Interfaces graphiques - JavaFX

responsable : Wiesław Zielonka
zielonka@liafa.univ-paris-diderot.fr
http://liafa.univ-paris-diderot.fr/~zielonka

February 16, 2016

- La classe principale d'un programme JavaFX doit être une sous-classe de `javafx.application.Application`. L'exécution commence dans la méthode `start()` de `Application`.
- l'argument `Stage` de `start()` – le container principal de l'application JavaFX.
- `Scene` - un container. Le contenu de `Scene` une hiérarchie de noeuds.
- `main()` contient l'appel à la méthode `launch()` pour lancer l'application.

HelloWorld

```
public class HelloWorld extends Application {

    @Override
    public void start(Stage primaryStage) {
        Button btn = new Button();
        btn.setText("Say 'Hello World'");
        Label label = new Label();
        btn.setOnAction(new EventHandler<ActionEvent>() {

            @Override
            public void handle(ActionEvent event) {
                label.setText(btn.getText());
            }
        });
```

HelloWorld (suite)

```
        VBox root = new VBox(8);
        root.setAlignment(Pos.CENTER);
        //root.getChildren().add(btn);
        //root.getChildren().add(label);
        root.getChildren().addAll(btn,label);
        Scene scene = new Scene(root, 300, 250);

        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }
}
```

Le graphe de la scène

- Le graphe de la scène est un arbre composé de noeuds (classe `Node`). Dans l'exemple précédent `VBox`, `Button`, `Label` sont tous des sous-classes de `Node`.
- `VBox` est à la racine du graphe de la scène et il a deux enfants, `label` et `Button`.
- Le noeud qui peut avoir plusieurs enfants est on objet de la classe `Parent`, donc `VBox` à la racine est une sous-classe de `Parent`.
- Chaque `Parent` est aussi un `Node` (la classe `Parent` est une sous-classe de `Node`).

Layouts – gestionnaires de positions

Toutes les classes layouts sont `Parent`.
Quelques layouts :

- BorderPane
- HBox et VBox
- StackPane
- GridPane
- FlowPane
- TilePane
- AnchorPane

BorderPane

Les positions : top, bottom, left, right, center.

```
BorderPane border = new BorderPane();
border.setLeft(node1);
border.setCenter(node2);
border.setTop(node3);
//etc
```

HBox - boîte horizontale

```
HBox hbox = new HBox();
//espace autour HBox
hbox.setPadding(new Insets(15, 12, 15, 12));
hbox.setSpacing(10);//entre les noeuds
hbox.setStyle("-fx-background-color: #336699;");

Button buttonCurrent = new Button("Current");
buttonCurrent.setPrefSize(100, 20);

Button buttonProjected = new Button("Projected");
buttonProjected.setPrefSize(100, 20);
hbox.getChildren().addAll(buttonCurrent, buttonProjected);
```

VBox - boîte verticale.

FlowPane

Les noeuds "flottent" un après l'autre soit horizontalement (par défaut) soit verticalement.

```
Image images[] = { ... };
FlowPane flow = new FlowPane();
flow.setVgap(8);
flow.setHgap(4);
flow.setPrefWrapLength(300); // preferred width = 300
for (Image im : images) {
    flow.getChildren().add(new ImageView(im));
}
```

GridPane

GridPane - container de type grille.

```
GridPane grid = new GridPane();
grid.setAlignment(Pos.CENTER);
grid.setHgap(10);
grid.setVgap(10);
grid.setPadding(new Insets(25, 25, 25, 25));
```

Horizontal/vertical gap - espace horizontal/vertical entre les les lignes et les colonnes de la grille.
padding – espace autour de la grille.
Alignement - la position d'un noeud dans dans une cellule.

GridPane (suite)

Ajouter des éléments :

```
//colonne=1,ligne=0
grid.add(new Label("Welcome"), 1, 0);
TextField tf = new TextField(12);
//colonne=0,ligne=1,colspan=2,rowspan=1
grid.add(tf,0,1,3,1);
Button btn = new Button("Sign in");
HBox hbBtn = new HBox(10);
hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
hbBtn.getChildren().add(btn);
grid.add(hbBtn, 1, 3);
//grille visible pour debugage
grid.setLinesVisible(true);
```

StackPane

`StackPane` permet d'empiler les noeuds enfants un sur l'autre, le nouveau noeud sur le précédent (par exemple un texte sur un cercle).

TilePane

Similaire à FlowPane

AnchorPane

Permet accrocher les enfant sur les bords du layout.

```
AnchorPane anchorPane = new AnchorPane();
// List should stretch as anchorPane is resized
ListView list = new ListView();
//accrocher au top, left, right au meme temps
AnchorPane.setTopAnchor(list, 10.0);
AnchorPane.setLeftAnchor(list, 10.0);
AnchorPane.setRightAnchor(list, 65.0);
// accrocher le boton a droite et au bottom
Button button = new Button("Add");
AnchorPane.setBottomAnchor(button, 10.0);
AnchorPane.setRightAnchor(button, 10.0);
anchorPane.getChildren().addAll(list, button);
```

Widgets (contrôles/controls)

- Label
- Button
- RadioButton et ToggleGroup
- CheckBox
- ChoiceBox et ComboBox (ComboBox à utiliser si le nombre de d'éléments élevé, si le nombre d'éléments limité les deux font affaire); ComboBox peut être éditable, ChoiceBox non.
- TextField
- PasswordField
- ScrollBar
- ScrollPane
- ListView
- TableView, TreeView, TreeTableView

- Separator (à utiliser par exemple pour séparer des éléments dans ListView)
- Slider
- ProgressBar et ProgreIndicator
- Hyperlink
- Tooltip
- HTMLEditor
- TitlePane et Accordion
- ColorPicker
- DatePicker
- PaginationControl
- FileChooser

Label

```
String title = "Layouts";
Label label = new Label(title);
label.setFont(Font.font("Cambria", 20));
```

CheckBox

```
CheckBox[] check = new CheckBox[16];
for (int i = 0; i < check.length; i++) {
    check[i] = new CheckBox();
}
ChangeListener<Boolean> chListener =
                    new ChangeListener<Boolean>() {
  @Override
  public void changed(ObservableValue<? extends Boolean> ov,
                      Boolean old_val, Boolean new_val) {
    int i = 0;
    for (int j = check.length -1; j >= 0; j--) {
       i = (i << 1) | (check[j].isSelected() ? 1 : 0);
    }
    model.setValue(i);
  }
};
for(int i=0 ;i < check.length; i++){
   check[i].selectedProperty().addListener(chListener);
}
```

CheckBox suite

- `ChangeListener` est en fait `javafx.beans.value.ChangeListener`.
- Listener surveille les changements de la propriété `selected` d'un `CheckBox`, `selected` est une propriété booléenne.
- les arguments `old_val` et `new_val` de la méthode `changed` de listener – l'ancienne et nouvelle valeur de la propriété.

CheckBox avec EventHandler

Au lieu d'installer `ChangeListener` on peut installer EventHandler qui sera activé par un `actionEvent` déclenché suite à l'action de l'utilisateur :

```
EventHandler eh = new EventHandler<ActionEvent>(){
    @Override
    public void handle(ActionEvent event) {
        int i = 0;
        for(int j= check.length-1; j>=0; j--){
            i=(i<<1)|(check[j].isSelected()?1:0);
        }
        model.setValue(i);
    }
};
for(int i=0; i<check.length; i++){
    check[i].setOnAction(eh);
}
```

TextField

```
class TextView extends TextField implements Observer{
  private int base;
  private Model model;
  TextView(int base, Model model){
     //largeur de TextField
     setPrefColumnCount(16);
     this.base = base;
     this.model=model;
```

TextField – installer EventHandler

```
setOnAction(new EventHandler<ActionEvent>(){
    @Override
    public void handle(ActionEvent event) {
      int i;
      try{
        i = Integer.parseUnsignedInt(getText(),base);
        if(i > 0xFFFF){
          //lancer un dialogue d'erreur
          Alert alert = new Alert(AlertType.ERROR,
                  "the number cannot be greater than " +
                  Integer.toString(0xffff,base));
          alert.showAndWait();
          model.setValue(model.getValue());
          return;
        }
        model.setValue(i);
     }
     catch(NumberFormatException e){
          Alert alert = new Alert(AlertType.ERROR,
                  "Incorrect number format");
          alert.showAndWait();
          model.setValue(model.getValue());
```

ProgressBar et ProgressIndicator

Les éléments passifs (pas de handlers, pas de listeners à installer).

Les deux contrôles prennent les valeurs entre 0 et 1.

```
ProgressBar bar = new ProgressBar();
ProgressIndicator indicator = new ProgressIndicator();

float i = ... ; //valeur stockee dans le modele
int j = 0xFFFF; //valeur maximale
bar.setProgress(i / j);
indicator.setProgress(i / j);
```

ChoiceBox

Dans cet exemple `ChoiceBox` contient des éléments de la classe Integer.

```
public class ChoiceBoxView extends ChoiceBox<Integer> {
    private Model model;
    //intList sert a stocker les elements de ChoiceBox
    ObservableList<Integer> intList;

    {
        Integer[] inta = new Integer[16];
        int k = 1;
        for (int i = 0; i < 16; i++) {
            inta[i] = new Integer(k);
            k <<= 1;
        }
        //les donnees de ChoiceBox sont stockees
        //dans ObservableList<Integer>
        intList = FXCollections.observableArrayList(inta);
    }
```

ChoiceBox (suite)

Installer listener activé par le changement de la propriété `selectedIndex` :

```
public ChoiceBoxView(Model model) {
    this.model = model;
    setItems(intList);
    //installer listener pour la propriete selectedIndex
    getSelectionModel().selectedIndexProperty()
            .addListener(new ChangeListener<Number>() {

        @Override
        public void changed(ObservableValue<? extends Number> o
                Number old_val, Number new_val) {
          model.setValue(intList.get(new_val.intValue())
                  .intValue());
        }

     });
}
```

Notez l'utilisation `new_value` qui donne l'index de l'élément sélectionné par l'utilisateur.

Slider

```
Slider lSlide = new Slider();
lSlider.setMin(0);
lSlider.setMax(0xFF);
lSlider.setShowTickLabels(true);
lSlider.setShowTickMarks(true);
lSlider.setMajorTickUnit(16);
lSlider.setBlockIncrement(1);
```

Slider - installer ChangeListener

`ChangeListener<Number>` pour la propriété `valueProperty`. On utilisera le même listener pour les deux Sliders, `lSlider` et `hSlider`.

```
ChangeListener<? super Number> listener;
listener = new ChangeListener<Number>() {
    @Override
    public void changed(ObservableValue<? extends Number> ov,
                Number old_val, Number new_val) {
        //recuperer les valeurs de deux Sliders
        int l = lSlider.valueProperty()
                    .getValue().intValue();
        int h = hSlider.valueProperty()
                    .getValue().intValue();
        model.setValue(l | (h << 8));
    }
};
```

Les noeuds avec un titre

```
public class TitledView extends VBox{

    public TitledView(String title, Node ... node){
        Label label = new Label(title);
        label.setFont(Font.font("Cambria", 20));
        getChildren().add(label);
        getChildren().addAll(node);
    }

}
```