

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 réalisée avec JavaFX 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 à la souris.
 - De sélectionner des figures puis d'éditer les figures sélectionnées :
 - Déplacement, rotation, facteur d'échelle.
 - Changement de style.
 - Réordonnancement des figures.
 - Destruction des figures sélectionnées.
 - De filtrer l'affichage des figures suivant plusieurs critères (type de figure, couleur de remplissage ou de trait, types 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 gérer les Undos / Redos à chaque fois qu'une figure est ajoutée, supprimée ou modifiée.
 - D'effacer l'ensemble des figures

Vous pourrez trouver une ébauche du code à réaliser dans l'archive :

/pub/ILO/TPEditor/FiguresEditor.zip

Documentation Java : <https://docs.oracle.com/en/java/javase/11/docs/api/>

Documentation JavaFX : <https://openjfx.io/javadoc/15/>

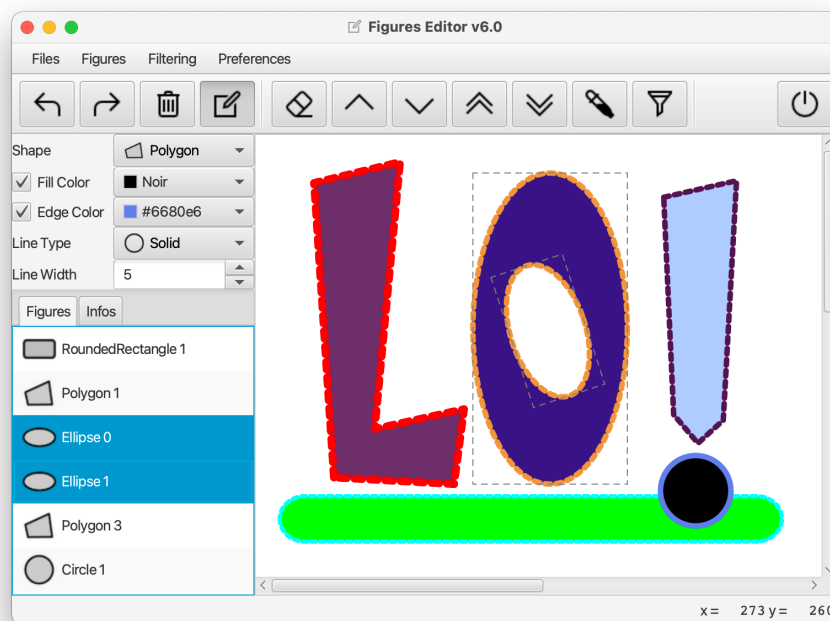


Figure 1 : Éditeur de formes géométriques

Figures

La classe Figure qui vous est fournie sera la classe mère de toutes les figures que vous aurez à construire.

Les figures sont composées (entre autres) de :

- **Shape shape** : une forme géométrique à dessiner (au sens de JavaFX). Cette forme est la représentation géométrique de notre figure dans la zone de dessin (un simple `Panel`).
- **Color edge/fill** : les couleurs de remplissage et de trait des figures à dessiner, une figure peut ne pas avoir de couleur remplissage ou de couleur de trait, mais pas les deux en même temps. L'une ou l'autre doit subsister.
- **LineStyle** : le style du trait (continu, pointillé ou sans trait).
- Une épaisseur de trait (comprise entre 1 et 32).
- 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 la liste des figures ainsi que dans un panneau d'information dans l'interface graphique.
- **Rectangle selectionRectangle** : un rectangle qui permettra de représenter la sélection / désélection de cette figure dans la zone de dessin.
- **Group root** un groupe JavaFX contenant `shape` et éventuellement `selectionRectangle` afin que les deux restent toujours liées même si la figure subit des transformations géométriques (translation, rotation, facteur d'échelle). C'est donc sur ce groupe que s'appliqueront ces transformations.
- Un état de sélection indiquant si la figure est sélectionnée ou non. Lorsque la figure est sélectionnée, `selectionRectangle` est ajouté à `root` et inversement lorsque la figure est désélectionnée.
- Puisque JavaFX implémente un graphe de scène, dessiner une figure dans la zone de dessin consiste simplement à ajouter le `root` d'une figure aux enfants de la zone de dessin. Et effacer une figure consiste à retirer le `root` d'une figure de la zone de dessin.

Vous devrez créer les classes filles suivantes : `Circle`, `Rectangle`, `RoundedRectangle` (un rectangle avec des coins arrondis), `Polygon` (composé d'un nombre de points > 2), `Ngon` (polygone régulier à n arêtes, avec $n \geq 3$) et `Star` (étoile à n branches avec $n \geq 3$).

Les figures contiennent les méthodes nécessaires pour :

- Créer une nouvelle figure à un point donné de taille 0 : Ce constructeur sera utilisé pour créer une nouvelle figure à partir d'événements souris.
- Créer une nouvelle figure à un point donné avec l'ensemble de ses caractéristiques : Par exemple le rayon pour un cercle, la largeur et la hauteur pour une ellipse ou un rectangle, les points pour un polygone, ou encore les rayons extérieur, intérieur et le nombre de branches pour une étoile.
- Déplacer le 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'événements souris.

Ainsi que

- Les accesseurs permettant d'obtenir (et/ou) de modifier les attributs mentionnés plus haut.
- Un groupe de méthode permettant d'obtenir :
 - Le nom de la figure constitué du nom de la classe suivi par un numéro d'instance unique dans chaque type de figure.
 - Le centre de la figure.
 - La largeur de la figure.
 - La hauteur de la figure.
 - Le point en haut à gauche et le point en bas à droite.
 - Ces méthodes pourront être utilisées pour déterminer le `selectionRectangle` et/ou pour remplir un panneau d'information lorsque la souris passera au-dessus d'une figure dans la zone de dessin.
- Les opérations classiques de la class `Object` : `equals`, `hashCode`, `toString`. Une méthode `equals(Figure)` sera implémentée dans chaque sous classe. Cette méthode est indispensable car elle permettra de comparer les figures lors de la création ou de la mise en place des Mementos dans la gestion des Undos / Redos afin de ne pas produire de doublons dans les Mementos.

- Toutes les opérations sur les figures faisant intervenir des événements souris seront déléguées à des outils (rassemblés dans le package `tools`. Voir section Les Outils (package `tools`) ci-dessous)

Modèle de dessin

Le modèle de dessin représente le cœur de notre application (notre modèle de données) et est implémenté dans la classe `Drawing`.

La classe `Drawing` se présente comme suit :

- Elle contient une liste de Figures (les figures à dessiner et à gérer).
- Elle se comporte comme une Liste Observable de Figures (en héritant de la classe `ModifiableObservableListBase<Figure>`) ce qui lui permettra d'être utilisée comme contenu d'une `ListView<Figure>` JavaFX.
- Elle se comporte aussi comme un `Originator<Figure>` afin de pouvoir créer un Memento sauvegardant sa liste de figures ou de mettre en place une nouvelle liste de figures à partir d'un Memento qui lui est fourni. Ceci permettra de gérer les Undos / Redos pour toutes les opérations qui créent, suppriment ou modifient les figures de la liste.
- Et enfin elle implémente l'interface `ListChangeListener<Figure>` afin qu'elle puisse être notifiée des changements de sélection dans la `ListView` qui servira à afficher la liste de figures pour que ces changements de sélection puissent se refléter dans la zone de dessin où sont dessinées les figures.
- La classe `Drawing` gère aussi les paramètres du dessin:
 - Les couleurs de remplissage et de trait à appliquer aux prochaines figures à dessiner
 - Le type et la largeur du trait
 - Ces paramètres sont implémentés en utilisant des propriétés JavaFX afin que ces propriétés puissent être liées (bind) à des éléments de l'interface graphique. Le changement d'une propriété dans l'interface graphique (par exemple la couleur de remplissage) sera alors automatiquement reporté dans la propriété du modèle de dessin sans que l'on ait besoin d'implémenter un callback.
- On peut filtrer le contenu de la classe `Drawing` avec la méthode `filtered(Predicate<Figure> predicate)` qui permet d'appliquer un prédicat sur l'ensemble des figures et de renvoyer la liste des figures qui ont vérifié ce prédicat. Cela nous sera utile pour filtrer les figures en fonction :
 - Du type de figure.
 - Des couleurs de remplissage ou de trait.
 - Du type de trait et de l'épaisseur du trait.

Les Outils (package `tools`)

Le package `tools` contient tous les outils pour réaliser des opérations (sur les figures ou pas) à partir d'événements souris (`MouseEvent`). Parmi ces différentes opérations on peut trouver :

- Le dessin de nouvelles figures dans la zone de dessin (classe `tools.creation.AbstractCreationTool` et héritières) :
 - Pressed → Drag → Release : pour les figures rectangulaires comme les cercles, ellipses et rectangles (classe `tools.creation.RectangularShapeCreationTool`).
 - Click → Click → ... → Right Click : pour ajouter des points à un polygone et terminer l'ajout des points avec un clic droit par exemple.
- La mise à jour de la position du curseur de la souris dans la zone de dessin (en bas à droite dans l'UI) en utilisant les événements `MOUSE_MOVED` et `MOUSE_EXITED` dans la zone de dessin (classe `tools.CursorTool`).
- La mise à jour d'un panneau d'information des figures lorsque le curseur de la souris passe au-dessus d'une figure dans la zone de dessin avec les événements `MOUSE_ENTERED_TARGET` et `MOUSE_EXITED_TARGET` (classe `application.panels.InfoPanelController`).
- La sélection ou la désélection de figures directement dans la zone de dessin (classe `tools.SelectionTool`).
- Les transformations géométriques appliquées aux figures dans la zone de dessin (translation, rotation, facteur d'échelle) (classe `tools.TransformTool`)

- Vous serez probablement amenés à créer vos propres outils de création de figures dans les package `tools.creation`.

Interface Graphique avec JavaFX

Nous allons utiliser la séparation entre la définition de l'interface graphique qui sera définie dans le fichier `EditorFrame.fxml` dans le package `application` et le Contrôleur de cette interface graphique défini dans la classe `Controller` du même package. Le programme principal (class `Main`) se charge de lire le fichier `EditorFrame.fxml` et d'instancier le contrôleur.

Dans le cadre d'une architecture MVC le fichier `EditorFrame.fxml` définit la Vue, la classe `Drawing` représente le Modèle et la classe `Controller` comme son nom l'indique représente le Contrôleur et correspond au moteur de notre application.

Vue

La vue définie dans `EditorFrame.fxml` peut être éditée avec le programme `ScenBuilder` (voir Mise en Place de JavaFX, page 8) et devrait ressembler à ceci :

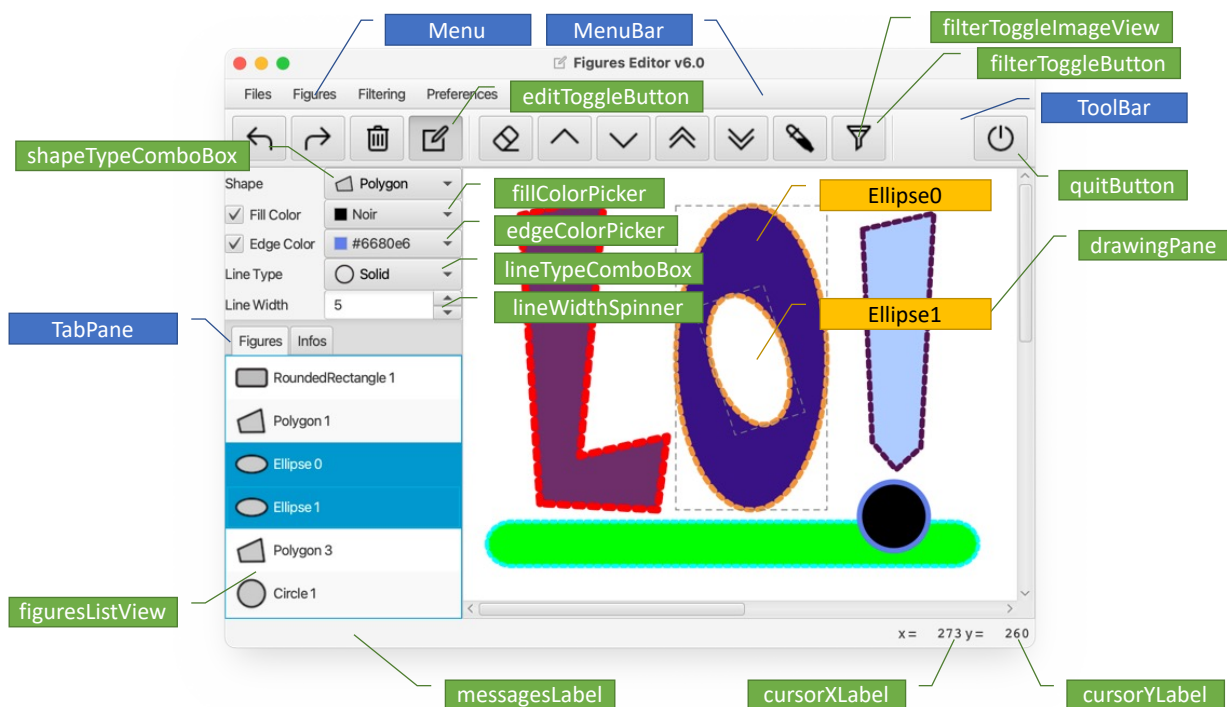


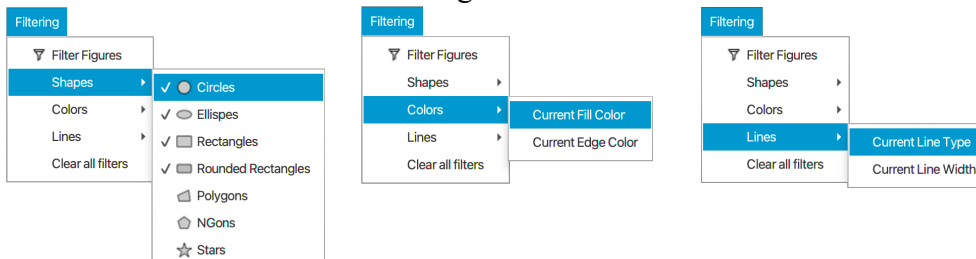
Figure 2 : éléments de l'interface graphique

Parmi les autres éléments de la Vue, on peut trouver :

Les menus



Détail des sous-menus de « Filtering »



Widgets customisables

Les `ListView<X>` et `ComboBox<X>` contiennent « effectivement » des éléments de type `X` en JavaFX. Ils sont par ailleurs customisables : On peut créer une Vue customisée (dans un fichier FXML associé à son propre contrôleur) pour les cellules d'un `ListView` ou d'un `ComboBox`.

Vous pourrez trouver dans package `application.cells` un couple de fichiers `FigureCell.fxml` / `FigureCellController.java` permettant de customiser l'affichage des Figures dans le `ListView<Figure>` `figuresListView` (voir Figure 2, page 4). Vous pourrez vous en inspirer pour faire vos propres `CustomCells` pour les types de figures et les types de lignes comme le montre la Figure 3, ci-dessous.

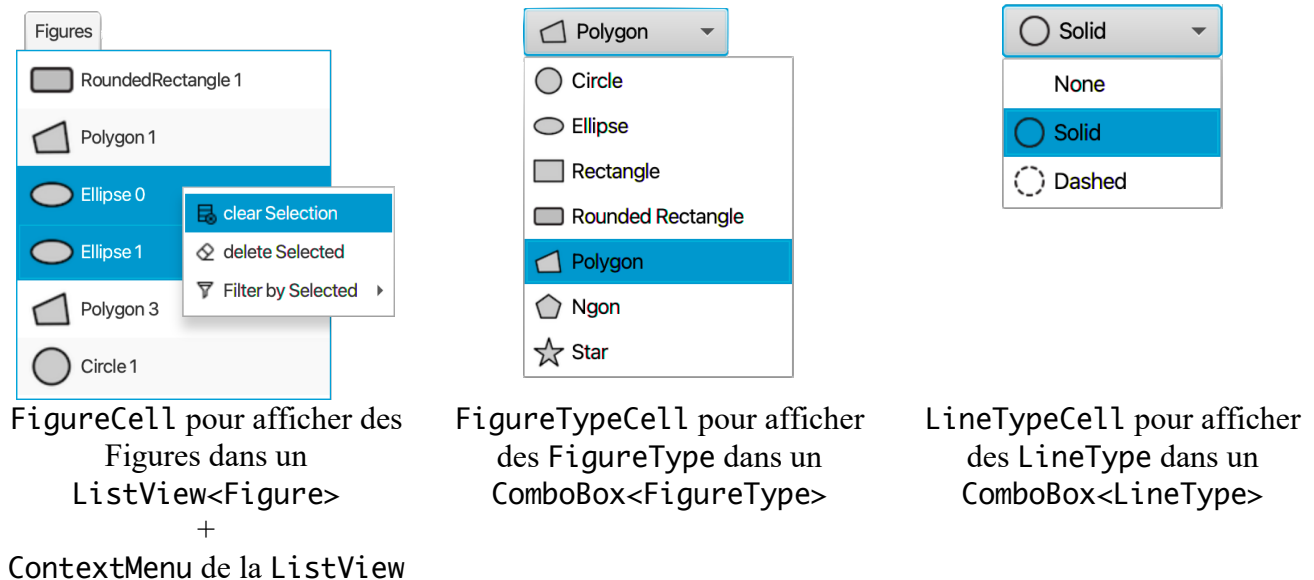


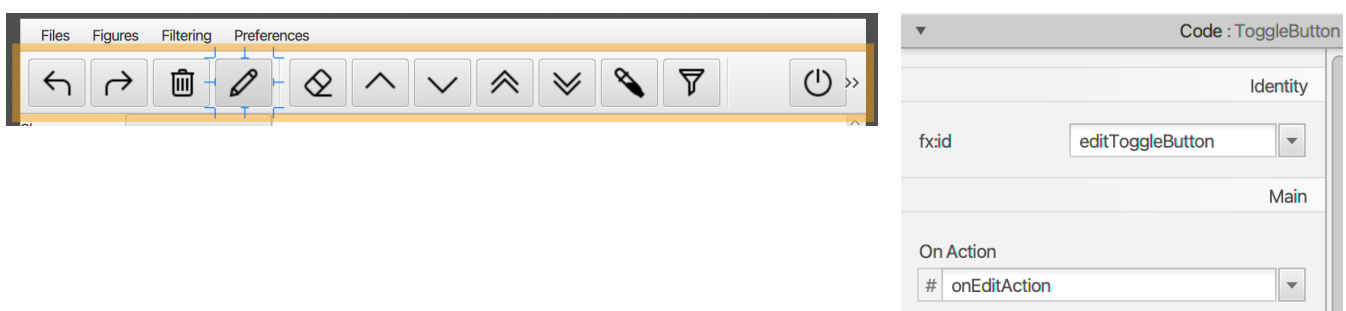
Figure 3 : Exemples de CustomCells

Contrôleur























Le contrôleur contient un certain nombre d'attributs (annotés avec l'annotation `@FXML`) qui reflètent certains éléments de l'interface graphique définie dans `EditorFrame.fxml` afin qu'ils puissent être utilisés dans le contrôleur. Vous devrez veiller à ce que chacun de ces attributs soit bien référencé dans le fichier `EditorFrame.fxml` avec un `fx:id` sans quoi leur utilisation lors du chargement du fichier FXML déclenchera l'erreur "Can't load FXML file ...".

Le contrôleur contient aussi des méthodes `onXXXAction(ActionEvent event)` annotées avec l'annotation `@FXML` et que l'on peut associer à un ou plusieurs éléments de la Vue.

Par exemple, en éditant le fichier `EditorFrame.fxml` dans SceneBuilder, et en sélectionnant le `ToggleButton` `editToggleAction`, on peut lui associer le callback `onEditAction` dans le Contrôleur (Voir Figure 4 ci-dessous) en remplissant le champ "On Action" (le champ `fx:id` n'est pas nécessaire dans ce cas mais peut être utilisé si l'on a besoin d'une référence à ce bouton dans le contrôleur (pour changer son icône par exemple)) :



Les différentes actions à réaliser dans le Controller ou bien au travers d'outils (voir section Les Outils (package tools), page 3) mis en place par le Controller sont les suivantes :

- File
 - Save (inutilisé pour l'instant)
 - Load (inutilisé pour l'instant)
 -  Quit : pour quitter l'application.
-  Création de figures en utilisant des événements souris en utilisant les paramètres suivants :
 - Shape Types
 -  Circle
 -  Ellipse
 -  Rectangle
 -  RoundedRectangle
 -  Polygon
 -  NGon
 -  Star
 - Fill Color en utilisant un ColorPicker
 - Stroke Color en utilisant un ColorPicker
 - Line Type:
 -  Solid
 -  Dashed
 - None
 - Line Width allant de 1 to 32 en utilisant un Spinner
-  Edition
 -  Undo : pour annuler la dernière action.
 -  Redo : pour refaire la dernière action.
 -  Clear : pour effacer le dessin.
 - Figures édition
 -  Drop Style : pour appliquer le style courant (couleur de fond, couleur de trait, épaisseur et type du trait) à la ou les figures sélectionnées.
 -  Move Up : Pour déplacer une figure dans la liste des figures un cran vers l'avant.
 -  Move Top : Pour déplacer une figure dans la liste des figures un cran vers l'arrière.
 -  Move Bottom : Pour déplacer une figure dans la liste des figures devant toutes les autres figures.
 -  Move Bottom : Pour déplacer une figure dans la liste des figures derrière toutes les autres figures.
 -  Delete Selected : Pour détruire la ou les figures sélectionnées.
 - Sélection / Désélection des figures en cliquant sur les figures dans la zone de dessin.
 - Transformation des figures à la souris (click -> drag -> release)
 - Translate
 - Rotate (avec le modificateur shift)
 - Scale (avec le modificateur ctrl)
-  Filtrage des figures :
 - Suivant le type de figure
 - Suivant la couleur de remplissage actuellement sélectionnée.
 - Suivant la couleur de trait actuellement sélectionnée.
 - Suivant le type de ligne actuellement sélectionné.
 - Suivant l'épaisseur de ligne actuellement sélectionnée.

- Mise à jour de la position du curseur de la zone de dessin dans la barre d'information en bas à droite.
- Mise à jour du panneau d'information (InfoPanel.fxml / InfoPaneController.java) avec les informations d'une figure située sous le curseur de la zone de dessin (Voir Figure 5 ci-dessous qui montre le contenu du panneau d'informations lorsque le curseur se trouve au-dessus de la forme Ellipse0 de la Figure 2, page 4).

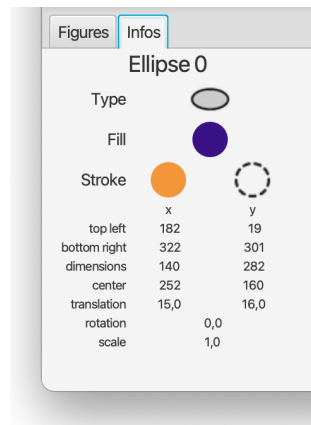


Figure 5 : Panneau d'information

Pour résumer, la classe Controller du package application contient :

- Les attributs annotés @FXML qui reflètent les éléments de la Vue utilisables dans le contrôleur. Et en particulier le Panel de la zone de dessin : Pane drawingPane.
- Les attributs propres du contrôleur :
 - Un Logger permettant de d'émettre des messages d'information ou de débogage.
 - Une instance du modèle de dessin : Drawing.
 - Une instance d'un filtre composite (contenant d'autres filtres) qui pourra être appliqué sur le modèle de dessin pour filtrer les figures
 - Une instance de HistoryManager permettant d'enregistrer les Mementos du modèle de dessin ou de mettre en place un Memento dans le modèle de dessin.
 - Des instances de différents outils :
 - currentTool qui contiendra
 - Soit l'outil de création de figures adapté au choix du type de figure à créer en mode "Création"
 - Soit l'outil de sélection / désélection de figures en mode "Edition"
 - transformTool qui contiendra l'outil permettant de transformer les figures (translation, rotation, facteur d'échelle) en mode "Edition" seulement.
 - cursorTool qui met à jour dans la barre d'information en bas de la fenêtre la position du curseur dans la zone de dessin.
- Les callbacks qui peuvent être liés aux éléments de la vue : onXXXAction(ActionEvent event).

Mise en place de JavaFX dans Eclipse

Un tutoriel complet d'installation et d'utilisation de JavaFX dans les différents IDEs tels que Eclipse, IntelliJ, NetBeans et Visual Studio Code est disponible à l'adresse : <https://openjfx.io/openjfx-docs/>. Ce qui suit ne fait que refléter ce tutoriel en utilisant les machines de l'école.

- Pour utiliser JavaFX dans Eclipse, vous aurez besoin d'une version d'Eclipse équipée du plugin E(fx)clipse, vous pourrez trouver une telle version d'Eclipse dans `/pub/ILO/JavaFX/efxclipse/eclipse`. Il vous faudra donc lancer cette version d'Eclipse et non plus la version par défaut installée sur les machines de l'école.
- Une fois cette version d'eclipse lancée, il vous faudra la configurer (Preferences → JavaFX) pour lui indiquer où se trouvent
 - JavaFX : qui se trouve dans le répertoire `/pub/ILO/JavaFX/javafx-sdk-16/lib`
 - SceneBuilder qui vous permettra d'ouvrir les fichiers FXML pour les éditer graphiquement. Le programme SceneBuilder se trouve dans de répertoire `/pub/ILO/JavaFX/SceneBuilder`
 - Ceci vous permettra de créer des projets JavaFX mais pas d'importer le projet que l'on vous fournit.
- Pour pouvoir importer et faire tourner le projet qu'on vous fournit vous allez avoir besoin :
 - De définir une « User Library » dans Eclipse correspondant à JavaFX : Eclipse → Window → Preferences → Java → Build Path → User Libraries → New et appelez la « JavaFX16 », puis éditez cette nouvelle librairie avec « Add External JARs... » et ajoutez tous les fichiers .jar contenu dans `/pub/ILO/JavaFX/javafx-sdk-16/lib`.
 - Dans le projet « Figure Editor » faites un clic droit sur la racine du projet et éditez ses « Properties » → Java Build Path et ajoutez la « User Library » que vous venez de créer au ClassPath en remplaçant au besoin celle qui s'y trouvait déjà. Votre projet est maintenant compilable, il nous reste à effectuer une dernière étape pour que vous puissiez lancer l'exécution du Main.java du package application.
 - Si vous tentez de lancer Main.java : Run As → Java Application vous allez avoir une erreur car il faut préciser à java où trouver les modules JavaFX. Pour ce faire, au lieu de Java Application, sélectionnez Run As → Run Configurations.... Double cliquez sur Java Application pour créer une nouvelle configuration que vous appellerez « Figure Editor » pour la retrouver facilement par la suite.
 - Dans cette nouvelle configuration, allez à l'onglet « Arguments » et collez les arguments suivants dans le champ « VM Arguments » :

```
--module-path ${PATH_TO_FX} --add-modules javafx.controls,javafx.fxml,javafx.graphics
```

 - La variable `${PATH_TO_FX}` doit être définie en cliquant sur le bouton “Variables” et en ajoutant une variable `PATH_TO_FX` contenant le chemin vers les librairies JavaFX : `/pub/ILO/JavaFX/javafx-sdk-16/lib`
 - Cliquez sur le bouton « run » en bas à droite et votre application devrait se lancer. Pour les lancements suivants, l'utilisation du bouton run dans la barre d'outils en haut d'Eclipse devrait suffire.

Travail à réaliser

Ce travail est à réaliser en binôme. Si vous souhaitez travailler en monôme, nous adapterons le barème en conséquence.

De manière générale les classes existantes à compléter contiennent des commentaires :

// TODO class.methode : actions à effectuer.

- Afin de vous faire découvrir le code du projet, commencez par répondre aux questions du fichier Questions.md (que vous pourrez trouver dans /pub/ILO/TPEditor/).
- Votre objectif au début du projet sera d'obtenir une première version fonctionnelle de l'application permettant de dessiner des cercles (la classe Circle existe déjà ainsi que l'outil de création de formes rectangulaires RectangulaShapeCreationTool). Pour ce faire vous devrez
 - Commencer par compléter la vue (le fichier EditorFrame.fxml).
 - Compléter la classe Figure.
 - Compléter au moins l'initialisation de la classe Controller.
 - Compléter la classe Drawing (au moins la méthode initiateFigure).
- Vous pourrez ensuite...
 - Ajouter de nouvelles classes héritant de Figure
 - Ajouter de nouveaux outils de création de figures en héritant de la classe tools.creation.AbstractCreationTool.
 - Lorsque vous aurez complété la classe Drawing, vous pourrez commencer à l'utiliser dans les méthodes onXXXAction de la classe Controller.
 - Ajouter les nouveaux outils dont vous aurez besoin.
 - Ajouter des CustomCells pour les FigureType et les LineType.
 - Ajouter de nouveaux filtres pour filtrer les figures en fonction des couleurs de remplissage et de trait, du type de trait et de l'épaisseur du trait.
 - Compléter l'HistoryManager pour pouvoir gérer les Undo/Redos à chaque action ajoutant, supprimant ou modifiant les figures.

Planning

La première séance dédiée à ce mini-projet aura lieu lundi 04/04/2022 et la dernière séance lundi 09/05/2022 pour un total de 7 séances d'1h45 s'étalant sur un peu plus d'un mois. Il vous est fortement recommandé de travailler chez vous en dehors des séances encadrées sur ce projet. Vous pourrez alors profiter des séances encadrées pour soumettre les problèmes que vous aurez rencontrés à votre chargé de TD.

Vous pourrez déposer votre projet au plus tard vendredi 13/05/2022 23:59 sur exam.ensiie.fr en utilisant le dépôt ilo-tp-editor.