TP1 Java fx

Exercice 1

Première fenêtre

Nous allons voir comment créer une fenêtre avec un code très simple :

On va créer une classe intitulé **PremiereFenetre**

```
public class PremiereFenetre extends Application {
    @Override
    public void start(Stage primaryStage) {
        Group root = new Group();

        Scene scene = new Scene(root, 300, 250, Color.WHITE);

        primaryStage.setTitle("Ma première fenêtre");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

✓ La classe `Application`

```
public class PremiereFenetre extends Application {
```

La classe principale **`PremiereFenetr`**`e` hérite de la classe **`Application`**. La classe `Application` s'occupe de gérer le cycle de vie de notre fenêtre, elle permet de lancer la fenêtre dans un autre thread que le thread principal.

✓ La méthode `start`

```
@Override
public void start(Stage primaryStage) {
```

La méthode `start` est le point d'entrée de toutes les applications en JavaFX, elle fournit un `Stage` (nommé `primaryStage`) qui représente la fenêtre de notre application.

✓ La scène

```
Group root = new Group();

Scene scene = new Scene(root, 300, 250, Color.WHITE);
```

Pour pouvoir afficher quelque chose dans notre **`Stage`**, il nous faut ce qu'on appelle une **`Scene`**. Cette scène devra contenir ce qu'on appelle un **`Parent`**. Un `Parent` est un objet qui peut

contenir des enfants, c'est une sorte de conteneur, ici notre `Parent` se nomme **`root`**. De plus, nous donnons des dimensions à notre `Scene` (ici 300 de largeur pour 250 de hauteur) ainsi qu'une couleur d'arrière-plan (ici blanc).

La classe **Group** est un container de nœuds de JavaFX, elle permet de stocker facilement plusieurs nœuds de JavaFX. Group est une sous-classe de la classe Parent qui est elle-même une sous-classe de la classe Node (éléments de base de JavaFX)

C'est le prototype du constructeur de notre scène (un des nombreux) :

```
Scene(parent, largeur, hauteur, couleur de fond)
```

✓ Le stage

```
primaryStage.setTitle("Ma première fenêtre");
primaryStage.setScene(scene);
primaryStage.show();
```

✓ La méthode `main`

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

La fameuse méthode `main` se contente de lancer l'application avec `launch` (en passant ses arguments). Une fois l'application lancée, la méthode `start` sera appelée.

Récapitulatif

- Notre classe principale doit hériter de la classe `Application`
- La méthode `start` sera appelée par l'application quand elle sera lancée (méthode `launch`)
- Notre fenêtre est représentée par un objet `Stage` qui contient un objet `Scene` qui contient un objet `Parent` qui contient plusieurs objets `Node`
- Le `Stage` représente la fenêtre (on pourra par exemple lui changer son titre)
- La `Scene` représente le contenu du `Stage` (on pourra par exemple lui donner une taille ainsi qu'une couleur de fond)
- La `Scene` contient un objet `Parent` qui lui, va contenir les composants de notre fenêtre (par exemple des boutons)

Dessin de forme géométrique

Pour ajouter un ou plusieurs objets à notre objet de type Group, vous pouvez procéder de la manière suivante:

ObservableList<Node> liste = groupe.getChildren();

liste.add(ligne);

Nous récupérons un objet de type ObservableList qui contient tous les nœuds enfants de l'objet groupe. Pour rajouter un élément, il suffit de l'ajouter avec la méthode add (c'est en fait une ArrayList).

Pour ajouter un nœud

```
root.getChildren().add(noeud);
```

a) Ajouter une ligne, un rectangle et un cercle

Sachant que les constructeurs des différentes formes sont :

Rectangle `(double x, double y, double width, double height)`

Line `(double startX, double startY, double endX, double endY)`

Circle `(double centerX, double centerY, double radius)`

Exercice 2

Création d'un formulaire d'authentification

Premièrement Vous créez une class ***Welcomeform*** dans un projet javafx

```
public class Welcomeform e extends Application {

    @Override
public void start(Stage primaryStage) {
        primaryStage.setTitle("JavaFX Welcome");

        primaryStage.show();
    }
```

Création de GridPane Layout

Dans le formulaire de connexion, on va utiliser un layout GridPane qui permet de créer une grille flexible de lignes et de colonnes dans laquelle on va déposer les contrôles. Vous pouvez placer des contrôles dans n'importe quelle cellule de la grille et vous pouvez faire en sorte que les contrôles s'étendent sur les cellules selon vos besoins.
Ajouter ce code dans la méthode **start** avant la ligne **primaryStage.show();**

```
GridPane grid = new GridPane();
grid.setAlignment(Pos.CENTER);
grid.setHgap(10);
grid.setVgap(10);
grid.setPadding(new Insets(25, 25, 25, 25));

Scene scene = new Scene(grid, 300, 275);
primaryStage.setScene(scene);
```

La propriété d'alignement modifie la position par défaut de la grille du haut à gauche de la scène au centre

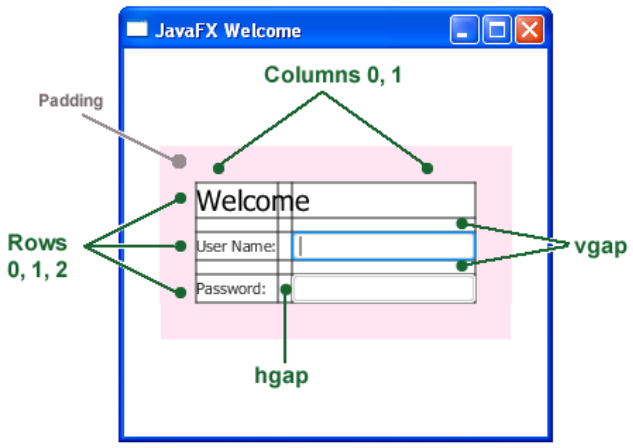
Les propriétés **gap** (d'écart) gèrent l'espacement entre les lignes et les colonnes, tandis que la propriété **Padding** gère l'espace autour des bords du volet de la grille.

Ajout de Text, Labels, and Text Fields

Comme le montre la figure, on va ajouter texte welcome, 2 label un textfield et un password textfield

```
Text scenetitle = new Text("Welcome");
scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL,
20));
grid.add(scenetitle, 0, 0, 2, 1);

Label userName = new Label("User Name:");
grid.add(userName, 0, 1);

TextField userTextField = new TextField();
grid.add(userTextField, 1, 1);

Label pw = new Label("Password:");
grid.add(pw, 0, 2);

PasswordField pwBox = new PasswordField();
grid.add(pwBox, 1, 2);
```

La méthode grid.add()ajoute le composant dans le layout grid.




```
Button btn = new Button("Sign in");
HBox hbBtn = new HBox(10);
hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
hbBtn.getChildren().add(btn);
grid.add(hbBtn, 1, 4);
```

Après la création du button nomé `btn` avec le label `Sign in`, puis on a créé un layout `HBox` pane hbBtn avec un espacement de 10 pixels. L'alignement du buttuon est de `Pos.BOTTOM_RIGHT`,

Le button est ajouté en tant que child du `HBox` pane, et le `HBox` pane est ajouté dans la grille dans la colonne 1, ligne 4.

Pui on ajoute le contrôle text dans lequel on va afficher le message.

```
final Text actiontarget = new Text();
        grid.add(actiontarget, 1, 6);
```

Ajout du code pour gerer (handle)l'évènement

Le message `("Sign in button pressed")` sera affiché dans le texte lorsqu'on clique sur le button

```
btn.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent e) {
        actiontarget.setFill(Color.FIREBRICK);
        actiontarget.setText("Sign in button pressed");
    }
});
```

Travail à faire :

- Ajouter un deuxième boutton **Exit** qui permet de fermer l'application sachant que l'instruction

  System.exit(0)
- Ajouter du code pour vérifier si le mot de passe est correct ou non