Interaction Humain-Machine

TP : Gestion des événements (3/3)

Objectifs

— mettre en œuvre de ListView

— comprendre la notion de modèle associé à un widget et s'abonner à ses changements

1 Préambule : rappels du cours

Les listes sont des widgets qui permettent de sélectionner un ou plusieurs éléments dans un ensemble dont le cardinal peut être important et variable. Il est possible de sélectionner un élément par simple clic d'un bouton de souris, plusieurs éléments en maintenant la touche CTRL enfoncée ou des intervalles en utilisant la touche SHIFT. L'exemple ci-dessous illustre la création d'une liste en utilisant la classe ListView [1] de JavaFX avec la mise en place d'un ListChangeListener [2] qui permet d'être prévenu lorsque la sélection des éléments de la liste change. Un SelectionModel [3], qui contient l'ensemble des items sélectionnés, est associé à la ListView. La méthode getSelectedItems() [4] permet d'obtenir une ObservableList [5] qui représente la liste des items sélectionnés.

```
public class ListeSimple extends Application {
  Label label;

  class MonListChangeListener implements ListChangeListener<String> {
    public void onChanged(Change<? extends String> report) {
      label.setText("Selection de " + report.getList());
    }
  }

  public void start(Stage stage) {
    label = new Label("Aucune selection");
    ListView<String> list = new ListView<String>();
    list.getItems().addAll("Paris", "Berlin", "Londres", "Rome", "Lisbonne", "Madrid",
                           "New York", "Tokyo", "Pekin");
    list.getSelectionModel().getSelectedItems().addListener(new MonListChangeListener());

    HBox root = new HBox();
    root.setAlignment(Pos.CENTER_LEFT);
    root.setSpacing(10.0);
    root.setPadding(new Insets(3, 3, 3, 3));
    root.getChildren().addAll(list,label);

    Scene scene = new Scene(root, 400, 150);
    stage.setTitle("Simple liste");
    stage.setScene(scene);
    stage.show();
  }

  public static void main(String[] args) {
    Application.launch(args);
  }
}
```

Q 1. Testez l'exemple ci-dessus.

1. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/ListView.html
2. https://docs.oracle.com/javase/8/javafx/api/javafx/collections/ListChangeListener.html
3. https://docs.oracle.com/javafx/2/api/javafx/scene/control/SelectionModel.html
4. http://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/MultipleSelectionModel.html#getSelectedItems--
5. https://docs.oracle.com/javase/8/javafx/api/javafx/collections/ObservableList.html

2 FileExplorer

Le but de cet exercice est de réaliser un mini explorateur de fichiers qui affiche le contenu d'un répertoire et, le cas échéant, le contenu d'un sous-répertoire, comme illustré sur la figure 1. L'interface comprend deux `ListView` et un `Label`. La `ListView` de gauche affiche le contenu d'un répertoire principal. Quand un répertoire de cette liste est sélectionné, son contenu est affiché dans la `ListView` de droite. Les fichiers sélectionnés dans le répertoire principal ou un des sous-répertoires sont affichés dans le `Label`. Rien n'est affiché dans la `ListView` de droite quand des fichiers sont sélectionnés dans la `ListView` de gauche. Pour distinguer les répertoires des fichiers, l'affichage des `ListView` est personnalisé par des icônes  [6]  [7]. La `ListView` de gauche ne supporte que la simple sélection alors que la `ListView` de droite peut supporter la multi-sélection.

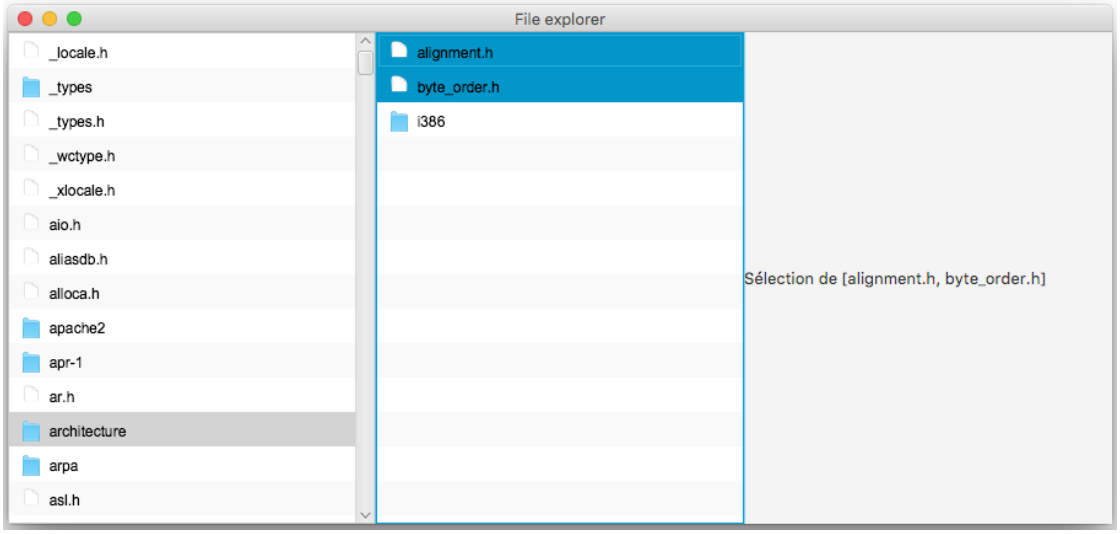

FIGURE 1 – FileExplorer.

Pour obtenir la liste des fichiers et sous-répertoires d'un répertoire, vous pouvez utiliser le code ci-dessous. La méthode `isDirectory()` de la classe File [8] permet de savoir si le chemin associé correspond à un fichier ou un répertoire.

```
File path = new File("/usr/include/");
String[] filelist = path.list();
```

Q 2. Modifiez l'exemple donné dans le préambule pour afficher l'ensemble des fichiers et sous-répertoires d'un répertoire.

Q 3. Modifiez l'interface de manière à vous approcher de ce qui est représenté sur la figure 1.

Q 4. Lorsque vous sélectionnez un répertoire dans la liste de gauche, affichez le contenu du sous-répertoire dans la liste de droite. À l'issue de cette question, le comportement doit correspondre à ce qui est demandé au début de l'exercice. Il ne reste plus qu'à personnaliser l'affichage des listes afin de distinguer les répertoires des fichiers.

6. `file.png`
7. `folder.png`
8. `https://docs.oracle.com/javase/8/docs/api/java/io/File.html`

La personnalisation de l'affichage des cellules (Cell [9]) se fait par l'utilisation d'un CellFactory, comme illustré ci-dessous.

```
malistview.setCellFactory(new Callback<ListView<String>,
        ListCell<String>>() {
            @Override
            public ListCell<String> call(ListView<String> list) {
                return new MonRenduDeCellule();
            }
        }
);
```

Le rendu de chacune des cellules se fait ici par un appel de la méthode updateItem [10] qui permet ici de dessiner dans un Canvas.

```
class MonRenduDeCellule extends ListCell<String> {
    @Override
    public void updateItem(String item, boolean empty) {
      super.updateItem(item, empty);
      Canvas c = new Canvas(200, 20);

      // A Completer

      setGraphic(c);
    }
}
```

Q 5. Gérez votre rendu de chaque cellule de manière à ajouter les bonnes icônes à côté des fichiers et répertoires.

3 Bonus

Q 6. Ouvrez directement des fichiers avec l'application dédiée en double cliquant sur un fichier de la liste de droite. Pour cela utiliser la commande xdg-open [11] et `Runtime.getRuntime().exec(votre commande)` pour exécuter une commande système. Testez d'abord `xdg-open` dans un terminal.

Q 7. Changer le comportement des deux listes pour pouvoir explorer une arborescence de fichiers : il doit être désormais possible d'ouvrir un répertoire dans la liste de droite. Dans ce cas, le contenu de la liste de droite passe à gauche et la liste de droite affiche le contenu du nouveau répertoire. Il faut ajouter un bouton pour remonter dans l'arborescence.

9. https://docs.oracle.com/javafx/2/api/javafx/scene/control/Cell.html
10. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Cell.html#updateItem-T-boolean-
11. https://linux.die.net/man/1/xdg-open