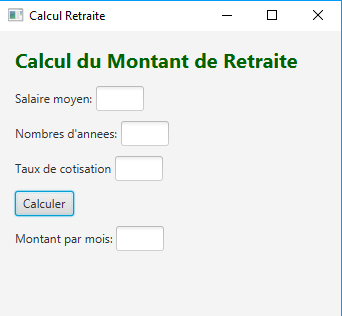
**TD3- partie 1**

Architecture MVC

Dans ce TP, nous allons étudier un exemple d'application pour illustrer différentes manières de réaliser le découpage du code en respectant les principes de l'architecture MVC en exploitant plusieurs des techniques pour réaliser le contrôleur (classes EventHandler, expressions lambda, binding de propriétés, …).

L'application *Retraite* calcule de façon simplifiée le montant de la retraite mensuelle d’une personne, en fonction du montant de son salaire moyen, du nombre d’années cotisées et du taux de cotisation.



**Travail demandé :**

Ecrire les classe correspondants au modèle, à la vue et au contrôleur de cette application en proposant 3 variantes pour le contrôleur :

* Variante 1 : Avec un contrôleur réalisé sous forme de classe 'ordinaire'
* Variante 2 : Avec un contrôleur réalisé sous forme d'expression lambda
* Variante 3 : Avec un contrôleur réalisé sous forme de liaisons de entre les données d'entrée (saisies par l'utilisateur) et les données de sortie (calculées)

En vous aidant des extraits de code (des éléments importants) suivants :

1. **Modèle de l'application**

• Interface (java) du modèle

• Classe implémentant cette interface

public interface IRetraiteModel {

void setSalaire(double amount);

void setAnnees(int ans);

void setTaux(Double ta);

double getMontant();

}

//---------------------------------------------------

**// Classe model**

**public** **class** RetraiteModel **implements** IRetraiteModel {

**private** **double** salaire = 0.0;

**private** **int** annees = 0;

**private** **double** taux = 0.0;

//---------------------------------------------------

**public** RetraiteModel() {

}

//----------------------------------------------------

@Override

**public** **void** setSalaire(**double** amount) {

**if** (amount < 0) **throw** **new** IllegalArgumentException("Amount < 0");

**this**.salaire = amount;

}

//-------------------------------------------------------------------------

@Override

**public** **void** setAnnees(**int** ans) {

**if** (ans < 0) **throw** **new** IllegalArgumentException("Annees < 0");

**this**.annees = ans;

}

//------------------------------------------------------------------------

@Override

**public** **void** setTaux(Double ta) {

**if** (ta <= 0) **throw** **new** IllegalArgumentException("Taux <= 0");

**this**.taux = ta;

}

//------------------------------------------------------------------------

@Override

**public** **double** getMontant() {

**return** (salaire\*taux + (salaire\*annees)/12);

}

}

1. **Variante 1 du controleur : classe externe**

**import** javafx.event.ActionEvent;

**import** javafx.event.EventHandler;

**public** **class** RetraiteController **implements** EventHandler<ActionEvent> {

**private** RetraiteModel model; //model class

**private** ViewTP3 view; //view class

//-------------------------------------------------------------------------

// Constructor receives model and view references

//-------------------------------------------------------------------------

**public** RetraiteController(RetraiteModel model, ViewTP3 view) {

**this**.model = model;

**this**.view = view;

}

//------------------------------------------------------------------------

// Method executed when 'Calculate' button is pressed

//------------------------------------------------------------------------

@Override

**public** **void** handle(ActionEvent event) {

//--- Get values from view (check for errors) and update model data

**try** {

**double** sal = view.getSalaireValue();

**int** ans = view.getAnneesValue();

**double** ta = view.getTauxValue();

model.setSalaire(sal);

model.setAnnees(ans);

model.setTaux(ta);

}

**catch** (IllegalStateException e) { // Errors in some input values

**return**;

}

//--- Update view with values from model output data

view.updateMontant(model.getMontant());

}

}

**//Vue**

**package TP3;**

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.\*;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.stage.Stage;

public class ViewTP3 extends Application {

private TextField tfSalaire;

private TextField tfAnnees;

private TextField tfTaux;

private TextField tfResult;

private Button cal;

private RetraiteModel model;

@Override

// Override the start method in the Application class

public void start(Stage primaryStage) {

VBox pane = new VBox();

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

pane.setSpacing(10);

FlowPane p1 = new FlowPane();

tfSalaire = new TextField();

p1.getChildren().addAll(new Label("Salaire moyen: "), tfSalaire);

FlowPane p2 = new FlowPane();

tfAnnees = new TextField();

p2.getChildren().addAll(new Label("Nombres d'annees: "), tfAnnees);

FlowPane p3 = new FlowPane();

tfTaux = new TextField();

p3.getChildren().addAll(new Label("Taux de cotisation "), tfTaux);

//-----------------------------------------

cal = new Button("Calculer");

model= new RetraiteModel();

RetraiteController rc = new RetraiteController(model, this);

cal.addEventHandler(ActionEvent.ACTION, rc);

//-----------------------------------------

FlowPane p4 = new FlowPane();

tfResult = new TextField();

p4.getChildren().addAll(new Label("Montant par mois: "), tfResult);

tfSalaire.setPrefColumnCount(6);

tfAnnees.setPrefColumnCount(6);

tfTaux.setPrefColumnCount(6);

tfResult.setPrefColumnCount(6);

Label title = new Label("Calcul du Montant de Retraite");

title.setFont(Font.font("System", FontWeight.BOLD, 20));

title.setTextFill(Color.DARKGREEN);

pane.getChildren().addAll(title, p1, p2, p3, cal, p4);

// Create a scene and place it in the stage

Scene scene = new Scene(pane, 350, 300);

primaryStage.setTitle("Calcul Retraite"); // 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);

}

public double getSalaireValue() {

// TODO Auto-generated method stub

return Double.parseDouble(tfSalaire.getText());

}

public int getAnneesValue() {

// TODO Auto-generated method stub

return Integer.parseInt(tfAnnees.getText());

}

public double getTauxValue() {

// TODO Auto-generated method stub

return Double.parseDouble(tfTaux.getText());

}

public void updateMontant(Double m) {

// TODO Auto-generated method stub

tfResult.setText( m + " ");

}

}

1. **Variante 2 du contrôleur : expression lambda**

Dans cette variante, on n’a pas de classe Controlleur, car le code du controleur est inseree directement dans la vue. La classe Model est inchangee**.**

**// variante2 controleur dans la vue avec lambda expression**

package TP3;

import javafx.application.Application;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.\*;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.FontWeight;

import javafx.stage.Stage;

public class ViewLambda extends Application{

private TextField tfSalaire;

private TextField tfAnnees;

private TextField tfTaux;

private TextField tfResult;

private Button cal;

private RetraiteModel model;

@Override

// Override the start method in the Application class

public void start(Stage primaryStage) {

VBox pane = new VBox();

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

pane.setSpacing(10);

FlowPane p1 = new FlowPane();

tfSalaire = new TextField();

p1.getChildren().addAll(new Label("Salaire moyen: "), tfSalaire);

FlowPane p2 = new FlowPane();

tfAnnees = new TextField();

p2.getChildren().addAll(new Label("Nombres d'annees: "), tfAnnees);

FlowPane p3 = new FlowPane();

tfTaux = new TextField();

p3.getChildren().addAll(new Label("Taux de cotisation "), tfTaux);

//-----------------------------------------

cal = new Button("Calculer");

model= new RetraiteModel();

cal.setOnAction(ev-> {

try {

model.setSalaire(getSalaireValue());

model.setAnnees(getAnneesValue());

model.setTaux(getTauxValue());

}

catch (IllegalStateException e) { // Errors in some input values

return;

}

//--- Update view with values from model output data

updateMontant(model.getMontant());

});

/\* le reste du code est le meme que le code de la vue dans la 1ere variante--------------------\*/

1. **Variante 3 du contrôleur : par data binding**

A preparer pour la prochaine seance.

Exemple de data binding :

package TP3;

import javafx.application.Application;

import javafx.beans.property.\*;

import javafx.geometry.Insets;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

public class User extends Application {

private TextField userInput = new TextField();

private Label firstLabel = new Label("Welcome to my site ");

private Label secondLabel = new Label();

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

public void start(Stage primaryStage) {

//

VBox pane = new VBox();

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

pane.getChildren().add(userInput);

HBox hBox = new HBox(15);

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

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

secondLabel.textProperty().bind(userInput.textProperty());

hBox.getChildren().addAll(firstLabel,secondLabel);

pane.getChildren().add(hBox);

// Create a scene and place it in the stage

Scene scene = new Scene(pane,300,300);

primaryStage.setTitle("Binding example"); // 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);

}

}