TP : Transfert de données

Objectifs

– se familiariser avec les techniques de drag and drop et (couper,copier) - coller.

Les tutoriels relatifs au drag and drop et au couper,copier - coller sont disponibles sur le site de Sun à l'adresse suivante http ://java.sun.com/docs/books/tutorial/uiswing/dnd/index.html [1]

1 Fonctionnalités directement supportées par Swing

La prise en charge des opérations de transfert de données est disponible pour les composants Swing suivants : JEditorPane, JFormattedTextField, JPasswordField, JTextArea, JTextField, JTextPane, JColorChooser, JFileChooser, JList, Jtable et JTree.

Question 1. Utilisez le fichier DefaultDnDSupport.java [2] pour tester les opérations de copier-coller et de drag and drop entre composants swing et avec des applications externes.

Le drop est supporté par défaut par ces composants (sauf pour `Jlist`, `Jtable` et `JTree`). Activez le drag sur chacun des composants avec la méthode `setDragEnabled(true)` et répétez les opérations de drag and drop. Constatez les différences.

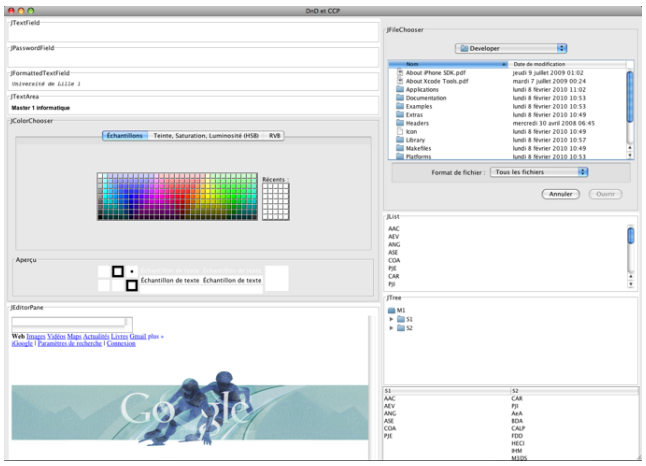

FIGURE 1 – Prise en charge par défaut du drag and drop et des (couper,copier) - coller par Swing.

2 La classe `TransferHandler`

La classe TransferHandler [3] permet de définir les méthodes d'importation et d'exportation de données. Chaque composant possède la méthode `setTransferHandler` (définie dans `JComponent`) qui permet soit de remplacer un `TransferHandler` défini par défaut ou de spécifier un `TransferHandler` pour un composant qui ne supporte pas le transfert de données initialement.

2.1 Transfert de propriétés JavaBeans

Les JavaBeans [4] sont des composants Java réutilisables et facilement intégrables au sein d'un IDE. A chaque composant Swing tel que JLabel correspond un JavaBean dont les propriétés peuvent être listées

1. `http://java.sun.com/docs/books/tutorial/uiswing/dnd/index.html`
2. `DefaultDnDSupport.java`
3. `http://java.sun.com/javase/6/docs/api/javax/swing/TransferHandler.html`
4. `http://java.sun.com/docs/books/tutorial/javabeans/index.html`

en utilisant un IDE tel que NetBeans (Figure 3).

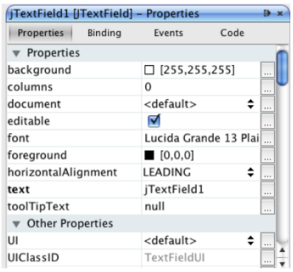

FIGURE 2 – Propriétés JavaBean du composant JLabel.

Question 2. Réalisez une interface comprenant un composant `JLabel` et deux composants `JFormattedTextField`. Modifiez la police par défaut de chacun de ces composants et ajoutez les TransferHandler correspondants pour transférer la propriété de police d'un composant à un autre. Pour détecter le geste de drag sur le `JLabel`, vous spécifierez le listener adéquat avec la méthode `addMouseMotionListener` dans lequel vous appellerez la méthode `exportAsDrag` qui se charge d'appeler les méthodes d'exportation de données (voir le cours pour plus de détails).

Testez les opérations de drag and drop et de copier-coller entre les différents composants. Notez qu'il est nécessaire de sélectionner du texte pour pouvoir démarrer le drag du `JFormattedTextField`. Comment résoudre ce problème ?

Lancez une deuxième instance de votre application et testez les opérations de transfert entre les deux fenêtres.

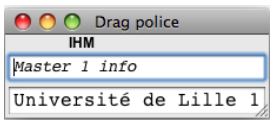

FIGURE 3 – Transfert de propriétés JavaBeans entre composants.

2.2 Ecrire son propre `TransferHandler`

Nous allons ici créer notre propre TransferHandler [5] afin de pouvoir transférer du texte entre un JLabel et d'autres composants Java ou des applications externes.

2.2.1 Exportation de chaînes de caractères

Question 3. Créez une classe qui hérite de `TransferHandler` et définissez les méthodes `getSourceActions`, `createTransferable` et `exportDone` (voir le cours pour avoir plus de détails sur la façon de définir ces méthodes). Il doit être possible de copier ou de déplacer le texte du JLabel. Testez et vérifiez qu'il est possible de copier ou déplacer le texte du JLabel vers un autre composant qui accepte le texte ou une autre application.

2.2.2 Importation de chaînes de caractères

Question 4. Définissez les méthodes `canImport` et `importData` pour vérifier la possibilité d'importation de données et effectuer l'importation lorsque cela est possible. Pour l'importation, vous pourrez par exemple concaténer le texte importé au texte existant.

5. `http://java.sun.com/javase/6/docs/api/javax/swing/TransferHandler.html`

3 DataFlavor et Transferable personnalisés

3.1 Transfert de couleurs

Le but est ici de créer sa propre saveur de données (DataFlavor) pour transférer la couleur d'arrière plan d'un composant à un autre. On pourrait très bien utiliser les propriétés JavaBeans mais nous souhaitons également pouvoir transférer la couleur sous forme de texte vers des applications externes ou des composants qui ne supportent que le format texte.

Pour créer un DataFlavor, il faut préciser la classe correspondante. Nous utiliserons ici la classe Couleur définie dans le fichier Couleur.java [6].

Question 5. En vous aidant du cours, créez une classe `TransferableCouleur` qui implémente l'interface `Transferable` afin de définir le DataFlavor couleurFlavor et les méthodes d'accès aux données.

Question 6. Définissez maintenant la classe `TransfertCouleur` qui hérite de TransferHandler qui permet d'importer et exporter des données de type `Couleur`.

Question 7. Testez le drag and drop et copier-coller de couleurs entre composants Java et avec des applications externes.

3.2 Transfert d'images

La saveur de données pour le transfert d'images est donnée par `DataFlavor.imageFlavor`. Il vous reste cependant à écrire les `Transferable` et `TransferHandler` correspondants.

Question 8. En vous inspirant des questions précédentes, écrivez `TransferableImage` implémentant l'interface `Transferable` et `ImageHandler` qui hérite de la classe `TransferHandler`.

Question 9. Ecrivez une application de test avec deux labels auxquels vous aurez affecté une image avec la méthode `setIcon`. Il doit être possible de faire un drag and drop d'un label à l'autre pour remplacer l'image initiale ; de faire un DnD d'un label vers une application externe pour copier l'image d'un label ; et de faire un DnD d'une application externe (type firefox) vers un des deux labels pour remplacer l'image du label.

FIGURE 4 – Transfert d'images.

Question 10. Modifiez les méthodes `canImport` et `importData` de la classe `ImageHandler` afin de pouvoir ouvrir une image glissée du gestionnaire de fichier sur un label. Pour cela vous ajouterez le support pour le `DataFlavor.javaFileListFlavor` dans la méthode `canImport`. Pour modifier la méthode `importData`, les lignes de code suivantes permettent de récupérer le premier nom de fichier avec le chemin absolu d'une liste de fichiers déposée sur un composant.

```
String filename = "";
List files = (List) (t.getTransferData(DataFlavor.javaFileListFlavor));
filename = ((File)files.get(0)).getAbsolutePath();
```

6. Couleur.java

4 Ajout du support pour le copier-coller

Le (couper,copier)-coller utilise le même `TransferHandler` que celui défini pour un drag and drop. Pour appeler les méthodes du `TransferHandler`, il est nécessaire d'utiliser les input et action maps qui seront vues en cours.

Pour ajouter l'option de coller dans un label une image qui a été copiée précédemment et qui se trouve dans le presse-papier, vous pouvez ajouter les deux lignes de code suivantes à votre label :

```
label.getInputMap().put(KeyStroke.getKeyStroke(KeyEvent.VK_V, InputEvent.CTRL_MASK), "paste")
label.getActionMap().put("paste", TransferHandler.getPasteAction());
```