
M2105: INTERFACES HOMME-MACHINE
Feuille de TP n°1
Les composants JavaFX

L'objectif de cette feuille est de manipuler les différents composants de l'API graphique `javafx`. Au cours de cette feuille nous allons réaliser un formulaire de saisie d'informations concernant une personne. Ce formulaire sera complètement passif, ce premier TP va simplement consister à agencer les composants dans une fenêtre.

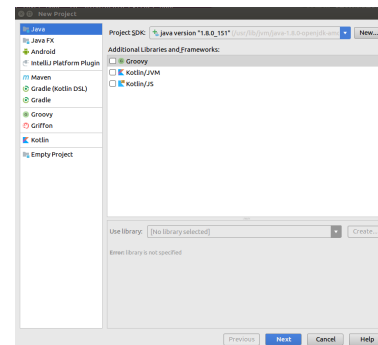
Exercice 1 *IntelliJ*

Afin de faciliter l'utilisation de l'API `javafx`, nous allons travailler dans un environnement de développement intégré (EDI) IntelliJ. Cet environnement est assez complexe mais notre objectif est d'utiliser uniquement les fonctionnalités les plus simples.

Vous pouvez lancer IntelliJ à partir des menus de votre environnement graphique. Lors du premier lancement, il vous est demandé de finaliser l'installation de IntelliJ. Nous vous conseillons de prendre les options par défaut (skip remaining steps...). Ensuite, l'écran de bienvenue apparaît.



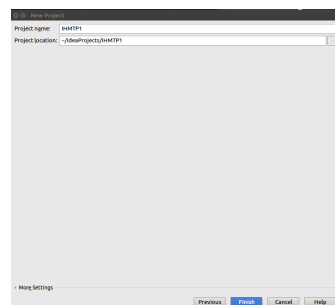
Écran de bienvenue



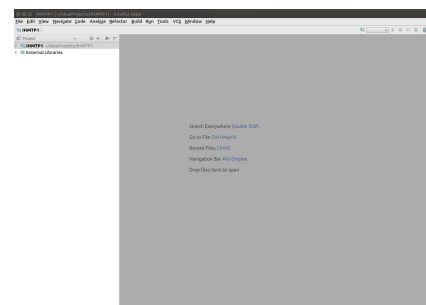
Création du projet

Il suffit de cliquer sur l'icône *Create New Project* et choisir sur l'écran suivant *Java* (et pas *Java FX* pour nous éviter d'avoir à supprimer des fichiers).

En cliquant sur *Next* deux fois on arrive sur l'écran suivant où vous saisissez le nom de votre projet *IHMTP1* puis cliquez sur *Finish*.



Choix du nom de projet

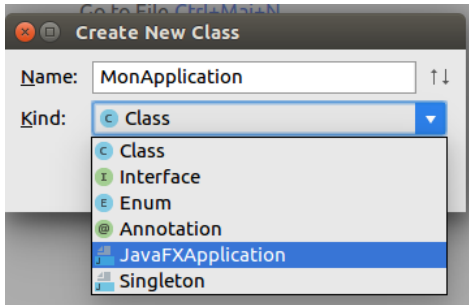


Projet créé

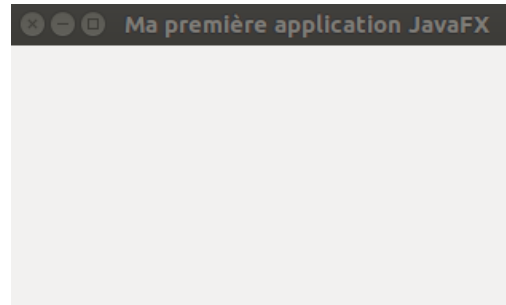
Après avoir lu (ou non) l'astuce, vous devez avoir une fenêtre comme ci-dessus. Voila ! Votre projet est créé ! Il ne contient encore rien mais il est créé !

Exercice 2 Créer des classes

Nous allons maintenant créer la classe qui servira de test et contiendra la méthode `main`. Cette classe hérite de la classe `javafx.application.Application`.
Créer la classe *MonApplication*. Après avoir développé l'item `IHMTP1` dans l'explorateur, on va cliquer avec le bouton de droite de la souris sur l'item `src`, puis choisir l'option *New* et choisir l'option *Java Class*. Sur l'écran suivant donner comme nom `MonApplication` et choisissez l'option *JavaFX Application*.



Menu création d'une classe



Ma première application

Vous obtenez ainsi le squelette de votre application.

```
import javafx.application.Application;
import javafx.stage.Stage;
public class MonApplication extends Application {
    public static void main(String[] args) {
        launch(args);
    }
    @Override
    public void start(Stage primaryStage) {
    }
}
```

- La méthode `start(Stage primaryStage)` est la méthode appelée lors du lancement de l'application. C'est une méthode de la classe `Application` que nous devons implémenter (c'est le sens de l'assertion `@Override`. C'est dans cette méthode que nous allons créer la scène graphique. Le paramètre `primaryStage` est le *théâtre* principal de l'application (là où on va placer la scène). Dans un premier temps nous allons juste mettre un titre à la fenêtre et l'afficher (voir ci-dessous).
- La méthode `main` va simplement appeler la méthode `launch` de la classe `Application` qui lance l'application.

```
@Override
public void start(Stage primaryStage) {
    primaryStage.setTitle("Ma première application JavaFX");
    primaryStage.show();
}
```

Maintenant vous pouvez cliquer sur l'option *Run 'MonApplication'* du menu *run* pour tester votre application. Cette application mériterait d'être un peu enrichie!

Exercice 3 Le graphe de scène

Pour créer notre scène nous allons définir une méthode `creerFormulaire()` dans notre classe qui aura comme résultat une `Scene` qui contiendra le graphe de scène permettant de gérer un formulaire. Le groupe principal de cette scène sera un `FlowPane` qui permet de ranger les

éléments graphiques les uns à la suite des autres. Notez que pour ajouter un élément dans un `FlowPane` il faut passer par `getChildren()`.

Dans la méthode `start()`, on place la scène créée par notre méthode `creerFormulaire()` dans le *théâtre* de l'application.

```
public Scene creerFormulaire(){
    Label lNom= new Label("Nom :");
    TextField tNom= new TextField();
    FlowPane panel= new FlowPane();
    panel.getChildren().add(lNom);
    panel.getChildren().add(tNom);
    return new Scene(panel,500,100);
}

@Override
public void start(Stage primaryStage) {
    primaryStage.setTitle("Ma première application JavaFX");
    primaryStage.setScene(creerFormulaire());
    primaryStage.show();
}
```

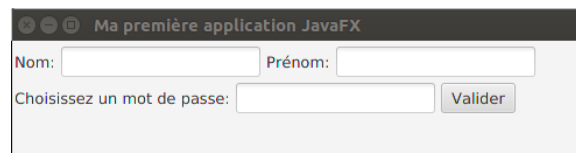
Un `FlowPane` met ses éléments graphiques les uns à la suite des autres et passe à la ligne en fonction de la place qu'il a. Vous pouvez redimensionner la fenêtre en cours d'exécution pour voir son comportement. On peut remarquer que les objets graphiques sont collés les uns aux autres et aussi collés aux bords de la fenêtres. Pour modifier ce comportement `FlowPane` possède plusieurs méthodes :

- `setVgap(int)` qui détermine l'espacement entre deux lignes,
- `setHgap(int)` qui détermine l'espacement entre deux objets sur une même ligne,
- `setPadding(Insets)` qui détermine l'espacement par rapport aux bords de la fenêtre. Par exemple `setPadding(new Insets(10,5,6,11))` signifie que l'on souhaite un espacement de 10 par rapport au haut de la fenêtre de 5 par rapport au bord droit, de 6 par rapport au bas et de 11 par rapport au bord gauche.

Ajoutez dans votre méthode `creerFormulaire()` les instructions nécessaires pour obtenir un affichage plus aéré.

Exercice 4 Une scène plus complète

Vous allez maintenant compléter la méthode `creerFormulaire()` afin d'obtenir un affichage comme celui présenté ci-dessous. Les champs de saisie des mots de passe sont des `PasswordField` et les boutons des `Button`.



On peut remarquer qu'il est difficile d'avoir une belle présentation de ce formulaire en utilisant un `FlowPane`.

Notez bien que pendant toute la feuille de TP, les formulaires que nous allons produire ne feront rien de particulier. On ne s'intéresse pour l'instant qu'à la partie présentation.

Exercice 5 Agencement d'une scène

L'agencement d'une scène est géré par des *Layout Pane*. Ces objets prennent en compte automatiquement les redimensionnements de la fenêtre. JavaFX propose un certain nombre de layouts. Nous allons en tester quelques uns dans cet exercice.

1. **FlowPane** est celui que nous avons vu dans l'exercice précédent. Lors de la construction d'un **FlowPane** on peut indiquer son orientation (c-à-d est-ce qu'on range les objets graphique en lignes ou en colonne). **Orientation.HORIZONTAL** est l'orientation par défaut, sinon on peut utiliser **Orientation.VERTICAL**.
2. **GridPane** positionne les composants suivant une grille dont le nombre de lignes et de colonnes est défini dynamiquement au cours des ajouts. Les ajouts dans un **GridPane** se font grâce la méthode **add()** comme dans l'exemple ci-dessous où on ajoute un label en colonne 2, ligne 1 de la grille.

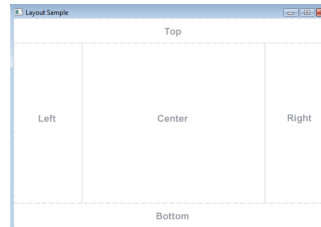
```
GridPane p =new GridPane();  
p.add(lNom,2,1);
```

Créer une méthode **creerFormulaireGrid()** qui permet de créer une scène dont le layout est une grille et qui s'affiche comme ci-dessous.



Noter que l'on peut rendre le bord des cases visibles grâce à la méthode **setGridLinesVisible(true)**.

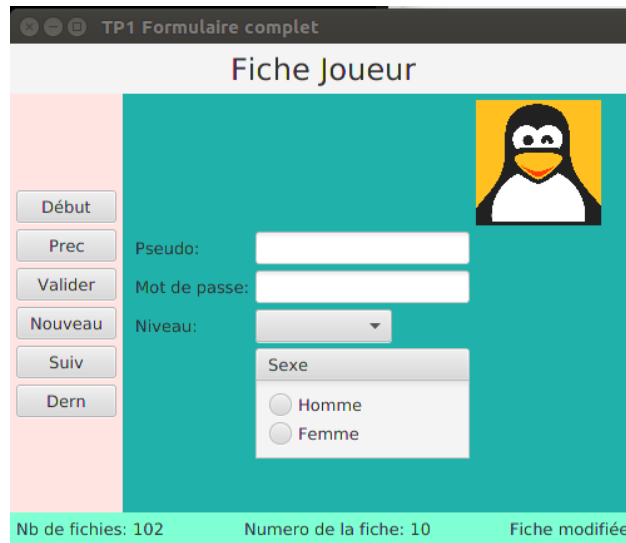
3. **HBox** et **VBox** permettent respectivement d'arranger les objets graphiques en une seule ligne ou une seule colonne. L'ajout des objets se fait comme pour le **FlowPane**.
4. **BorderPane** permet de gérer l'espace en 5 zones : le *top*, le *bottom*, le *left*, le *right* et le *center*, comme représenté ci dessous.



Cela permet d'organiser son affichage en différentes zones. La plupart du temps les différentes parties de l'écran sont elles-même remplies par des **Pane**. Les méthodes permettant d'affecter un contenu à chaque zone de l'écran sont **setTop()**, **setBottom()**, etc.

Exercice 6 Structurer une fenêtre

Dans cet exercice, on vous demande de créer une application JavaFX qui permet de créer le formulaire ci-dessous



Voici quelques indications :

- La scene est organisée par un **BorderPane**.
- La partie basse est aussi organisée par un **BorderPane**.
- La partie gauche est organisée par un **VBox**.
- La partie centrale est organisée par un **GridPane**.
- La méthode `prefWidth()` permet de contrôler la largeur des boutons.
- Le code ci-dessous permet d'afficher une image. Attention il faut copier l'image dans le répertoire `out/production/MonProjet` de votre projet.

```
ImageView img = new ImageView("/monimage.png");  
img.setFitWidth(100); // retaille l'image  
img.setPreserveRatio(true); // préserve le ratio hauteur/largeur
```