TP : Gestion des événements

Objectifs

- se familiariser avec les différents types d'événements et leur gestion
- gestion des événements du type ActionEvent, MouseEvent, ScrollEvent sur des widgets simples
- prise en main du ChangeListener sur des widgets plus évolués

Pré-requis

- Maîtriser le cours 4 [1].

1 Retour sur le dernier TP

Consultez la correction du dernier TP disponible sur Moodle et posez des questions à votre enseignant, si vous en avez.

2 ActionEvent, MouseEvent, ScrollEvent

Le code ci-dessous permet d'obtenir l'interface représentée Figure 1.

```
public class ButtonsAndLabel extends Application {

        public void start(Stage stage) {
                Label label = new Label("0");
                label.setMaxSize(Double.MAX_VALUE, Double.MAX_VALUE);
                label.setStyle("-fx-background-color: lightblue;"
                                + " -fx-alignment: center;"
                                + " -fx-font: 30px Verdana;");
                Button bmoins = new Button("  -  ");
                Button bplus = new Button("  +  ");

                VBox vbox = new VBox(3);
                vbox.setPadding(new Insets(3, 3, 3, 3));
                vbox.setAlignment(Pos.CENTER);
                HBox hbox = new HBox(3);
                hbox.getChildren().addAll(bmoins, bplus);
                vbox.getChildren().addAll(label, hbox);

                Scene scene = new Scene(vbox);
                stage.setTitle("Counter");
                stage.setScene(scene);
                stage.show();
        }

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

Q 1. Ajoutez une classe interne pour gérer les `ActionEvent` sur les boutons, de façon à incrémenter le compteur quand on clique sur le bouton "+" et le décrémenter quand on clique sur le bouton "-".

Q 2. On souhaite maintenant pouvoir interagir directement **avec le label** en utilisant la souris : un clic droit sur le label incrémente sa valeur, un clic gauche la décrémente. À l'aide d'une lambda expression

1. https://www.iut-info.univ-lille.fr/~gery.casiez/R2.02/Cours/R2.02-Cours4.pdf

FIGURE 1 – Capture d'écran de l'interface compteur. Les boutons "+" et "-" permettent respectivement d'incrémenter et de décrémenter le compteur. Il est aussi possible de le modifier avec des clics droit et gauche sur le label, un cliquer-glisser de la souris ou un geste de défilement.

gérez l'événement souris "MousePressed" pour mettre correctement à jour le label suivant le bouton utilisé.

Q 3. On souhaite à présent mettre à jour le label par un cliquer-glisser de la souris : cliquer et faire glisser la souris **sur le label** vers la droite augmente la valeur initiale du label d'une valeur proportionnelle à la distance entre la position initiale du pointeur et sa position courante. La valeur diminue quand le pointeur de la souris va vers la gauche. Gérez les événements "MouseDragged" en implémentant directement un EventHandler dans la classe ButtonsAndLabel.

Q 4. Enfin on souhaite modifier la valeur du label en utilisant les événements de défilement (rotation de la molette de la souris ou déplacement de deux doigts sur un pavé tactile). En utilisant la façon de gérer les événements de votre choix, gérez les événements "Scroll".

3 Convertisseur de devises : gestion des ActionEvent et mise en oeuvre des ChangeListener

Dans cet exercice, vous allez concevoir une application permettant de convertir une devise en une autre.

3.1 Version initiale avec des boutons

Q 5. Pour cette version, vous définirez une interface graphique (voir figure 2), composée de deux zones de saisies et de deux boutons réalisant les conversions d'une devise vers l'autre. Il faudra bien sûr chercher le taux de conversion correct.

FIGURE 2 – Convertisseur de devises avec des boutons. Il faut cliquer sur un des deux boutons pour réaliser la conversion.

3.2 Version sans bouton

Q 6. Les boutons de l'interface précédente ne sont pas très utiles. Dès que l'utilisateur met à jour un montant, il est facile de savoir dans quel sens réaliser la conversion. Réalisez une interface pour obtenir un résultat similaire à la figure 3. Le montant est mis à jour dans l'autre devise quand l'utilisateur appuie sur la touche entrée (gestion des `ActionEvent` sur les TextField).

FIGURE 3 – Convertisseur de devises sans bouton.

3.3 ChangeListener

Q 7. L'objectif est maintenant de mettre à jour l'autre devise dès que l'utilisateur en modifie une. Il faut pour cela ajouter un ChangeListener [2] à chaque TextField. L'ajout se fait par la méthode `addListener` de `textProperty()` du `TextField`. Commencez par ajouter un ChangeListener sur un des deux TextField. Ajoutez-en ensuite un sur l'autre TextField et constatez le problème que cela pose. Pour résoudre le problème, il est nécessaire de connaître lequel des deux TextField a le focus (méthode `isFocused()`).

3.4 Version avec une combo box

Q 8. Nous souhaitons maintenant convertir plusieurs devises (Euros <-> Livres, Euros <-> Dollars, etc). Adaptez l'interface précédente en y ajoutant une combo box pour indiquer le type de conversion à réaliser (figure 4). Pour gérer les événements de la combo box il faudra à nouveau utiliser un `ChangeListener` et l'ajouter avec la méthode `getSelectionModel().selectedItemProperty().addListener`.

FIGURE 4 – Convertisseur de devises avec combo box pour choisir le type de conversion.

3.5 Version avec deux interfaces

Il est fréquent de devoir créer des interfaces avec des enchaînements d'écrans, où chaque écran présente une interface particulière. Dans ce cas, passer d'une interface à une autre consiste à remplacer le noeud racine de la scène courante par un autre avec la méthode `setRoot` de l'objet `Scene`.

Q 9. Reprenez les interfaces créées pour les deux questions précédentes et ajoutez à chacune un bouton destiné à passer d'une interface à l'autre. Rajoutez la gestion des événements sur ces boutons pour effectivement afficher une interface ou l'autre, de manière à obtenir un résultat similaire à la Figure 5.

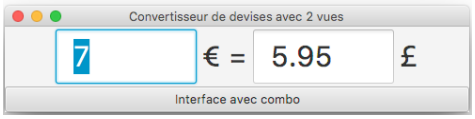

FIGURE 5 – Convertisseur de devises avec deux interfaces.

4 BONUS : sauvegarde de préférences

Java met à disposition la classe Preferences [3] qui permet de sauvegarder de manière persistante des préférences d'une application. Java se charge d'enregistrer les données de la meilleure façon possible pour chaque système utilisé.

L'accès aux paramètres se fait par l'utilisation de méthodes get, en précisant le nom du paramètre et sa valeur par défaut. La modification des paramètres se fait par l'utilisation de méthodes put. L'exemple

2. http://docs.oracle.com/javase/8/javafx/api/javafx/beans/value/ChangeListener.html
3. https://docs.oracle.com/javase/8/docs/api/java/util/prefs/Preferences.html

minimal suivant illustre l'utilisation des préférences. Notez bien ce qui est affiché dans la console lors du premier lancement et lors des lancements suivants de l'application.

```
public class HelloSettings {

        public static void main(String[] args) {
                Preferences prefs = Preferences.userRoot().node("lesPreferencesDeMonApplicati
                double p1 = prefs.getDouble("parametre1", 10.0);
                boolean p2 = prefs.getBoolean("parametre2", true);
                System.out.println("Valeur p1 = " + p1 + " valeur p2 = " + p2);

                prefs.putDouble("parametre1", 11.0);
                prefs.putBoolean("parametre2", false);
                p1 = prefs.getDouble("parametre1", 10.0);
                p2 = prefs.getBoolean("parametre2", true);
                System.out.println("Valeur p1 = " + p1 + " valeur p2 = " + p2);
        }
}
```