Travaux Dirigés n°2

Java Avancé

—M1—

Modèle – Vue – Contrôleur (MVC)

Conception object, héritage, interfaces graphiques, délégation, polymorphisme

---

— Pour ce TD, travaillez dans le projet `fr.dauphine.ja.`*`nomprenom`*`.shapes` du TD précédent.
— **Attention :** vous devez structurer correctement le projet *shapes*. À la fin de ce TD, vous devez avoir un package `view`, un package `model` et un package `controller`. À vous de répartir les classes au mieu dans ces packages.
— N'oubliez pas de commiter régulièrement et de synchroniser votre dépot local avec votre dépot GitHub en utilisant git push !

▸ Exercice 1. Graphiques 2D

On souhaite utiliser les fonctionnalités du SDK Java pour dessiner des formes à l'écran. Pour cela on utilisera les classes `Graphics` et `Graphics2D` du package `java.awt` et les classes `JFrame et JPanel` du package `javax.swing`. Dans ce premier exercice vous allez vous familiariser avec les différentes primitives de dessin disponibles dans le SDK.

1. Lisez la documentation et déterminez la fonction de chacune de ces classes ainsi que les éventuels liens qui les connectent.
2. Créez un nouveau package `view` dans le projet *shapes* et créez une classe `MyDisplay` qui hérite de la classe `JPanel`.
3. Ajoutez une methode `main(...)` à la classe `MyDisplay` avec le contenu suivant.

```
1 public static void main(String []args){
2     JFrame frame = new JFrame("Java Avancé - Graphic Display");
3     frame.setSize(new Dimension(500,500));
4     frame.setVisible(true);
5     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
6
7     MyDisplay d = new MyDisplay();
8     frame.add(d);
9 }
```

   Trouvez dans la documentation la fonction de chacune de ces instructions.
4. Redéfinissez la méthode `paintComponent()` de la classe mère `JPanel` et utilisez la méthode `drawLine(...)` de la classe `java.awt.Graphics` pour dessiner une ligne à l'écran.

5. La méthode `paintComponent()` est appelée lors du premier affichage du `JPanel`, à quel autre moment cette méthode est-elle appelée ?
6. Quelle différence fondamentale y a t'il entre le package `java.awt` et le package `javax.swing`.

▸ Exercice 2. Modèle et vue

Dans cet exercice, vous allez modifier le package `view` pour en faire un package de visualisation pour les formes que vous avez faites lors du dernier TD.

Attention : Les classes qui serviront à représenter les formes géométriques (partie Modèle) doivent aller dans le package `model` alors que les classes qui servent à gérer l'affichage des formes à l'écran, doivent aller dans le package `view`. Prenez le temps d'organiser votre projet correctement.

1. Concevez une architecture objet en utilisant les différentes techniques vues en cours (héritage, délégation, interfaces, polymorphisme ...).
2. Dessinez le diagramme des classes de votre architecture. Vous pouvez vous inspirer de la représentation UML [(1)].
3. Faites la liste des changements nécessaires pour ajouter une nouvelle forme. Discutez et défendez votre architecture objet avec votre binôme et avec le reste de la classe. Si nécessaire, révisez l'architecture.
4. Implémentez et testez votre architecture.

▸ Exercice 3. Contrôleur

On souhaite maintenant ajouter un contrôleur pour modifier le modèle.

1. Ajoutez un package `controller`.
2. L'interface `java.awt.event.MouseMotionListener` disponible dans le SDK Java permet de réagir à un mouvement de la souris. Lisez la doc et comprenez son fonctionnement.
3. Ajoutez une classe `MouseController` qui implémente l'interface `java.awt.event.MouseMotionListener`.
4. Implémentez la méthode `mouseMoved(...)` de l'interface `MouseMotionListener` pour déplacer une forme du modèle lorsque que le pointeur se déplace.
5. (Optionnel) modifiez votre contrôleur et faites en sorte qu'il soit possible de choisir la forme qui sera déplacée avec la souris.

(1). https://fr.wikipedia.org/wiki/Diagramme_de_classes