Université Paris Diderot **Programmation Orientée Objet**
L2 **Année 2016-2017**

TP - Séance n°9

Interfaces graphiques

Certains serveurs d'email rejettent les messages envoyés depuis DidEL. Il serait donc bien de regarder régulièrement les annonces sur DidEL.

Pour faire ce TP, vous ne devez pas hésiter à consulter fréquemment la documentation Java :

— `https://docs.oracle.com/javase/8/docs/api/`

— `http://docs.oracle.com/javase/tutorial/uiswing/index.html`

pour savoir comment utiliser les méthodes des API graphiques.

Le but de ce TP est de faire une interface graphique, semblable à celle de la figure 1, permettant de visualiser une couleur à partir des ses composantes RVB (rouge, vert, bleu). Pour que les concepts mis en jeu apparaissent clairement, **on essayera de respecter l'architecture dite *modèle-vue-contrôleur***, qui consiste à séparer :

— le modèle de données,

— la vue utilisateur, qui présente les données du modèle à l'utilisateur,

— le contrôleur, qui, en fonction des événements qu'il reçoit, modifie les données du modèle et synchronise la vue.

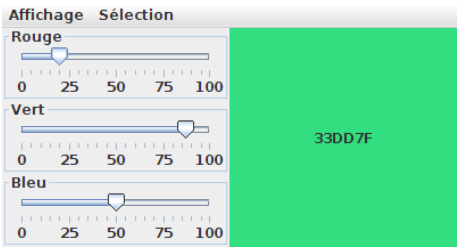

FIGURE 1 – Interface à engendrer

Étape 1 On s'intéresse d'abord à la vue. Dans une classe `Palette`, on définit une classe interne `Vue` qui étendra `JFrame`. Il s'agit d'afficher une fenêtre divisée en deux parties égales : le panneau de choix et le panneau de couleur . Pour l'instant, les deux panneaux afficheront deux couleurs différentes choisies par le programmeur (comme sur la figure 2).

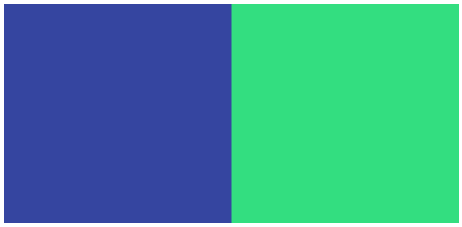

FIGURE 2 – Étape 1

Pour cela, dans le constructeur de `Vue`, on donnera au conteneur principal de notre vue (un `JPanel` obtenu par `getContentPane()`) une mise en page de grille (`GridLayout`) (utilisez la méthode `setLayout(LayoutManager mgr)` du `JPanel`). Ce conteneur principal contiendra deux panneaux, celui de gauche appelé `panneauChoix` (qui contiendra plus tard les curseurs de choix) et l'autre `panneauColore`. Pour ajouter un panneau au conteneur, utilisez la méthode `add(Component comp)`.

Les couleurs des panneaux seront affectées avec la méthode `setBackground(Color c)` et la couleur sera engendrée par le constructeur de `Color` qui prend des flottants. On pensera à écrire `0.5f` pour indiquer que 0.5 est un flottant et non pas un double.

Enfin, on rappelle que pour afficher l'interface graphique, on aura besoin (par exemple dans le constructeur de `Palette`, qui sera indirectement appelé par `main`) du code suivant :

```
vue.pack(); // pour mettre en place la vue
vue.setVisible(true); // pour la rendre visible
```

où `vue` est l'instance de `Vue`. Afin d'éviter d'éventuels problèmes d'affichage reliés à des interférences entre plusieurs threads, on utilisera dans `main` l'invocation "magique" à travers `SwingUtilities.invokeLater()` (comme vu en cours). Donc votre `main` devrait ressembler à ceci :

```
public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            Palette palette = new Palette();
        }
    });
}
```

Important : *Pensez à faire similairement pour votre projet ! Sinon, un projet qui semble fonctionner dans une configuration donnée pourrait soudainement afficher une fenêtre grise le jour de la soutenance !*

Étape 2 La deuxième étape consiste à ajouter les curseurs dans le panneau de gauche, sans se préoccuper de l'écouteur correspondant.

Pour cela, on choisira de nouveau une mise en page de grille pour `panneauChoix`.

Les curseurs seront fabriqués avec la classe `JSlider`. Il peut être utile d'utiliser un tableau de `JSlider` de dimension 3. Lire la documentation pour trouver comment indiquer les graduations et les nombres correspondants. Pour écrire le nom de chaque curseur, on utilisera :
`unCurseur.setBorder(BorderFactory.createTitledBorder("Rouge"));`

Étape 3 On s'intéresse maintenant au modèle. On définit simplement une classe interne `Modele` qui aura trois champs *entiers* correspondant aux valeurs de rouge, de vert et de bleu ; ainsi que les accesseurs et les modifieurs de ces champs.

Étape 4 Dans la classe `Vue`, on définit une méthode `miseAJour()` qui synchronise le panneau `panneauColore` avec le modèle (il s'agit de récupérer la couleur stockée dans le modèle et de changer la couleur de `panneauColore` en conséquence). Pour l'instant on ne s'occupe pas des valeurs des curseurs ! Quels sont les objets de votre interface qui doivent être déclarés comme attributs de la classe `Vue` ?

Étape 5 Le contrôleur est défini dans une classe interne nommée `Controleur` qui implémente l'interface `ChangeListener`. Cette interface déclare la méthode abstraite `void stateChanged(ChangeEvent event)` (qui sera exécutée lorsqu'on déplace les curseurs) qu'on doit donc définir dans `Controleur`. Cette méthode doit :

1. récupérer les valeurs des curseurs,
2. modifier la couleur stockée dans le modèle,
3. exécuter la méthode `miseAJour()` qui synchronise la vue avec le modèle.

N'oubliez pas, dans la vue, de rendre écoutable chaque curseur avec la méthode `addChangeListener`.

Étape 6 On va rajouter quelques fonctionnalités à cette interface graphique. On définit dans notre vue une barre de menus, et un menu `Sélection` qui contiendra deux items, `Gris50` et `Complémentaire`.

— Quand on clique sur `Gris50`, le contrôleur doit stocker dans le modèle un gris moyen (rouge à 50%, vert à 50%, bleu à 50%), avertir la vue qu'elle doit *déplacer les curseurs en conséquence* et synchroniser le panneau de couleur.

— Similaire pour `Complémentaire`, sauf que la couleur doit maintenant être remplacée par la *couleur complémentaire* de la couleur courante (ex. si la valeur de rouge est à $X\%$ dans la couleur courante, la valeur de rouge est à $100 - X\%$ dans la couleur complémentaire).

Quelques remarques :

— En plus de `ChangeListener`, le contrôleur doit maintenant implémenter l'interface `ActionListener` pour écouter les boutons des menus.
On doit alors rajouter une méthode `void actionPerformed(ActionEvent event)`.

— N'oubliez pas, dans la vue, de rendre écoutables les boutons avec la méthode `addActionListener` et de définir, avec `setActionCommand`, la commande envoyée.
— En plus de `miseAJour()` qui synchronise le panneau de couleur avec le modèle, la vue doit contenir une méthode `miseAJourCurseurs()` qui déplace les curseurs pour les synchroniser avec le modèle.
— Il est possible qu'il y ait conflit entre `stateChanged` (qui va mettre à jour le modèle en fonction des curseurs) et `miseAJourCurseurs` : en effet, lorsque les curseurs sont automatiquement déplacés, ils enverront a priori des événements et la méthode `stateChanged` sera exécutée. Pour éviter cela, on peut rajouter à la classe `Controleur` un booléen qui sera temporairement modifié quand la mise à jour des curseurs sera effectuée, et que `stateChanged` testera pour éviter d'exécuter quoi que ce soit en même temps.

Étape 7 Rajoutez un menu `Affichage`, qui permettra de basculer (pensez à utiliser des "boutons radio" comme sur la figure 3) entre une sélection de couleur et une sélection de niveau de gris (un seul curseur). Quand on passe d'un mode de sélection à un autre, le panneau de contrôle est redessiné (notez que lorsqu'on enlève/rajoute un composant d'un panneau avec les méthodes `remove(Component comp)`, `add(Component comp)`, on doit remettre en place ledit panneau en utilisant la méthode `validate()`). On peut vouloir garder le gris sélectionné quand on passe du mode Gris au mode RVB, et convetir en gris (moyenne des trois composantes) la couleur sélectionnée quand on passe du mode RVB au mode Gris.

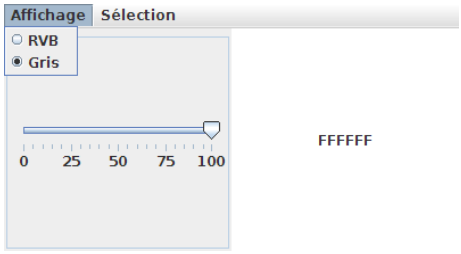

FIGURE 3 – Le menu Affichage avec deux boutons radio

Étape 8 S'il vous reste du temps, vous pouvez indiquer dans le panneau de couleur le *code hexadécimal* (obtenu en convertissant la valeur de chaque composante en un nombre hexadécimal compris entre `00` et `FF` (255 en décimal) ; par exemple un vert pur est codé par `00FF00`) de la couleur sélectionnée. Pensez à le mettre à jour en même temps que le reste du panneau de couleur !