

Swing

TD n°9

- TP noté, à réaliser seul, ou en binôme.
- Poursuite du TD n°8
- Implémentation de nouvelles fonctionnalités, de façon libre
- TD terminé à envoyer avant le 26 mai 12h00 sur <http://download.kobject.net>

Idées de fonctionnalités à ajouter :

- Autres outils : Texte, image, figures (segment, polygone)...
- Figures avec contour (couleur) et remplissage (couleur, motif)
- Gestion de calques de dessin
- Changement du pointeur de souris en fonction de l'outil sélectionné
- Annuler/refaire la dernière figure
- Défilement vertical ou horizontal de la zone de dessin (avec JScrollPane)

Correction de Bug

Symptôme : Sélection non apparente sur création Rectangle ou Ellipse

1 ligne à supprimer dans le constructeur :

```
public EcouteurSelection(MainController mainController) {
    super(mainController);
    this.clip = new Rectangle();
    borderColor = new Color(79, 129, 189);
    backgroundColor = new Color(205, 220, 237, 100);
}
```

A ajouter dans le changement d'outil courant :

```
@Override
public void propertyChange(PropertyChangeEvent evt) {
    ...
    case ELLIPSE_TOOL:
        changerOutil(eEllipse);
        zoneDeDessin.seteSelection(eEllipse);
        break;
    ...
}
```

TD n°8

- TP noté, à réaliser seul, ou en binôme.
- Projet initial à importer dans Eclipse : [versioninitd8.zip](#)
- Renommer le projet à votre nom (ou aux noms du binôme)
- TD terminé à envoyer en fin de séance sur <http://downupload.kobject.net>

Question 1

L'application de dessin ne propose pour l'instant que 2 outils de dessin (crayon et étoile). On souhaite ajouter 2 nouveaux outils, permettant de dessiner des figures géométriques simples : Rectangle et Ellipse.

De façon à faciliter cette implémentation, 1 nouveau contrôleur a été créé : **EcouteurSelection**, permettant de représenter à la souris la création d'un Rectangle ou d'une Ellipse.

- Modifier le modèle pour intégrer les figures de type **Rectangle** et **Ellipse**
- Créer les contrôleurs **EcouteurRectangle** et **EcouteurEllipse**, héritant du contrôleur **EcouteurSelection** permettant la création des figures correspondantes
- Ajouter l'élément de menu **Figure** dans le menu outil, disposant de 2 JMenuItem : Rectangle et Ellipse
- Modifier le contrôleur principal **MainController** en conséquence, pour permettre l'appel de ces contrôleurs
- Modifier la vue **ZoneDeDessin** pour intégrer le dessin des nouvelles figures

Question 2

L'application développée jusque là ne permet pas de choisir la couleur. On souhaite permettre à l'utilisateur de modifier les caractéristiques du pinceau avec lequel se fait le dessin en ajoutant ce choix dans la boîte de dialogue existante **PenSizeAndColorDialog**.

Le tutoriel de Java (<http://docs.oracle.com/javase/tutorial/uiswing/components/colorchooser.html>) donne la marche à suivre et fournit des exemples de choix de couleur.

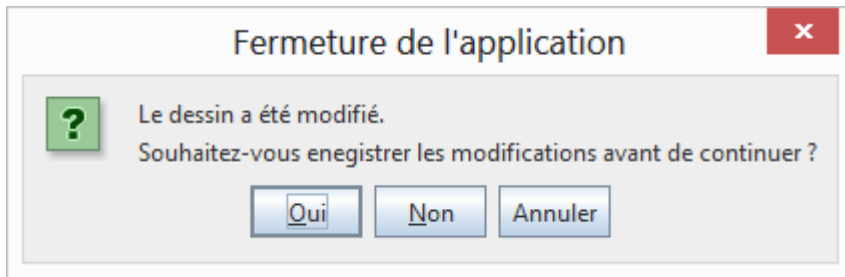
- Modifier le modèle pour que la couleur soit mémorisée (une seule couleur par figure)
- Modifier la vue **PenSizeAndColorDialog** pour qu'elle intègre le choix de la couleur
- Modifier le contrôleur principal **MainController** pour qu'il effectue la modification de couleur sur le modèle
- Modifier la vue **ZoneDeDessin** pour que le dessin se fasse en utilisant les couleurs voulues
- Afficher la couleur choisie dans la barre d'état (sous forme d'un triplet de valeurs entières indiquant les niveaux de rouge, vert et bleu).

TD n°7

TODO

Partie 1 : prévention de la perte potentielle du dessin effectué

Sur fermeture de la fenêtre principale, ou l'ouverture d'un Dessin, si le dessin en cours a été modifié (**dirty=true** dans MainController), la boîte de message suivante doit-être affichée :

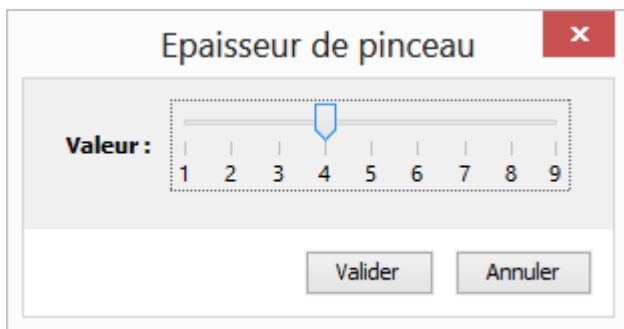


1. créer la méthode **closeQuery(String message)** dans **MainController** permettant de gérer l'enregistrement éventuel avant perte du dessin
2. Gérer les événements pouvant provoquer une perte du dessin en créant les écouteurs nécessaires.

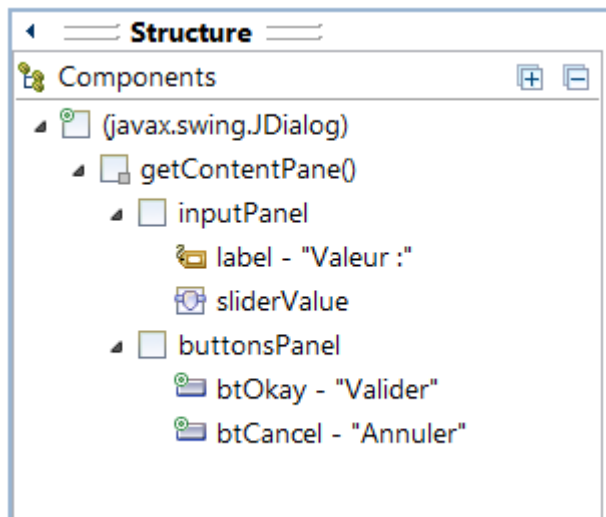
Partie 2 : Ajout de l'outil Épaisseur de trait

Ajouter la fonctionnalité permettant de modifier l'épaisseur du trait de dessin, accessible depuis le menu **Outil** :

- **1 - Boîte de dialogue JDialog:**
 - **a - aspect visuel :**



- Eléments :



- contentPane → [new BorderLayout(0, 0)]
- inputPanel → North JPanel [new FlowLayout(FlowLayout.CENTER, 10, 10)]
 - JLabel + JSlider → setMinorTickSpacing, setMajorTickSpacing, setPaintTicks, setPaintLabels, setFont(new Font("Segoe UI", Font.PLAIN, 11))
- buttonsPanel → South JPanel [new FlowLayout(FlowLayout.RIGHT, 10, 10)]

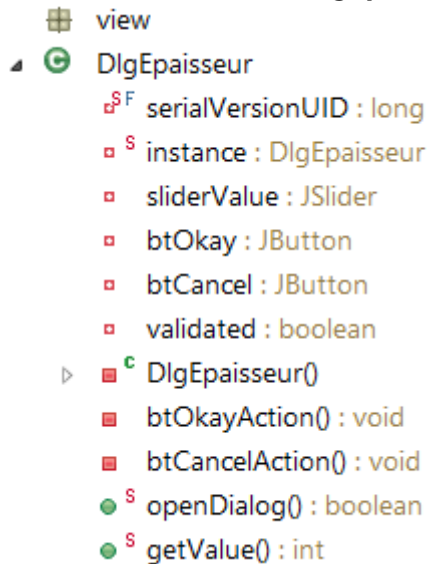
- **b - Comportement :**

Le comportement du dialog (essentiellement visuel) peut être implémenté directement dans la vue (sans

séparation MVC) :

- Implémenter en utilisant le design pattern Singleton
- Dialog à rendre modal
- Définir un **WindowListener** pour associer la fermeture au bouton annuler
- Ajouter des **ActionListener** sur les boutons
- **getValue()** retourne l'épaisseur choisie
- **openDialog()** ouvre le dialog et retourne vrai si le choix est validé par Okay

Structure de la classe DlgEpaisseur :



- **2 - Modification du modèle :**
 - Modifier le modèle pour intégrer la propriété épaisseur du trait
- **3 - Modification du contrôle :**
 - Ajouter un élément de menu dans **Outil**
 - Modifier les contrôleurs pour intégrer la modification potentielle de l'épaisseur

Ressources

1 - Boîtes de message :

on utilise la classe [JOptionPane](#)

```

public class DialogsEx {
    public static void main(String[] args) {
        JFrame maFenetre = new JFrame("JOptionPane exemples");
        UIManager.put("OptionPane.messageFont", new FontUIResource(new Font("Segoe UI", Font.PLAIN, 12)));
        UIManager.put("OptionPane.buttonFont", new FontUIResource(new Font("Segoe UI", Font.PLAIN, 12)));
        maFenetre.setBounds(50, 100, 700, 500);
        maFenetre.setResizable(true);
        maFenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        maFenetre.setVisible(true);
        JOptionPane.showMessageDialog(maFenetre, "L'enregistrement du fichier est effectué.", "Dialogue d'information", JOptionPane.INFORMATION_MESSAGE);
        JOptionPane.showMessageDialog(maFenetre, "Une erreur est survenue lors de l'enregistrement du fichier.\nVous pouvez réessayer avec un autre nom de fichier ou sur un autre support !", "Dialogue d'avertissement", JOptionPane.WARNING_MESSAGE);
    }
}
  
```

```
JOptionPane.showMessageDialog(maFenetre, "Une erreur est survenue et
l'opération a échoué !", "Dialogue d'erreur", JOptionPane.ERROR_MESSAGE);
    if (JOptionPane.showConfirmDialog(maFenetre, "Voulez-vous poursuivre ?",
"Dialogue de questionnement", JOptionPane.YES_NO_CANCEL_OPTION) ==
JOptionPane.YES_OPTION) {
        String prenom = JOptionPane.showInputDialog(maFenetre, "Prénom :",
"Pierre");
        if (prenom != null)
            JOptionPane.showMessageDialog(maFenetre, "Bonjour " + prenom + ".",
"Bonjour", JOptionPane.PLAIN_MESSAGE);
    }
}
```

2 - Ecouteurs sur fenêtre : Pour gérer les événements liés à la fenêtre (Fermeture, désactivation...) :

1. [WindowAdapter](#)
2. [WindowListener](#)

TD6

- [TD6 initial avec ActionListeners](#)
- [TD6 initial avec propertyChangeListener](#)

TODO

Partie 1

Dans chaque binôme :

1. 1 étudiant implémente les actions suivantes en utilisant des **ActionListener** dans la vue FenetreDeDessin dans le projet [TD6-ACTL](#)
 - changement d'outil sur menu (Etoile ou crayon)
 - Fermeture de l'application
 - Effacement du dessin
2. 1 autre implémente ces 3 mêmes actions en utilisant **PropertyChangeListener** au niveau du contrôleur dans le projet [TD6-PCL](#)
3. Comparer ensuite les 2 versions (avantages/inconvénients)

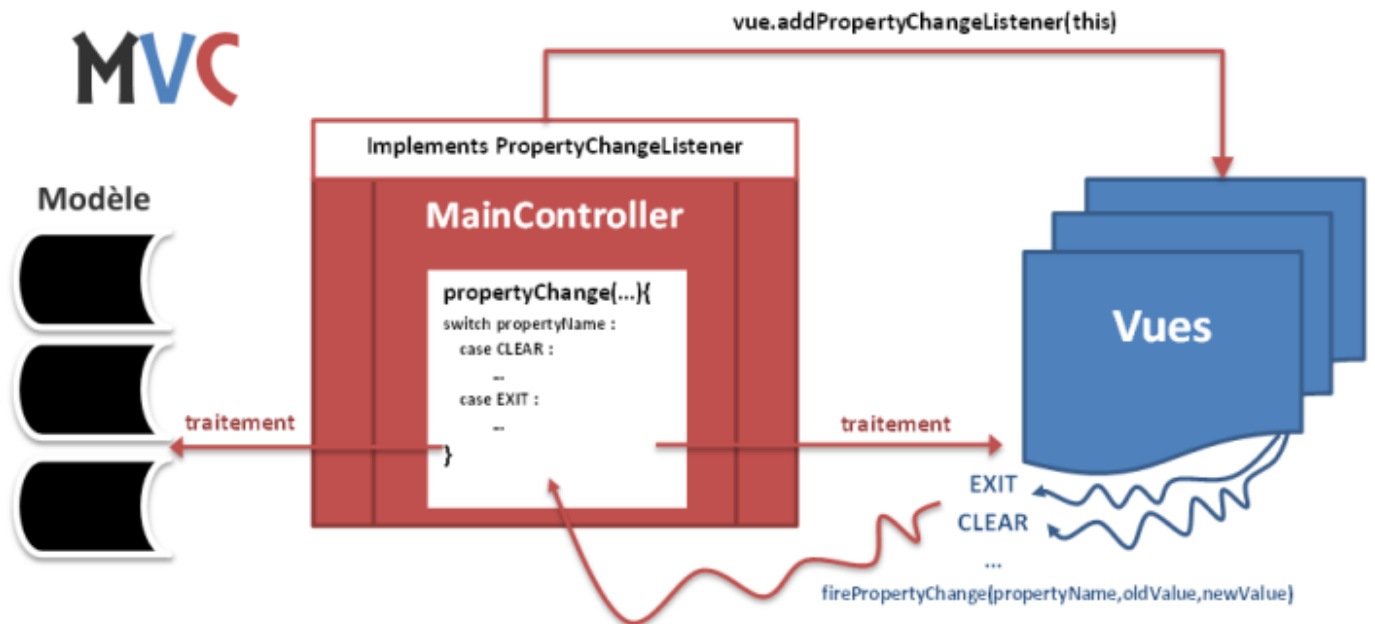
Partie 2

1. Implémenter les actions des éléments de menu **Ouvrir**, **Enregistrer** et **Enregistrer sous** :
 - observer et comprendre la mise en oeuvre de la sérialisation proposée (Interface, classe DAO, classes métier)
 - utiliser l'instance **dessinDao** du contrôleur pour l'ouverture et la sauvegarde, et la propriété **dirty** (drapeau pour désigner la modification)
 - utiliser **JFileChooser** pour les boîtes de dialogue (voir **FileDialogEx.java**) voir [JFileChooser javadoc](#)
 - Gérer les exceptions

Partie 3

1. Créer une boîte de dialogue **A propos de...** et ajouter un élément de menu correspondant dans un menu ?/A propos de... voir **JDialog**
2. Proposer des solutions pour qu'une seule instance de **DialogAbout** soit créée pendant l'exécution de l'application
3. Créer un sous menu **Couleur** dans Outil, proposant de changer la couleur de l'outil (noir, bleu, rouge, vert, jaune), modifier le modèle pour prendre en compte la couleur des figures.

Boîte de dialogue A propos de... :



Ressources

- Charger une image contenue dans les sources java :

```
String imagePathAndFileName="CheminRelatifVersImage";
java.net.URL imageURL = getClass().getResource(imagePathAndFileName);
```

```
img = new ImageIcon(imageURL).getImage();
```

- Dessiner sur un JPanel :

```
JPanel panel = new JPanel() {  
    @Override  
    public void paint(Graphics g) {  
        //dessin sur g...  
    }  
};
```

- L'interface [PropertyChangeListener](#)
- l'EDT Swing : [Event Dispatching Thread](#)

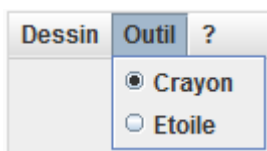
TD5

- [TD5 projet initial](#)

TODO

1. Terminer l'implémentation Dessin Swing avec 2 outils + Contrôleur principal
2. Ajouter le changement d'outil sur frappe touche du clavier (e → Etoile , c → Crayon) voir **addKeyListener**
3. Ajouter le changement d'outil dans un menu **Outil** composé de 2 **JRadioButtonMenuItem**, intégrés dans un **ButtonGroup** :

Menu Outil :



TD4

- [Sujet](#)
- [applet V1](#)
- [Applet V1 avec contrôleur principal](#)

Composants

JFrame

[java 7 JFrame API](#)

Paramètres de base

- setTitle(String)

- setSize(int, int)
- setDefaultCloseOperation(int)
- setLocationRelativeTo(Component)
- setJMenuBar(JMenu)
- setLayout(LayoutManager)
- add(Component, Object)
- setVisible(Boolean)

Ajout de listeners :

- addMouseListener(MouseListener l)
- addMouseMotionListener(MouseMotionListener l)
- addKeyListener(KeyListener l)

JMenuBar

[Java 7 JMenuBar API](#)

- add(JMenu)

JMenu

- add(JMenuItem)
- addSeparator()

JMenuItem

- addActionListener(ActionListener)

LayoutManager

[Java 7 LayoutManager API](#)

BorderLayout

```
parent.setLayout(new BorderLayout());  
parent.add(component, BorderLayout.CENTER);
```

GridLayout

```
parent.setLayout(new GridLayout(1, 3));  
parent.add(component);
```

From:
<http://slamwiki2.kobject.net/> - **SlamWiki 2.1**

Permanent link:
<http://slamwiki2.kobject.net/slam4/gui/swing?rev=1400567829>

Last update: **2019/08/31 14:39**

