TP6

MVC :

REALISATION D'UN FEU TRICOLORE

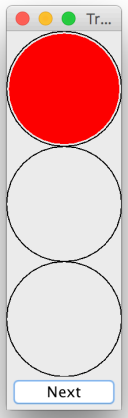
UNIVERSITÉ CAEN NORMANDIE

1. Objectif et conception MVC

Le but de ce TP est de réaliser un feu de circulation routière (tricolore), passant en boucle du vert à l'orange puis au rouge.
La conception respectera intégralement le pattern MVC tel que vu en cours :

- un modèle complètement indépendant de la vue et du contrôleur (mais écoutable)
- une vue qui s'abonne au modèle, et qui se redessine à chaque fois que le modèle prévient qu'il a changé d'état
- un contrôleur qui agit sur le modèle, en lui demandant de passer dans son état suivant (par exemple du rouge au vert).

La réalisation de l'interface graphique pourra ressembler à l'exemple ci-dessous : trois feux dont un seul est allumé (selon l'état du modèle), et un bouton agissant sur le modèle pour le faire passer dans son état suivant.



2. Réalisation du modèle

Le modèle possède comme état une couleur, parmi trois possibles.
On peut passer par des final static String pour définir trois constantes "vert", "orange" et "rouge", ou, mieux, utiliser une enum pour ceux qui savent les utiliser.

Une méthode public void suivant() permet de faire passer l'état dans la couleur suivante (avec retour au vert après le rouge)
Un getter permet d'accéder à l'état actuel du modèle (sa couleur actuelle)
Enfin, ce modèle doit être écoutable, et prévenir de chacun de ses changements. On s'appuiera pour cela sur les classes vues en cours (et disponibles sur le pdf de cours), et on fera hériter le modèle de la classe abstraite écoutable. Ne pas oublier de faire le fireChange() à l'endroit opportun.

Ce modèle doit pouvoir fonctionner en dehors de toute interface graphique (si on respecte bien le MVC). Vous devez donc pouvoir le tester dans un main, avec l'invocation de suivant() et un affichage avec System.out.println de son état.

3. Réalisation de la vue

Rappel : la vue doit avoir en attribut une référence sur son modèle, prendre ce dernier en paramètre de son constructeur, et bien sûr s'abonner auprès d'elle.
Dans ce TP, nous allons réaliser intégralement la vue en dessinant nous même la partie graphique du composant.
Pour ce faire, créer la classe de vue VueFeu qui extends JPanel, puis redéfinissez sa méthode public void paintComponent(Graphics g). En redéfinissant sa méthode, on va dessiner nous-même le contenu graphique du composant.
Afin de bien nettoyer le contenu graphique du panel avant de dessiner dedans, commencez par invoquer super.paintComponent(g), le code de la super-classe, qui dessine un contenu de panel vide.
Ensuite, utilisez les méthode de votre choix de la classe Graphics (notamment setColor, drawOval et fillOval), pour dessiner le feu tricolore conformément à l'état actuel du modèle.

Important : utilisez la méthode setPreferredSize(...) dans le constructeur pour donner la dimension souhaitée à ce composant graphique (à défaut de quoi il aura une taille nulle par défaut, ne contenant aucun composant). Exemple (en ayant défini une constante DIM correspondant au diamètre des feux, et NB_COLORS étant défini à la valeur 3) :

```
setPreferredSize(new Dimension(DIM,NB_COLORS * DIM));
```

4. Réalisation de l'interface graphique (GUI)

On peut à présent créer un sous-classe de JFrame, appelée par exemple FeuGUI, qui est la partie vue+contrôleur d'un modèle de feu. Munissez-la d'un constructeur qui prend en paramètre un Feu, et un autre qui ne prend pas de paramètre et qui instancie son propre feu.
Dans le container de ce JFrame, qui sera organisé via un BorderLayout(), mettez une instance de vue au CENTER, et un bouton "suivant" qui contrôle le modèle au SOUTH.
Testez dans un main() en écrivant :

new FeuGUI();
Puis, lorsque le feu fonctionne correctement, testez ce code ci :
Feu f=new Feu();
new FeuGUI(f);
new FeuGUI(f);
Observez ce qui se passe lorsque l'on appuie sur le bouton de l'un et de l'autre de ces deux interfaces. Attention, au lancement, les deux JFrame se superposent, il faut déplacer la seconde pour voir apparaître la première...

5. Complément

Faire en sorte que la vue adapte son contenu à sa taille : par exemple, lorsque l'on agrandit la fenêtre, les feux s'agrandissent de sorte à occuper toute la hauteur ou toute la largeur disponible, tout en respectant le ratio (les feux doivent rester ronds, pas ovales).