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 et Polygone).
- 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'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, couleurs de remplissage ou de contour, types de traits).

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/

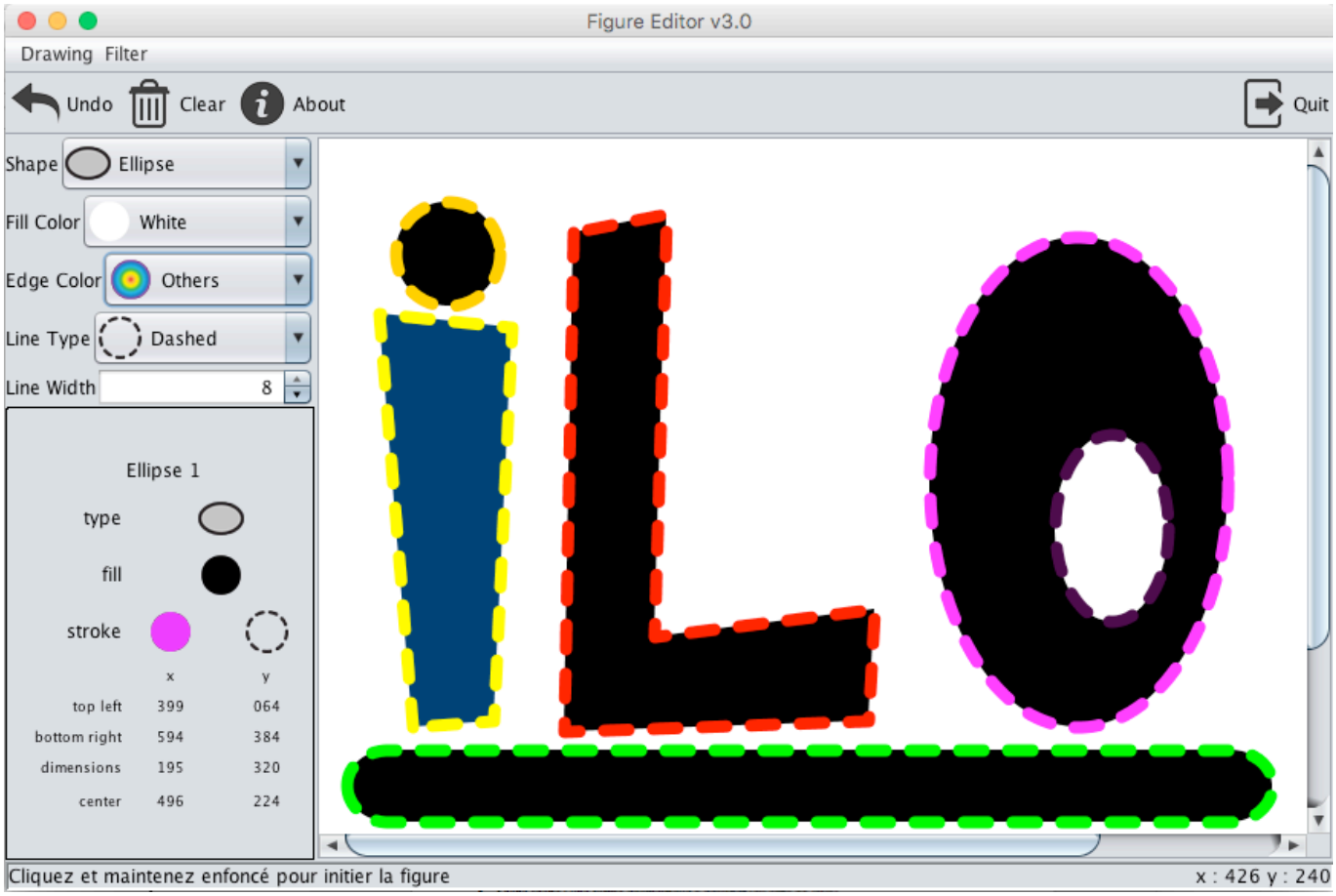


Figure 1 : Editeur de formes géométriques

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 des informations les concernant dans un panneau d'information dans l'interface graphique.

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.
- 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.
- Plus des accesseurs pour
    - Le nom de la figure : Son Type + son numéro d'instance
    - Ses bornes : le rectangle représentant les bornes de la figure
    - Le barycentre de la figure
    - Le type de la figure, son remplissage, son contour et son type de trait.

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 au 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).
- Retirer la dernière figure de la collection (action qui sera utilisée pour annuler la dernière figure)
- Supprimer l'ensemble des figures.
- Fournir un flux de figures afin qu'elle 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.

Le modèle de dessin communique suivant deux modes :

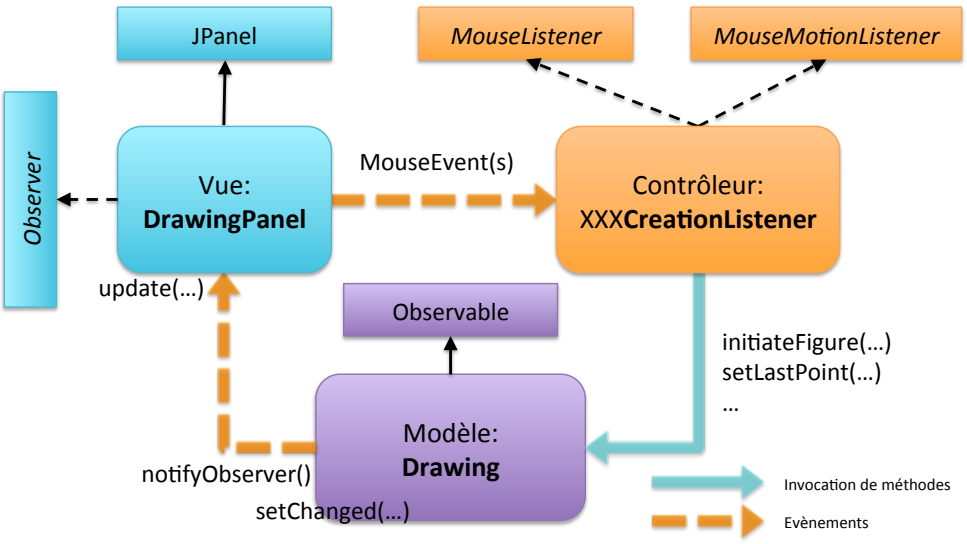
- Le modèle de dessin est piloté par les différents contrôleurs (xxxListener) associés aux divers widgets.
- Le modèle de dessin est un Observable qui met à jour son ou ses Observers lorsque son état interne change (lorsqu'une figure est ajoutée, modifiée, supprimée ou lorsque les filtres changent). Le JPanel de dessin (DrawingPanel) sera 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. Plusieurs « Actions » (à compléter) vous sont fournies et pourront être associées à vos boutons et items de menu :

- 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 type 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 AbstractCreationListener fournit un squelette de base des listeners que vous aurez à créer pour construire graphiquement les différents types de figures.
Ces différents éléments permettent de mettre en place une architecture Model / View / Controller vue en cours, pour réaliser la construction et le dessin des figures :



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 :

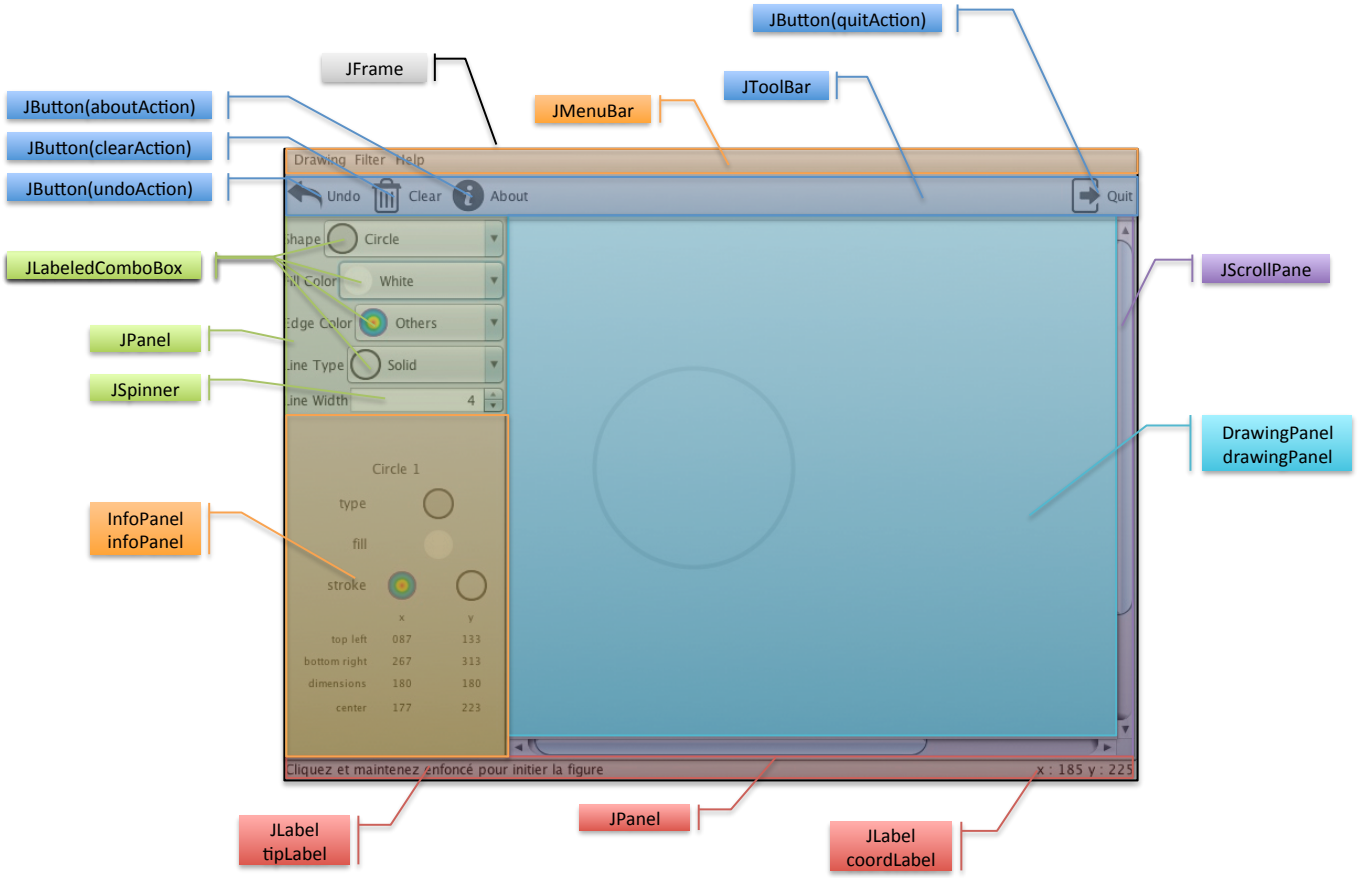
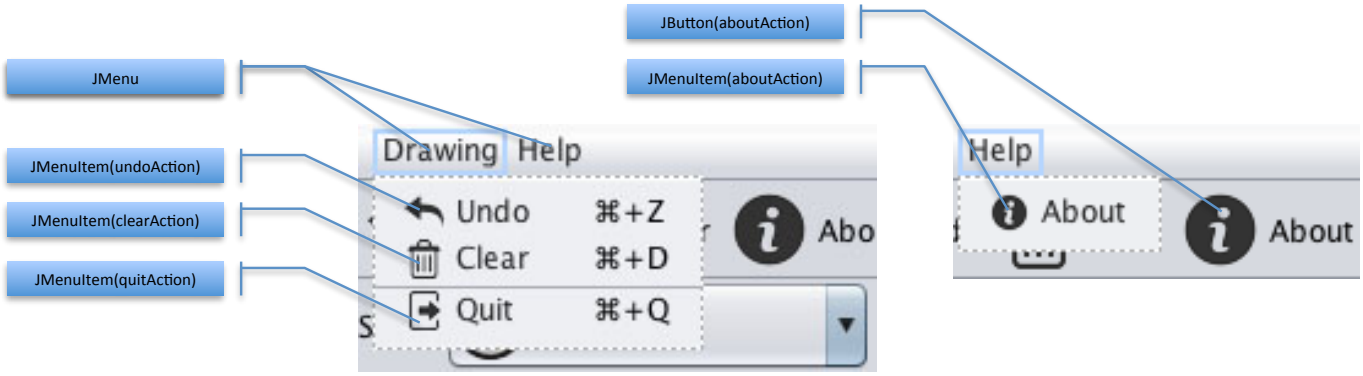


Figure 2 : Structure de l'interface graphique



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 » :

- Le menu Drawing contient les items (de type JMenuItem) suivants :
  - Undo : pour annuler la création de la dernière figure
  - Clear : pour effacer l'ensemble des figures
  - Quit : pour quitter l'application
- 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 chaque type de figure.
    - Sous menu Colors :
        - Fill Color : pour mettre en place le filtrage des figure 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 figure 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 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 :

- Undo
- Clear
- 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).
- Panneau d'information (InfoPanel)

Les JLabeledComboBox sont des combobox associés à un titre 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.

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.

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.
- Lui affecter le XXXCreationListener relatif à la figure en cours de création.
- Afficher les coordonnées du pointeur de la souris dans un label de la barre d'état.

Travail à réaliser

- Un certain nombre de classes restent à créer ou à compléter
  - utils.FlyweightFactory
  - figures.Drawing
  - widgets.DrawingPanel
  - widgets.EditorFrame : Créez l'interface graphique en éditant ce fichier avec WindowBuilder intégré à eclipse : Menu contextuel → Open With → WindowBuilder Editor
    - Complétez les différents Listeners déclarés en tant que classes internes et agissant principalement sur l'attribut drawingModel (instance de la classe Drawing).
  - widgets.InfoPanel
- Par ailleurs il vous faudra créer les classes suivantes
  - Dans le package figures (héritières de AbstractFigure)
    - Circle
    - Ellipse
    - Rectangle
    - RoundedRectangle
    - Polygon
    - Toute autre type d'héritière à Figure … Auquel cas il faudra mettre à jour l'enum enums.FigureType pour y ajouter vos nouvelles figures.
  - Dans le package figures.creationListeners (héritières de AbstractCreationListener)
    - Tout les listeners dont vous aurez besoin pour créer graphiquement les figures ci-dessus.

Note : Pour pouvoir utiliser le WindowBuilder d'Eclipse dans un projet utilisant Java8 sur les machines de l'école, il faut explicitement lancer eclipse en lui indiquant en argument la machine virtuelle (VM) Java 8 depuis un terminal :

```
> eclipse –vm /usr/lib/jvm/jdk-8-oracle-x64/bin/java
```