Interaction Homme-Machine

TP 2 : Mise en forme de nœuds graphiques et introduction à l'architecture MVC

Objectifs :

- Mise en forme des nœuds graphiques

- Utilisation des fichiers FXML pour programmer une interface graphique

- Utilisation de l'architecture Modèle – Vue – Contrôleur (MVC)

EXERCICE 1 : Utilisation du conteneur GridPane

Dans cet exercice vous allez créer une interface graphique permettant de calculer le périmètre d'un rectangle. Cette interface graphique doit être affichée de la manière suivante :

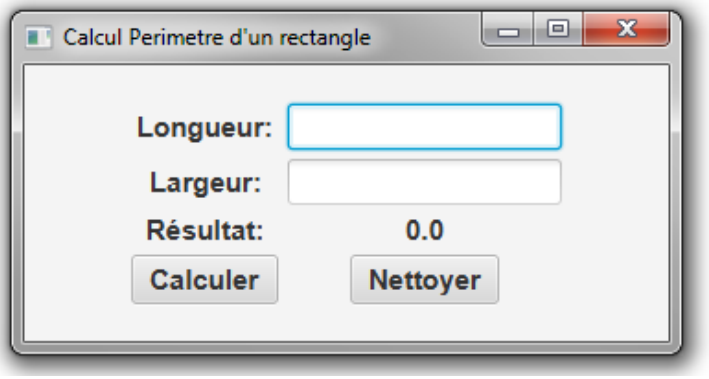


Figure 1 : Interface graphique « Calcul Périmètre d'un rectangle »

Dans cette interface graphique, les nœuds (**Label**, **TextField**, **Button**) sont contenus en utilisant un **GridPane**. Les nœuds graphiques seront positionnés de la manière suivante :

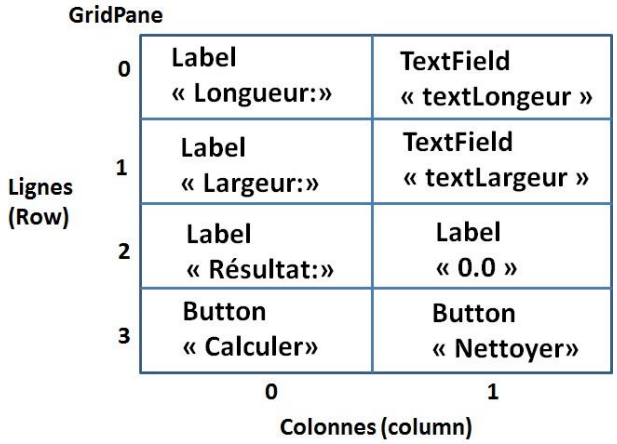


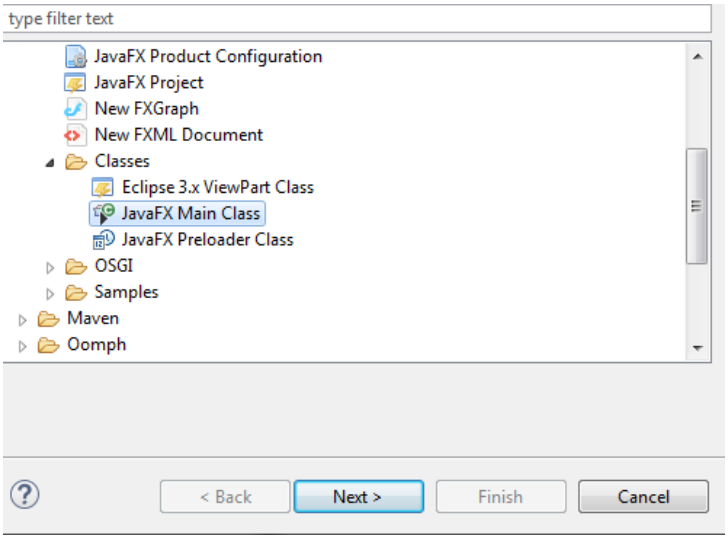
Figure 2 : Nœuds positionnés dans le GridPane

Suivez les indications suivantes pour programmer l'interface graphique de la figure 1 :

1.- Créez un nouveau projet « TP2 ».

2.- Créer une classe principale JavaFX de la manière suivante :

Cliquez droite sue le dossier « src » --> New --> Other…--> JavaFX Main Class



Cliquez **Next,** puis nommez votre classe **PerimetreGUI** et cliquez **Finish**

3.- L'étape précédent générera automatiquement les deux méthodes nécessaires pour programmer une application graphique en JavaFX : méthode *start* et méthode *main*. Dans votre méthode *start*, créez et ajoutez les nœuds graphiques suivants :

a. Créez un nouveau conteneur de type **GridPane**
b. Positionner votre **GridPane** au centre de la scène utilisant la méthode *setAlignement*
c. Définir un écart de 5 pixels entre chaque colonne en utilisant la méthode setHgap
d. Définir un écart de 5 pixels entre chaque ligne en utilisant la méthode setVgap
e. Créez les contrôles utilisateur (**Label**, **TextField**, **Button)** et ajoutez-les dans le **GridPane** en respectant l'ordre spécifié par la figure 2.
f. Mettre en forme la police, le style et la taille du texte de tous les nœuds **Label** et **Button.** La police du texte sera définie comme « Arial », le style comme Bold et la taille sera 15. Utiliser la méthode *setFont*. Dans cette méthode, il faut

passer comme paramètre la mise en forme : (Font.font("Arial", FontWeight.BOLD, 15)

g. Programmer le comportement du bouton « Calculer » en ajoutant un événement *setOnAction* avec le comportement suivant :

```
setOnAction( e -> {
            //Si les TextFields ne sont pas vides
                    //Récuperer la valeur de la longueur
                    //Récuperer la valeur de la largeur
                    //Calculer le périmètre
                    //Afficher le résultat dans le Label valResultat
              });
```

h. Programmer le comportement du bouton « Nettoyer » en ajoutant un événement *setOnAction* avec le comportement suivant :

```
setOnAction( e -> {
              //Effacer la valeur de chaque TextField()
              //Remplacer la valeur de 0.0 dans le Label valResultat
              });
```

i. Centrez tous les contrôles utilisateur (**Label**, **TextField**, **Button**) horizontalement dans chaque cellule en utilisant la méthode *setHalignment*

j. Mettez le titre *« Calcul Périmètre d'un rectangle »* dans votre fenêtre principale (Stage).

k. Ajoutez la scène à votre Stage en utilisant la méthode *setScene*

l. Finalement, affichez votre application JavaFX avec la méthode *show.*

4. Finalement, si toutes les indications sont bien respectées, l'exécution de la classe principale (**PerimetreApp**) doit afficher l'interface graphique montrée dans la figure 1 et les boutons déclencheront les actions correspondantes.

EXERCICE 2 : Architecture Modèle - Vue - Contrôleur

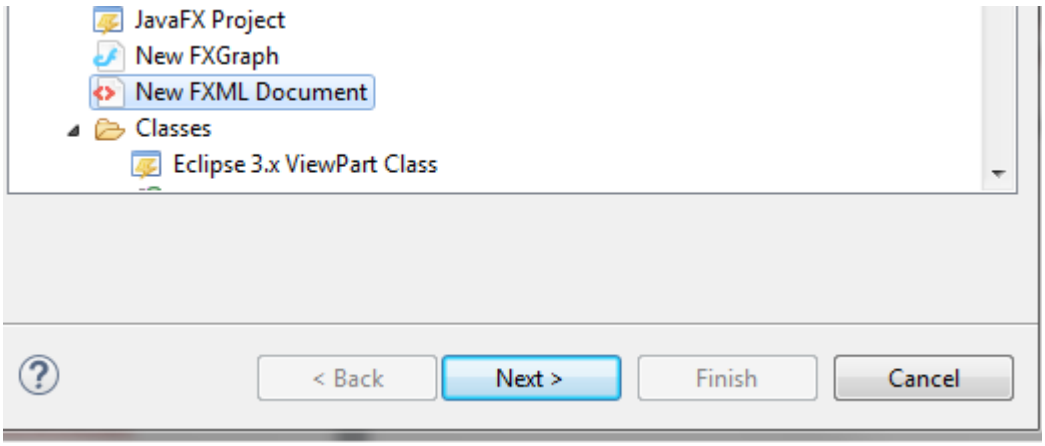
Dans cet exercice vous allez programmer l'application précèdent (« Calcul Périmètre d'un rectangle ») en utilisant l'architecture MVC. L'architecture Modèle-Vue-Contrôleur (MVC) est une façon d'organiser une interface graphique d'un programme. Elle consiste à distinguer trois entités distinctes qui sont, le modèle, la vue et le contrôleur ayant chacun un rôle précis dans l'interface.

- Modèle : gestion et manipulation des données
- Vue : interface graphique utilisateur (entrées et sorties des données)
- Contrôleur : gestion des événements et synchronisation

Le modèle MVC permet une conception plus claire et efficace grâce à la séparation des données de la vue et du contrôleur.

Respectez les indications suivantes afin de transformer votre application précédente vers une application avec une architecture MVC.

1. Créez un nouveau projet TP2_MVC
2. Dans le projet TP2_MVC, créez un package nommé **perimetre**
3. Dans le package, créez les 4 fichiers suivants :
   a. Une classe principale JavaFX nommée **PerimetreApp**
   b. Une classe nommée **PerimetreController**
   c. Une classe nommée **PerimetreModele**
   d. Un document FXML nommé **PerimetreGUI.**
   Cliquez droite sur le package « perimetre » --> New --> Other…--> New FXML Document. Puis, cliquez sur **Next**, nommé votre fichier FXML et cliquez sur **Finish**



4. Dans la méthode *start* de la classe principale (**PerimetreApp**), respectez les indications suivantes :

   a. Ajoutez l'instruction suivante pour charger le fichier FXML

```
Parent root = FXMLLoader.load(getClass().getResource("PerimetreGUI.fxml")
```

   b. Créez une nouveau objet de type **Scène** et ajoutez les nœuds **root** à cette scène. Donnez à votre scène une largeur de 350 et une hauteur de 150.
   c. Mettez le titre *« Calcul Périmètre d'un rectangle »* dans votre fenêtre principale (Stage).
   d. Ajoutez la scène à votre Stage en utilisant la méthode *setScene*
   e. Affichez votre application JavaFX avec la méthode *show*

5. Dans le fichier **PerimetreGUI.fxml**, effacez le code généré automatiquement et copiez le code suivant :

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.net.*?>
<?import javafx.geometry.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.text.*?>

<!--Un GridPane avec un alignement centrée et un écart de taille 5 entre
    chaque colonne et chaque ligne  -->
<GridPane fx:controller="perimetre.PerimetreController"
    xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="5" vgap="5">

       <!--Un Label "Longueur", Colonne = 0, Ligne = 0 -->
       <Label text="Longeur:"
         GridPane.columnIndex="0" GridPane.rowIndex="0"/>

       <!--Ajoutez un TextField avec un identifiant textLongeur,
       Colonne = 1,  Ligne = 0 -->


       <!--Ajoutez un Label "Largeur", Colonne = 0, Ligne = 1 -->


       <!--Un TextField avec un identifiant textLargeur, Colonne = 1, Ligne = 1 -->
       <TextField fx:id="textLargeur"
         GridPane.columnIndex="1" GridPane.rowIndex="1"/>

      <!--Ajoutez un Label "Resultat", Colonne = 0, Ligne = 2 -->


      <!--Ajoutez un Label "0.0" avec un identifiant valResultat,
      Colonne = 1, Ligne = 2 -->


      <!--Ajoutez un Button "Calculer" qui appelera la méthode buttonCalculer,
      Colonne = 0, Ligne = 3 -->


     <!--Un Button "Nettoyer" qui appelera la méthode buttonNettoyer,
    Colonne = 1, Ligne = 3 -->
    <Button text="Nettoyer" onAction="#buttonNettoyer"
      GridPane.columnIndex="1" GridPane.rowIndex="3"/>

</GridPane>
```

Dans le code FXML précédent, les nœuds suivants sont déjà créés dans l'interface graphique : **GridPane**, **Label** (« Longueur »), **TextField** (« textlargeur »), **Button** (« Nettoyer »). Ajoutez les nœuds manquants en respectant les instructions des commentaires (`<!--commentaire -->`). Regardez et comprenez les instructions des nœuds déjà ajoutés pour savoir comment ajouter les nœuds manquants.

Mettez en forme la police, le style et la taille du texte de tous les nœuds **Label** et **Button.** La police du texte sera définie comme « Arial », le style comme Bold et la taille sera 15.

Afin de mettre en forme le texte, ajoutez les instructions suivantes dans le fichier FXML avant la création du première nœud graphique (avant le **Label** « Longueur ») :

```
<fx:define>
  <Font fx:id="FONT" name="Arial Bold" size="15" />
</fx:define>
```

Ensuite, ajoutez à chaque contrôle utilisateur le paramètre `font="$FONT"` qui définit la mise en forme spécifié dans l'instruction précédent.

Finalement, centrez tous les contrôles utilisateur (**Label**, **TextField**, **Button**) horizontalement dans chaque cellule en ajoutant à chaque control utilisateur le paramètre `GridPane.halignment="CENTER".`

6. Dans la classe **PerimetreModele** créez une méthode calculPerimetre() qui recevra comme paramètres une valeur flottante longueur et une valeur flottante largeur. La méthode retournera le calcul du périmètre à partir de ces deux paramètres.

7. Dans la classe **PerimetreController** respectez les consignes suivantes :

   a. Déclarez une instance de la classe **PerimetreModele** et initialisez cette instance dans le constructeur de la classe **PerimetreController**
   b. Pour chaque identifiant (`fx:id`) défini dans le fichier **PerimetreGUI.fxml,** déclarez une variable qui accèdera au nœud graphique ajouté. Chaque variable référencée sera précédée d'une référence `@FXML` avant la déclaration.

      Par exemple, pour `<TextField fx:id="textLargeur"` dans le fichier **PerimetreGUI.fxml,** déclarez la variable de la manière suivante dans la classe **PerimetreController**

      ```
      @FXML
      private TextField textLargeur;
      ```

      Déclarez les références pour `textLongeur` et `valResultat.`

   c. Pour chaque action (`onAction`) défini dans le fichier **PerimetreGUI.fxml,** programmez une méthode qui déclenchera un événement correspondant. Chaque méthode référencée sera précédée d'une référence `@FXML` avant la déclaration.

      Par exemple, pour `onAction="#buttonNettoyer"` dans le fichier **PerimetreGUI.fxml,** déclarez la méthode de la manière suivante dans la classe **PerimetreController**

      ```
      @FXML
      private void buttonNettoyer(ActionEvent event) {
             //Effacer la valeur de chaque TextField()
             //Remplacer la valeur de 0.0 dans le Label valResultat
         }
      ```

      Programmez la méthode correspondante à l'action du Button « Calculer ». Dans cette méthode, appelez à la méthode *calculPerimetre*() défini dans la classe **PerimetreModele.**

8. Finalement, si toutes les indications sont bien respectées, l'exécution de la classe principale (**PerimetreApp**) doit afficher une interface graphique similaire à celle montrée dans la figure 1 et les boutons déclencheront les actions correspondantes.