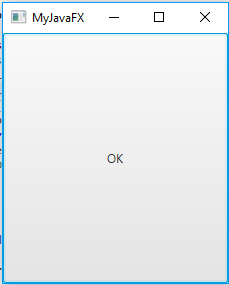
**TD1**

Initiation à JavaFX

Commençons par la construction d’une interface très simple : une simple contenant un bouton, dessinée ci-dessous :



Le code qui permet d’afficher cette interface est le suivant :

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.stage.Stage;

public class MyJavaFX extends Application {

@Override // Override the start method in the Application class

public void start(Stage primaryStage) {

// Create a button and place it in the scene

Button btOK = new Button("OK");

Scene scene = new Scene(btOK, 200, 250);

primaryStage.setTitle("MyJavaFX"); // Set the stage title

primaryStage.setScene(scene); // Place the scene in the stage

primaryStage.show(); // Display the stage

}

/\*\*

\* The main method is only needed for the IDE with limited

\* JavaFX support. Not needed for running from the command line.

\*/

public static void main(String[] args) {

launch(args);

}

}

Ce programme permet d’afficher une seule scène. On peut afficher plusieurs scènes :

**Exercice 2** : affichage de 2 scènes.

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.stage.Stage;

public class MultipleStageDemo extends Application {

public void start(Stage primaryStage) {

// Create a scene and place a button in the scene

Scene scene = new Scene(new Button("OK"), 200, 250);

primaryStage.setTitle("MyJavaFX"); // Set the stage title

primaryStage.setScene(scene); // Place the scene in the stage

primaryStage.show(); // Display the stage

Stage stage = new Stage(); // Create a new stage

stage.setTitle("Second Stage"); // Set the stage title

// Set a scene with a button in the stage

stage.setScene(new Scene(new Button("New Stage"), 100, 100));

stage.show(); // Display the stage

}

/\*\*

\* The main method is only needed for the IDE with limited

\* JavaFX support. Not needed for running from the command line.

\*/

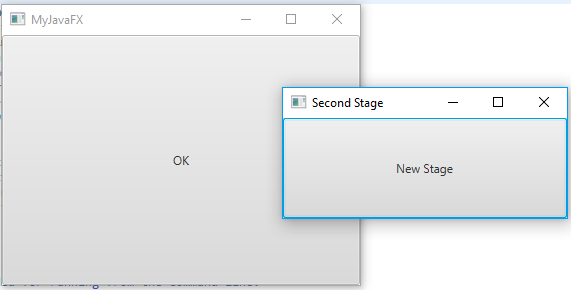
public static void main(String[] args) {

launch(args);

}

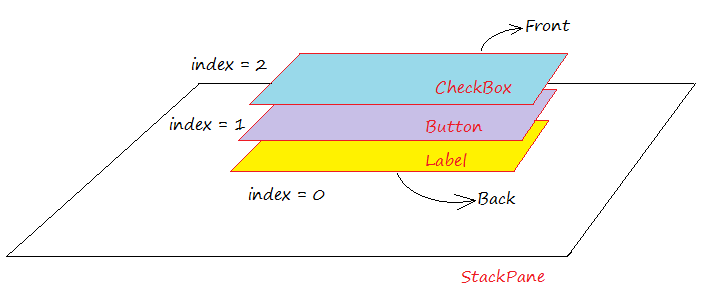
}

On obtient l’affichage suivant :



Il existe plusieurs types de pane (sous-classes de la classe ***Pane***):

1. ***StackPane*** : Les éléments sont « empilés » les uns sur les autres



**Exercice 3**

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.StackPane;

import javafx.stage.Stage;

public class ButtonInPane extends Application {

public void start(Stage primaryStage) {

// Create a scene and place a button in the scene

StackPane pane = new StackPane();

pane.getChildren().add(new Button("OK"));

Scene scene = new Scene(pane, 400, 100);

primaryStage.setTitle("Button in a pane"); // Set the stage title

primaryStage.setScene(scene); // Place the scene in the stage

primaryStage.show(); // Display the stage

}

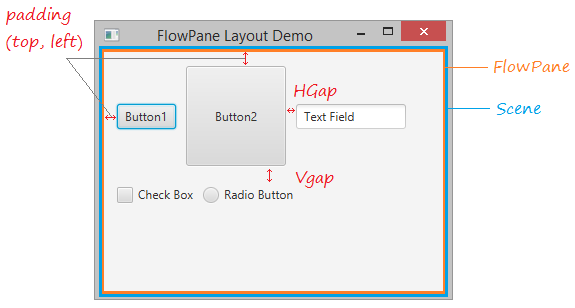
public static void main(String[] args) {

launch(args);

}

}

1. ***FlowPane***: les éléments sont placés de gauche à droite et de haut en bas



**Exercice 4**:

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.layout.FlowPane;

import javafx.stage.Stage;

public class ShowFlowPane extends Application {

public void start(Stage primaryStage) {

// Create a pane and set its properties

FlowPane pane = new FlowPane();

pane.setPadding(new Insets(11, 12, 13, 14));

pane.setHgap(5);

pane.setVgap(5);

// Place nodes in the pane

pane.getChildren().addAll(new Label("First Name:"),

new TextField(), new Label("MI:"));

TextField tfMi = new TextField();

tfMi.setPrefColumnCount(1);

pane.getChildren().addAll(tfMi, new Label("Last Name:"),

new TextField());

// Create a scene and place it in the stage

Scene scene = new Scene(pane, 200, 250);

primaryStage.setTitle("ShowFlowPane"); // Set the stage title

primaryStage.setScene(scene); // Place the scene in the stage

primaryStage.show(); // Display the stage

}

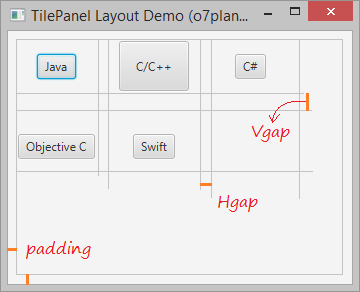
public static void main(String[] args) {

launch(args);

}

}

1. ***TilePane****:* les éléments sont alignés de gauche à droite et de haut en bas avec des espaces égaux.



**Exercice 5** :

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.TilePane;

import javafx.stage.Stage;

public class TilePaneDemo extends Application {

    @Override

    public void start(Stage primaryStage) throws Exception {

        TilePane root = new TilePane();

        root.setPadding(new Insets(10,10,10,10));

        root.setHgap(20);

        root.setVgap(30);

        Button button = new Button("Java");

        root.getChildren().add(button);

        // Short Button

        Button button1 = new Button("C/C++");

        button1.setPrefSize(70, 50);

        root.getChildren().add(button1);

        // Short Button

        Button button2 = new Button("C#");

        root.getChildren().add(button2);

        // Button

        Button longButton3 = new Button("Objective C");

        root.getChildren().add(longButton3);

        // Button

        Button button4 = new Button("Swift");

        root.getChildren().add(button4);

        Scene scene = new Scene(root, 500, 300);

        primaryStage.setTitle("TilePanel Layout Demo (o7planning.org)");

        primaryStage.setScene(scene);

        primaryStage.show();

    }

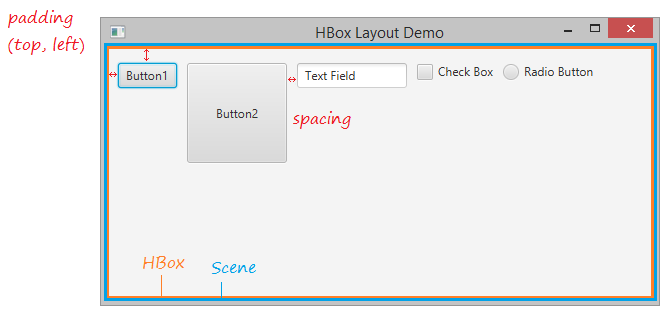
    public static void main(String[] args) {

        launch(args);

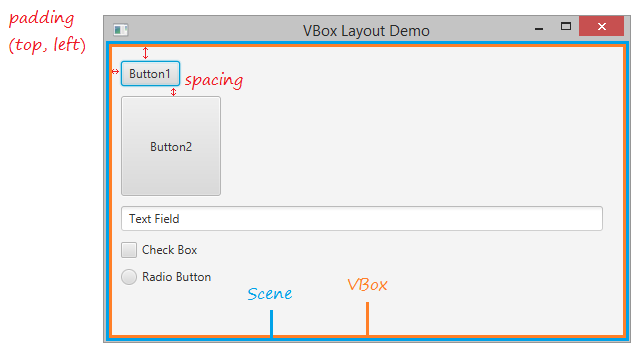
    }

}

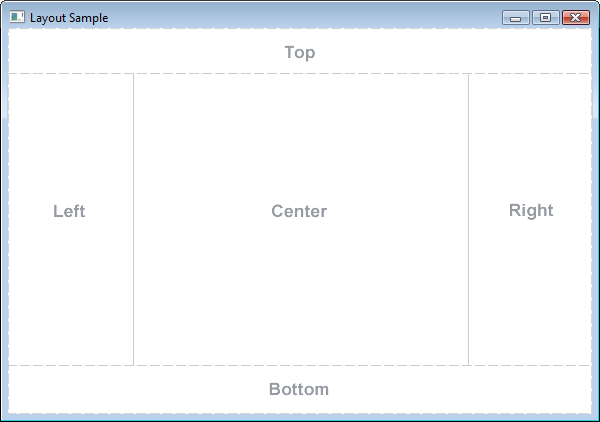
1. ***HBox et VBox****:*
   1. *HBox*: Les éléments sont disposés dans un container horizontal de gauche à droite



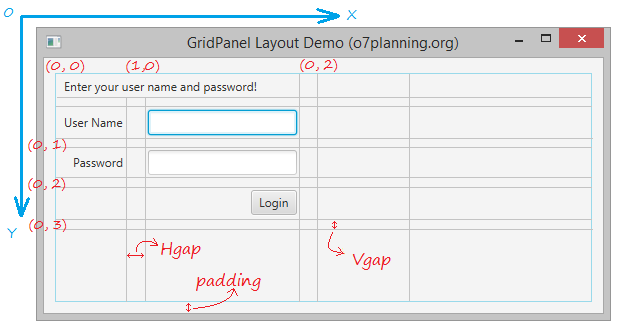
* 1. *Vbox :* Les éléments sont disposés dans un container vertical de haut en bas



1. ***BorderPane****:* Les éléments sont disposés selon 4 directions et un centre.



1. ***GridPane****:* les éléments sont disposés selon une grille avec des indices (colonne, ligne).



**Remarque :**

Tous les panes précédents peuvent être imbriqués les uns dans les autres pour créer des interfaces complexes et variées.

Exemple 6 utilisant un *BorderPane* avec *HBox* et *VBox* :

import javafx.application.Application;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

public class ShowHBoxVBox extends Application {

@Override // Override the start method in the Application class

public void start(Stage primaryStage) {

// Create a border pane

BorderPane pane = new BorderPane();

// Place nodes in the pane

pane.setTop(getHBox());

pane.setLeft(getVBox());

// Create a scene and place it in the stage

Scene scene = new Scene(pane);

primaryStage.setTitle("ShowHBoxVBox"); // Set the stage title

primaryStage.setScene(scene); // Place the scene in the stage

primaryStage.show(); // Display the stage

}

private HBox getHBox() {

HBox hBox = new HBox(15);

hBox.setPadding(new Insets(15, 15, 15, 15));

hBox.setStyle("-fx-background-color: gold");

hBox.getChildren().add(new Button("Computer Science"));

hBox.getChildren().add(new Button("Chemistry"));

// ImageView imageView = new ImageView(new Image("image/us.gif"));

// hBox.getChildren().add(imageView);

return hBox;

}

private VBox getVBox() {

VBox vBox = new VBox(15);

vBox.setPadding(new Insets(15, 5, 5, 5));

vBox.getChildren().add(new Label("Courses"));

Label[] courses = {new Label("CSCI 1301"), new Label("CSCI 1302"),

new Label("CSCI 2410"), new Label("CSCI 3720")};

for (Label course: courses) {

VBox.setMargin(course, new Insets(0, 0, 0, 15));

vBox.getChildren().add(course);

}

return vBox;

}

/\*\*

\* The main method is only needed for the IDE with limited

\* JavaFX support. Not needed for running from the command line.

\*/

public static void main(String[] args) {

launch(args);

}

}

**Exercice 7** avec un *GridPane* :

import javafx.application.Application;

import javafx.geometry.HPos;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.TextField;

import javafx.scene.layout.GridPane;

import javafx.stage.Stage;

public class ShowGridPane extends Application {

@Override // Override the start method in the Application class

public void start(Stage primaryStage) {

// Create a pane and set its properties

GridPane pane = new GridPane();

pane.setAlignment(Pos.CENTER); //affiche le pane en position centree H et V dans la scene

pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));

pane.setHgap(5.5);

pane.setVgap(5.5);

// Place nodes in the pane

pane.add(new Label("First Name:"), 0, 0);

pane.add(new TextField(), 1, 0);

pane.add(new Label("MI:"), 0, 1);

pane.add(new TextField(), 1, 1);

pane.add(new Label("Last Name:"), 0, 2);

pane.add(new TextField(), 1, 2);

Button btAdd = new Button("Add Name");

pane.add(btAdd, 1, 3);

GridPane.setHalignment(btAdd, HPos.RIGHT);

// Create a scene and place it in the stage

Scene scene = new Scene(pane);

primaryStage.setTitle("ShowGridPane"); // Set the stage title

primaryStage.setScene(scene); // Place the scene in the stage

primaryStage.show(); // Display the stage

}

/\*\*

\* The main method is only needed for the IDE with limited

\* JavaFX support. Not needed for running from the command line.

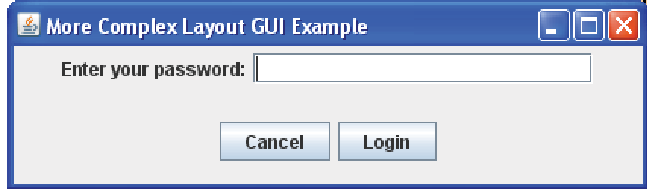
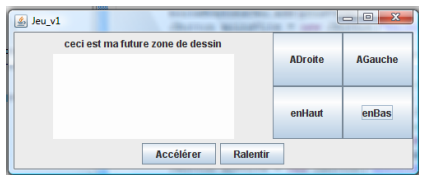
\*/

public static void main(String[] args) {

launch(args);

}

**Exercice 8**: En utilisant les différents panes, écrire les programmes permettant d’obtenir les interfaces suivantes :

1. 
2. 
3. 