Un éditeur de dessin en Java avec Swing

Master STS mention Informatique spécialités GL, Mitic et Miage

GLI - Année 2010/2011

1 Introduction

Ce premier sujet, qui occupera 3 séances de TP, a pour but de permettre une prise en main des API AWT et Swing de Java. Il n'est donc ici pas encore question d'utiliser un modèle d'architecture logicielle pour la conception de cette IHM mais simplement de manipuler les mécanismes de base des composants d'interfaces ainsi que les gestionnaires d'événements.

2 La version 1.0 de l'éditeur

2.1 Le tout début...

- Réaliser un éditeur de dessin minimaliste, qui permette de dessiner des rectangles d'une couleur donnée (jaune) à l'aide de la souris, dans une zone de dessin (un JPanel) de couleur cyan. Il faut absolument donner une représentation continue du rectangle lors de l'interaction, c'est-à-dire le dessiner lors du déplacement de la souris bouton appuyé et non pas seulement lors du relachement du bouton.
- Pour cela, il faut créer un composant Editeur (dérivé de JFrame) dans lequel on place un composant ZoneDeDessin (dérivé de JPanel) auquel on aura associé des gestionnaires d'événements (implémentant MouseListener et MouseMotionListener ou bien dérivés de MouseAdapter et MouseMotionAdapter).
- Dans cette toute première phase, on ne crée pas d'objets graphiques : il faut simplement redéfinir la méthode paint de ZoneDeDessin de façon à ce qu'elle dessine un contour rectangulaire lors du déplacement de la souris, et que ce rectangle reste visible après le relachement du bouton de la souris. Il faudra au préalable avoir mémorisé le point de départ de l'interaction au moment où l'on a appuyé sur le bouton de la souris.
- Noter que si l'on n'appelle pas la méthode paint de l'ancêtre, on aura des problèmes pour afficher par la suite, et on a déjà un manque au niveau de la couleur du fond de la ZoneDeDessin.
- Vérifier que le dessin reste bien en place lorsque la fenêtre subit des modifications.
- Remarque : on n'affiche jamais plus d'un dessin à la fois avec cette version de l'éditeur...

2.2 Créer de vrais objets dessins et les déplacer

- Créer de vrais objets Dessin(s) (héritiers de JPanel) au début de chaque interaction de dessin. Il faudra alors les ajouter dans la ZoneDeDessin, au bon emplacement et avec une taille nulle. Ensuite, en cours d'interaction, on changera la taille du dessin et si besoin son emplacement. A ce niveau, on se contentera d'un Dessin rectangulaire à qui on aura donné une couleur de fond jaune.
- On veut déplacer (toujours à l'aide de la souris) les dessins que l'on a créés.
- Associer à ces objets les Listeners adéquats pour traiter leur déplacement dans la ZoneDeDessin à l'aide de la souris.

2.3 Problème avec le layout manager de ZoneDeDessin

- Constater que lorsqu'on redimensionne la ZoneDeDessin (en redimensionnant l'Editeur par exemple) les instances de Dessin sont toutes alignées et de même taille : c'est l'effet du FlowLayout qui est associé par défaut à un JPanel et qui utilise de plus la "preferredSize" des objets à réorganiser, et dont la valeur par défaut est petite.
- On remédie à ce problème en supprimant le layout manager de ZoneDeDessin (on lui donne null comme layout manager).

3 la version 2.0 de l'éditeur

3.1 De nouveaux types d'objets à dessiner

- Offrir maintenant, via des actions sur des boutons, la possibilité de créer plusieurs types de dessins (rectangles, ellipses, ...). La caractéristique commune de tous ces dessins sera qu'ils peuvent s'inscrire dans un rectangle.
- Il va donc maintenant falloir créer plusieurs classes d'objets graphiques, pour lesquelles on redéfinira la méthode **paint** de façon à obtenir le résultat visuel souhaité (**il faudra dessiner dans le repère local de l'objet graphique !**).
- Il faudra également redéfinir la méthode **contains** de ces classes de façon à s'adapter d'une part aux nouvelles formes de ces objets, et d'autre part au fait que certains de ces objets sont creux et donc qu'il ne faudrait pouvoir les saisir qu'en cliquant sur leur contour.
- Il est souhaitable, pour une meilleure décomposition objet et pour une plus grande facilité d'évolution de l'éditeur, de ne pas faire de "switch" ni de successions de tests imbriqués pour déterminer quel type de dessin doit être réalisé en cours d'interaction : on préfèrera avoir recours à une classe permettant de créer le bon type d'objet graphique (utilisation d'un patron de conception State).
- L'utilisateur doit être informé du type de dessin en cours (utiliser par exemple des JRadioButton(s)).

3.2 Pouvoir choisir la couleur des objets à dessiner

- Offrir également la possibilité de choisir une couleur à utiliser pour les (prochains) dessins.
- Il faudra en tenir compte pour les dessins intermédiaires et les créations d'objets graphiques.
- L'utilisateur doit être informé du type de la couleur en cours :
  - utiliser ici des JCheckBox(es), "vus" comme des JRadioButton(s) (même si, d'un point de vue sémantique, ce n'est pas l'usage habituel des cases à cocher...),
  - visualiser cette couleur dans une zone appropriée : utiliser ici un patron de conception Observateur pour être prévenu de tout changement de couleur de dessin. Il sera possible ici d'utiliser l'attribut **foreground** de la zone de dessin pour stocker la couleur des dessins à créer, et profiter ainsi de la propagation des changements de valeurs de cet attribut vers les abonnés à ces changements, puisque les composants Swing sont construits selon le modèle des java beans.

3.3 Accéder de plusieurs façons aux fonctionnalités

- Offrir également l'accès aux fonctionnalités (choix du type de dessin et de la couleur) via des menus déroulants accessibles depuis une barre de menu (JMenuBar, JMenu et par exemple JRadioButtonMenuItem ou JCheckBoxMenuItem). Il sera possible de profiter ici de l'architecture MVC des composants Swing et d'utiliser le partage de modèles entre composants.
- Il faudra que les items des menus et les boutons soient dans un état cohérent.
- Associer des raccourcis clavier à certaines fonctionnalités.

3.4 Offrir plusieurs modes de fonctionnement de l'éditeur

- Permettre à l'utilisateur de déterminer quelle est l'action par défaut appliquable sur un dessin, par exemple le déplacer, l'éloigner ou le rapprocher, le détruire, le redimensionner, ou encore ne rien faire (c'est-à-dire autoriser de commencer un nouveau dessin en cliquant au dessus d'un dessin existant). Il faudra pour cela parcourir les dessins de l'éditeur et leur appliquer des changements de listeners...

3.5 Toujours plus d'informations pour l'utilisateur

- Insérer des ToolTip(s) là où c'est possible pour donner un maximum d'informations à l'utilisateur.
- Modifier l'indicateur de couleur (celui qui observe les changements de foreground de la zone de dessin) de façon à ce qu'il observe aussi les changements de créateur de dessins, et pourquoi pas aussi ses changements de background : il faut alors le faire réagir à ces changements en affichant un dessin de la bonne forme, avec la bonne couleur, sur le bon fond.