Interaction Humain-Machine

TP : Prototypage haute fidélité

Objectifs

- Utilisation du GUI Builder **JavaFX Scene Builder 2.0**
- Prise en main de FXML et de la gestion des événements

1 Préambule

Vous allez réaliser un prototype haute fidélité de la version basse fidélité conçue à la séance précédente. Aujourd'hui le but n'est pas de réaliser une interface complètement fonctionnelle mais de rendre interactives certaines parties critiques de l'interface pour lesquelles les retours des utilisateurs sont importants.

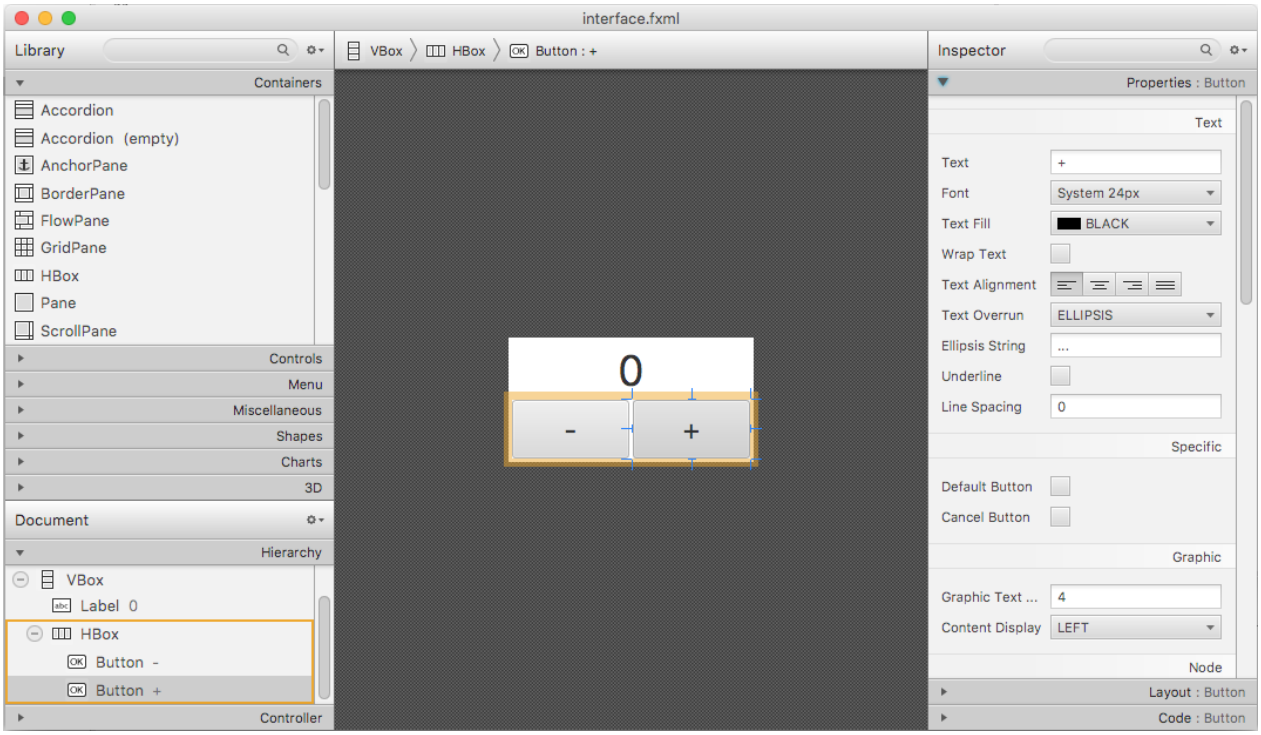

FIGURE 1 – Capture d'écran de JavaFX Scene Builder 2.0

2 JavaFX Scene Builder 2.0

JavaFX Scene Builder 2.0 est déjà installé sur les machines des salles TP (/opt/JavaFXSceneBuilder2.0/). Il peut être facilement installé sur n'importe quelle plateforme en suivant les instructions disponibles ici [1]. C'est un logiciel à part entière qui permet de créer des interfaces pour JavaFX par drag-and-drop de widgets et définition de leurs propriétés. Le logiciel génère un fichier fxml qui contient une représentation au format XML de l'interface. Ce fichier pourra ensuite être chargé par votre programme JavaFX.

1. https://gluonhq.com/products/scene-builder/

3 Création de l'interface

Q 1. En utilisant JavaFX Scene Builder 2.0, reproduisez l'interface du compteur présenté dans le TP 3, de manière à obtenir un résultat similaire à la Figure 1.

4 Chargement du fichier FXML

En supposant que votre interface est enregistrée dans le fichier interface.fxml, le code suivant permet de charger ce fichier et d'afficher l'interface. Le fichier fxml doit ici se trouver dans le même répertoire que les .java

```
public class FXMLdemo extends Application {

        public void start(Stage stage) throws IOException {
                FXMLLoader loader = new FXMLLoader();
                URL fxmlFileUrl = getClass().getResource("interface.fxml");
                if (fxmlFileUrl == null) {
                        System.out.println("Impossible de charger le fichier fxml");
                        System.exit(-1);
                }
                loader.setLocation(fxmlFileUrl);
                Parent root = loader.load();

                Scene scene = new Scene(root);
                stage.setScene(scene);
                stage.setTitle("FXML demo");
                stage.show();
        }

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

Q 2. Testez.

5 Gestion des événements

La gestion des événements pour incrémenter le label quand on clique sur le bouton + nécessite de :

1. Tout en bas à gauche de la fenêtre de Scene Builder, dans la partie "Controller", donnez le nom de la classe qui va gérer les événements dans le champ de texte "Controller class". Si par exemple la classe MonController du package tp6 va gérer les événements, on ajoutera : **tp6.MonController**.
2. Dans Scene Builder donner un **fx :id** au Label (dans la rubrique Code). Ici vous donnerez la valeur monLabel.
3. Préciser quelle méthode sera appelée quand le bouton + reçoit un Action Event. Dans la rubrique Code du bouton + indiquez pressedButtonPlus en dessous de On Action.
4. Créer la classe MonController comme donné en exemple ci-dessous. Notez que la syntaxe @FXML permet d'instancier automatiquement monLabel avec la référence du label de l'interface, grâce au fx :id qui a été précisé !

```
public class MonController {
        @FXML
        Label monLabel;

        public void initialize() {
                System.out.println("Initialisation...");
        }

        public void pressedButtonPlus(ActionEvent event) {
                int newValue = Integer.parseInt(monLabel.getText()) + 1;
                monLabel.setText("" + newValue);
```

```
        }
}
```

Q 3. Testez le code ci-dessus.

Q 4. Faites le nécessaire pour faire fonctionner le bouton -

6 A vous de jouer !

Q 5. Réalisez une version haute fidélité de votre prototype de projet et gérez les événements pour une partie de l'interface qui paraît essentielle.

7 Créer manuellement un jar exécutable

7.1 Exemple TP 2

La création du jar exécutable se fait avec la commande `jar`, de la façon suivante pour le TP2 :
`jar -cvfm carres.jar manifest.MF tp2/Carres.class tp2/Rectangle.class`

Le fichier manifest.MF précise la classe principale. Il est nécessaire d'ajouter une ligne vide en fin du fichier.

```
Manifest-Version: 1.0
Main-Class: tp2.Carres
Class-Path: .
```

L'exécution se fait avec la ligne suivante :

```
java --module-path /home/public/javafx-sdk-17.0.2/lib
 --add-modules javafx.controls,javafx.fxml  -jar carres.jar
```

La commande suivante permet de vérifier le contenu du jar : `jar tf carres.jar`

7.2 Exemple TP 6

```
Manifest-Version: 1.0
Main-Class: tp6.FXMLdemo
Class-Path: .
```

`jar -cvfm tp6.jar manifest.MF tp6/FXMLdemo.class tp6/MonController.class tp6/interface.fxml`