occurrence un BorderLayout permettant de placer les widgets en utilisant les directions cardinales (North : en haut, South : en bas, East : à droite, West : à gauche et Center : au centre).

Barre de menus (JMenuBar)

La barre de menu contient trois menus et chaque item de menu est associé à un « ActionListener » anonyme ou bien à une « Action » (par exemple « Undo » à une « UndoAction ») :

- Le menu Drawing contient les items (de type JMenuItem) suivants :
    - Undo : pour annuler la dernière opération sur les figures.
    - Redo : pour refaire la dernière opération annulée sur les figures.
    - Clear : pour effacer l'ensemble des figures.
    - Quit : pour quitter l'application.
- Le menu Edit contient les items (de type JMenuItem ou JCheckBoxMenuItem) suivants :
    - Edition : Un JCheckBoxMenuItem permettant de passer en mode « édition des figures » ou de revenir au mode « création des figures ».
    - Up : Pour remonter les figures sélectionnées dans la liste des figures.
    - Down : Pour descendre les figures sélectionnées dans la liste des figures.
    - Delete : Pour supprimer les figures sélectionnées
    - Style : Pour appliquer aux figures sélectionnées les styles courants de remplissage et de trait.
- Le menu Filter contient les items (de type JCheckBoxMenuItem) suivants :
    - Filtering : pour mettre en place ou annuler le filtrage des figures
    - Sous menu Figures :
        - Contient un item pour mettre en place un filtre pour chaque type de figure.
    - Sous menu Colors :
        - Fill Color : pour mettre en place le filtrage des figures sur la base de leur couleur de remplissage (couleur à choisir dans une boite de dialogue de choix de couleurs).
        - Edge Color : pour mettre en place le filtrage des figures sur la base de leur couleur de trait (couleur à choisir dans une boite de dialogue de choix de couleurs).
    - Sous menu Strokes :
        - Contient un item pour mettre en place un filtre pour chaque type de trait de figure.
- Help
    - About : pour afficher la boite de dialogue « A propos … »

Barre d'outils (JToolBar)

La barre d'outils contient des boutons associés aux mêmes actions que dans les menus :

- Undo
- Redo
- Clear
- Edition
- Up
- Down
- Delete
- Style
- About
- Quit (à droite de la barre d'outils)

Panneau d'outils (JPanel à gauche)

Le panneau d'outils contient les différentes listes déroulantes (en l'occurrence des JLabeledComboBox qui vous sont fournis) permettant de choisir les options de dessin et d'afficher des informations sur les figures :

- Choix du type de figure.
- Choix de la couleur de remplissage.
- Choix de la couleur de trait.
- Choix du type de trait.
- Epaisseur du trait (JSpinner).
- Un JTabbedPane contenant 2 onglets
    - Un premier onglet « structure » contenant un TreesPanel qui lui-même contient
        - Un JCombobox permettant de choisir le type d'arbre à afficher
        - Un JTree contenant l'arbre des figures, le contenu de cet arbre est géré par des classes implémentant l'interface TreeModel et qui permettent de transformer le contenu du modèle de dessin (une simple liste de figures) en un arbre structuré suivant différentes caractéristiques des figures (type, couleurs de remplissage ou de trait et type de trait).
          Cet arbre de figures peut prendre plusieurs formes :
            - Une simple liste des figures.
            - Un arbre de figures dont les nœuds de niveau 1 sont les types de figure et les nœuds de niveau 2 les figures.
            - Un arbre de figures dont les nœuds de niveau 1 sont les couleurs de remplissage et les nœuds de niveau 2 les figures remplies avec cette couleur.
            - Un arbre de figures dont les nœuds de niveau 1 sont les couleurs de trait et les nœuds de niveau 2 les figures possédant un trait de cette couleur.
            - Un arbre de figures dont les nœuds de niveau 1 sont les types de trait et les nœuds de niveau 2 les figures possédant ce type de traits.
    - Un second onglet « Info » contenant Panneau d'information (InfoPanel). Le panneau d'information contient des labels mis à jour lorsque le pointeur de la souris se trouve au-dessus d'une figure dans le panel de dessin.

Les JLabeledComboBox sont des combobox associés à un titre et dont chacun des éléments est composé d'un texte associé à une image. L'image chargée pour un item de la liste correspond à un fichier image (.png) portant le même nom que le titre de l'item et se trouvant dans le package images.

Barre d'état (JPanel en bas)

La barre d'état contient deux JLabel :

- tipLabel dans lequel le xxxCreationListener de la figure en cours affiche des conseils utilisateur.
- coordLabel dans lequel le panel de dessin affiche les coordonnées courantes du pointeur de la souris.

Zone de dessin (DrawingPanel à droite)

Le DrawingPanel est un observateur du modèle de dessin (Drawing) mais aussi un MouseListener et un MouseMotionListener afin de :

- Le mettre à jour : redessiner les figures du modèle de dessin lorsque celui-ci change.
- Afficher les coordonnées du pointeur de la souris dans un label de la barre d'état.
- Lui affecter le XXXCreationListener relatif à la figure en cours de création ou bien en cours de modification. Les différents listeners affectés à la zone de dessin pour écouter les évènements souris sont les suivants :
    - AbstractFigureListener : classe mère de tous les listeners concernant la manipulation de figures

        - SelectionFigureListener : listener simple permettant de changer une figure située sous le curseur de l'état non sélectionné à l'état sélectionné ou inversement en cliquant dans la figure.
        - AbstractCreationListener : classe mère de tous les listeners destinés à la création de nouvelles figures
            - RectangulaShapeCreationListener : listener utilisé lors de la création de toutes les figures inclues dans un rectangle : Ellipse, Rectangle, RoundedRectangle que l'on peut dessiner en pressant le bouton 1 de la souris, puis en tirant la souris (en maintenant le bouton enfoncé), puis en relâchant le bouton à une position différente du point de départ.
            - …
        - AbstractTransformShapeListener : classe mère de tous les listeners destinés à transformer géométriquement les figures sélectionnées (grâce aux transformations affines « translation », « rotation », « scale » contenues dans les figures). Pour pouvoir utiliser plusieurs Transform listeners simultanément, il suffit de leur affecter un masque lié à une touche de contrôle (par exemple Shift-Click pour tourner et Alt-Click pour changer l'échelle).
            - MoveShapeListener : listener utilisé pour déplacer les figures
            - RotateShapelistener : listener utilisé orienter les figures
            - ScaleShapeListener : listener utilisé pour changer l'échelle des figures

Travail à réaliser

Ce travail est à réaliser en binôme. Si vous souhaitez travailler en monôme, nous adapterons le barème en conséquence. Attention, comme pour le TP Ensembles, le plagiat sera sanctionné.

Vous ne partirez jamais de rien pour réaliser le travail demandé :

- **Interface graphique** : vous disposez d'un squelette de l'interface graphique dans la classe EditorFrame contenant déjà les actions à réaliser (à compléter toutefois). Vous pourrez donc repartir de cette trame ou bien faire votre propre interface graphique.
- **Figures** : vous disposez déjà de la figure Rectangle ainsi que du RectangularShapeCreationListener qui sert à dessiner ces figures dans le DrawingPanel.
- **Listeners** : Vous disposez du SelectionFigureListener qui permet de sélectionner une figure située sous le curseur dans le DrawingPanel ainsi que du MoveShapeListener qui permet de déplacer une figure sélectionnée.
- **Filtres** : vous disposez d'un filtre générique FigureFilter<T> que vous pouvez appliquer au flux de figures et dont vous pourrez hériter pour faire vos filtres de type de figure, couleur de remplissage, etc. Vous disposez aussi d'un filtre composite FigureFilters<T> constitué de multiples FigureFilter<T> dont le test renvoie vrai pour un élément si tous les filtres qu'il contient renvoient vrai.
- **Arbres** : Vous disposez d'une classe abstraite AbstractTreeModel qui contient l'essentiel de l'algorithmique pour (re)construire un arbre de figures dans le JTree lorsque le modèle de dessin change. Vous disposez aussi d'une première implémentation d'arbre (FigureTreeModel) qui construit simplement une liste de figures sous une racine commune.
- **Utilitaires** : Vous disposez d'un certain nombre de classes utilitaires comme :
    - Factories : Les factories dont vous disposez vous permettent de stocker des éléments communs réutilisés à plusieurs endroits de votre programme comme les icônes, les couleurs ou les types de traits.
        - IconFactory : classe permettant de charger une icône depuis un fichier et de la refournir plus tard sans avoir à recharger l'icône.
        - PaintFactory : classe permettant de centraliser toutes les couleurs appliquées aux figures, pour le remplissage ou la couleur de trait.
        - StrokeFactory : classe permettant de centraliser tous les types de contour appliqués aux figures.

Le travail demandé est donc le suivant :

1. Complétez la construction de l'interface graphique : EditorFrame.
2. Complétez le modèle de dessin : Drawing.
3. Complétez les différentes actions présentes dans l'interface graphique :
    a. ClearAction : effacement de l'ensemble des figures.
    b. ToggleCreationAction : permet de passer du mode création des figures au mode édition des figures.
        i. Mode création : ajout de nouvelles figures.
        ii. Mode édition : modification des figures sélectionnées.
    c. DeleteAction : suppression des figures sélectionnées.
    d. StyleAction : application du style courant (couleurs de remplissage et trait + type de trait) à l'ensemble des figures sélectionnées.
    e. MoveUpAction : déplacement des figures sélectionnées en tête de liste des figures (en conservant l'ordre relatif des figures sélectionnées).
    f. MoveDownAction : déplacement des figures sélectionnées en fin de liste des figures (en conservant l'ordre relatif des figures sélectionnées).
    g. xxxFilterAction : Les différentes action permettant de mettre en place ou d'enlever les différents filtres sur les différents types de figures et de traits ainsi que les filtres sur la couleur de remplissage et la couleur de trait.
    h. UndoAction et RedoAction (voir plus bas)
4. Complétez la classe Figure et ajoutez différents types de figures : Ellipse, RoundedRectangle (rectangles aux coins arrondis), Polygone quelconque, Polygone régulier (NGones) et Etoile. Vous aurez probablement besoin pour ce faire d'implémenter de nouveaux CreationListeners.
5. Ajoutez des TransformListeners pour pouvoir faire tourner et changer l'échelle des figures sélectionnées.
6. Ajoutez les filtres nécessaires pour pouvoir filtrer le flux de figures (méthode stream de la classe Drawing) suivant :
    a. Le type de figure (en utilisant le type de figure courant du modèle de dessin).
    b. La couleur de remplissage (en utilisant la couleur de remplissage courante du modèle de dessin).
    c. La couleur de contour (en utilisant la couleur de contour courante du modèle de dessin).
    d. Le Type de contour (en utilisant le type de trait courant du modèle de dessin).
7. Ajoutez différents modèles d'arbres à afficher dans le TreesPanel
    a. Figures groupées par type de figure.
    b. Figures groupées par couleur de remplissage.
    c. Figures groupées par couleur de contour et/ou type de trait.
8. Ajoutez une fonctionnalité de Undo / Redo en utilisant le Design Pattern Memento :
    a. Le modèle de dessin (la classe Drawing) crée une sauvegarde son état interne à un instant donné en créant une instance d'une classe « State ». Typiquement lorsque l'on s'apprête à changer le modèle de dessin en rajoutant / enlevant / modifiant des figures.
    b. L'interface stocke et gère ces états sauvegardés et lors des actions de undo ou redo remet en place un état particulier dans le modèle de dessin.