Université de Paris **POOIG**
L2 **Année 2020-2021**

TP - Séance n°9

Interfaces graphiques 1

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, nous utiliserons l'architecture dite *modèle-vue-contrôleur*, qui consiste à repartir les tâches du programme sur trois classes :

— **Modele** : le modèle de données,

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

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

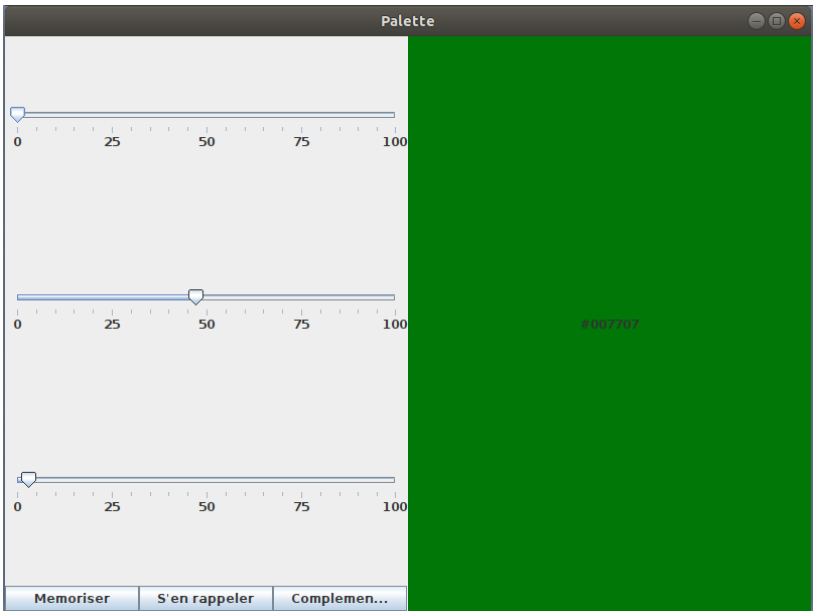

FIGURE 1 – Interface à engendrer

Affichage d'une fenêtre

Exercice 1 La classe Palette. La classe `Palette` contiendra la méthode `main` et servira à mettre en relation la vue, le modèle et le contrôleur. Créez une classe `Palette` qui contient le code suivant :

```
public class Palette {
    private Vue view;

    public Palette() {
        view = new Vue();
        view.setVisible(true);
    }

    public static void main(String[] args) {
        new Palette();
    }
}
```

Exercice 2 La classe Vue. En Java, une fenêtre est représentée par un objet de type `JFrame`.

1. Créez une classe `Vue` qui hérite de `JFrame`[1]. Ajoutez deux lignes d'import : `import javax.swing.*;` et `import java.awt.*;`
2. Dans la classe `Vue`, écrivez un constructeur sans paramètre, qui change le titre de la fenêtre (= `this`) en "Palette". Utilisez la méthode `void setTitle(String title)` de la classe `JFrame`, héritée de `Frame`[2].
3. Ajoutez une ligne au constructeur pourque la fenêtre (`this`) ait une largeur de 800 pixels et une hauteur de 600 pixels. Utilisez la méthode `void setSize(int width, int height)` de la classe `JFrame`, héritée de `Component`[3].
4. Nous souhaitons que l'application se termine lorsque l'utilisateur ferme la fenêtre. Pour faire cela, ajoutez la ligne `setDefaultCloseOperation(EXIT_ON_CLOSE);` au constructeur de `Vue`.
5. Compilez et exécutez `Palette` : votre application devrait afficher une fenêtre vide de 800 x 600 pixels avec titre "Palette". Vérifiez que l'application se termine lorsque vous fermez la fenêtre.

1. https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/javax/swing/JFrame.html
2. https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/java/awt/Frame.html
3. https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/java/awt/Component.html

Ajout du panneau coloré

Pour remplir la fenêtre, il faut y ajouter des objets de type `JComponent`[4]. Toute `JFrame` possède un `contentPane`, qui rassemble les différents éléments à afficher. Pour afficher quelque chose dans une fenêtre il faut donc :

— créer un objet de type `JComponent` à afficher ;

— récupérer le `contentPane` de la fenêtre grâce à la méthode `Container getContentPane()` ;

— ajouter l'objet de type `JComponent` au `contentPane`, avec la méthode `add`.

Exercice 3 Le panneau coloré. Dans cet exercice vous ajouterez à la classe `Vue` un panneau coloré, qui sera vert au début. Un panneau est un objet de type `JPanel`, une classe qui hérite de `JComponent`.

1. Ajoutez à `Vue` un attribut `panneauColore` de type `JPanel` et initialisez-le avec `new JPanel()`.
2. Dans le constructeur de `Vue`, définissez la couleur initiale (vert) de `panneauColore`. Une couleur en Java est un objet de type `Color`, qui possède un constructeur `Color(int red, int green, int blue)`. Un `JPanel` possède une méthode `void setBackground(Color c)`.
3. Ajoutez le `panneauColore` au `contentPane` de la fenêtre, avec la ligne : `this.getContentPane().add(panneauColore);`
4. Testez. Le panneau coloré remplira la fenêtre entière ; nous verrons plus tard dans ce TP comment mieux placer les composants.

Exercice 4 Nous voulons afficher le mot "Vert" au `panneauColore`. On utilisera pour cela un objet de type `JLabel`, doté d'une méthode `setText`. Ajoutez un attribut `JLabel etiqCouleur` à la classe `Vue`, mettez le mot "Vert" dans l'étiquette, et ajoutez-le au panneau coloré.

Attention : on ne veut pas ajouter l'étiquette à la fenêtre, mais au panneau ! Utilisez donc la méthode `add` du `JPanel panneauColore`, et pas celle du `contentPane`.

Placement des composants et panneau choix

Lorsqu'un `JPanel` contient plusieurs éléments on veut contrôler la manière dont ils seront disposés dans la fenêtre. Tout `JPanel`, ainsi que le `contentPane` de la fenêtre, dispose d'un "layout", qui peut être modifié avec `setLayout`.

Exercice 5 En utilisant le layout `BorderLayout`, faites en sorte que le mot "Vert" apparaisse au centre de la fenêtre :

1. Utilisez la méthode `setLayout` pour donner un nouveau `BorderLayout` au `panneauColore`.

4. https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/javax/swing/JComponent.html

2. La méthode `add` de `JPanel` admet un deuxième argument optionel qui spécifie le placement de l'élément dans le panneau. Ajoutez `BorderLayout.CENTER` comme deuxième argument à la ligne qui ajoute l'étiquette au panneau.
3. Modifiez l'alignement horizontale de l'étiquette : utilisez la méthode `setHorizontalAlignment` de la classe `JLabel`.

Exercice 6 Le panneau choix.

1. Ajoutez un `JPanel panneauChoix` (vide pour le moment) à la fenêtre créée dans `Vue`.
2. Changez le layout du `contentPane` en `GridLayout` [5], et affichez côte à côte les deux `JPanel` différents dans votre fenêtre.
3. Au panneau choix, ajoutez des `JButton` et des `JSlider` à l'un des panneaux pour obtenir un affichage similaire à la Figure 1. On veut avoir trois curseurs étiquetés "Rouge", "Vert" et "Bleu" et gradués de 0 à 100, ainsi que trois boutons "Mémoriser", "S'en rappeler" et "Complémentaire". Lisez les documentations correspondantes pour savoir comment configurer l'apparence de chacun des composants [6]. Les composants ne seront pas encore fonctionnels, cet exercice ne vous demande que de les afficher.

Modèle

Exercice 7 Créez une nouvelle classe `Modele`, qui servira à décrire l'état du système à un instant donné. Pour le moment `Modele` doit seulement représenter une couleur codée par trois valeurs (niveaux de rouge, vert et bleu).

Exercice 8 Dans la classe `Vue`, ajoutez un champ contenant un modèle ainsi qu'une méthode `miseAJour()` qui récupère la couleur du modèle et modifie le panneau coloré en conséquence. Modifiez `Palette` pour associer un modèle à la vue affichée.

(Bonus) Faites en sorte que `miseAJour()` affiche le codage hexadécimal de la couleur utilisée au centre du panneau coloré.

Interactions modèle-vue

Pour le moment le modèle peut agir sur la vue via la méthode `miseAJour` mais l'inverse n'est pas vrai. On veut pouvoir modifier le modèle en déplaçant les curseurs présents dans la vue.

Exercice 9 Définissez une classe `Controleur` avec deux champs correspondant à un modèle et à une vue, et ajoutez un champ `Controleur` à la classe `Vue`. Modifiez `Palette` pour y créer un triplet modèle-vue-contrôleur.

5. `https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/java/awt/GridLayout.html`
6. `https://docs.oracle.com/javase/tutorial/uiswing/components/button.html` et `https://docs.oracle.com/javase/tutorial/uiswing/components/slider.html`

Exercice 10 Définissez une méthode `sliderMoved` dans le contrôleur qui met à jour le modèle et la vue en accord avec la position des curseurs. On utilisera la méthode `getValue()` des curseurs pour obtenir leur valeur courante.

Exercice 11 Pour provoquer un effet lorsque l'on déplace un curseur, un `JSlider` dispose d'une méthode `addChangeListener` :

```
curseur.addChangeListener((event) -> { ... });
```

La méthode passée en paramètre sera appelée dès que `curseur` est modifié. Faites en sorte que le modèle et la vue soient mis à jour lorsque l'on déplace un curseur.

Exercice 12 Faites fonctionner les trois boutons "Mémoriser", "S'en rappeler" et "Complémentaire". Le premier doit mettre la couleur courante en mémoire, le second doit remplacer la couleur courante par la dernière couleur mémorisée (et donc changer le modèle, le panneau coloré et les positions des curseurs), et le troisième doit remplacer la couleur courante par sa couleur complémentaire.

Pour déclencher un effet lorsqu'un bouton est pressé on procède comme avec les curseurs, en utilisant la méthode `addActionListener`. Pour déplacer les curseurs on dispose de la méthode `setValue`.