# TP : Découverte de JavaFX

## Objectif :

Le but de ce TP est de vous familiariser avec JavaFX en développant une petite application graphique. Vous apprendrez à créer des fenêtres, des scènes, à gérer les événements et à utiliser des composants graphiques de base.

## Prérequis :

- Installation d'Eclipse avec le plugin JavaFX.
- Connaissance de base en programmation Java.

## Étapes du TP :

### Étape 1 : Création du projet JavaFX

1. Ouvrez Eclipse et créez un nouveau projet JavaFX.
2. Configurez le projet pour utiliser JavaFX en tant que bibliothèque.

### Étape 2 : Création d'une fenêtre principale

1. Créez une nouvelle classe nommée "MainApp" qui étend la classe `Application` de JavaFX.
2. Implémentez la méthode `start(Stage primaryStage)` pour créer une fenêtre principale.

```
import javafx.application.Application;
import javafx.stage.Stage;

public class MainApp extends Application {

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

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

### Étape 3 : Ajout d'une scène

3.1 Création d'une Scène :

1. **Instanciation de la Scène :**

- Dans la méthode `start(Stage primaryStage)`, créez une instance de la classe `Scene` en passant le conteneur racine (le conteneur qui contiendra tous les composants graphiques) et les dimensions de la scène.

```
Scene scene = new Scene(root, 800, 600);
```

2. **Définition de la Taille de la Scène :**
   - Définissez la largeur et la hauteur de la scène selon vos besoins.

3.2 Association à la Fenêtre Principale :

1. **Association de la Scène à la Fenêtre Principale :**
   - Associez la scène que vous avez créée à la fenêtre principale (`primaryStage`).

```
primaryStage.setScene(scene);
```

2. **Personnalisation du Titre de la Fenêtre :**
   - Vous pouvez personnaliser le titre de la fenêtre principale.

```
primaryStage.setTitle("Ma Application JavaFX");
```

## 3.3 Exemple Complet de l'Étape 3 :

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class MainApp extends Application {

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

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Ma Première Application JavaFX");

        // Création d'un conteneur VBox (étape 4)
        VBox vbox = new VBox();

        // Ajout de composants au VBox (étape 4)

        // Création d'une scène et association au conteneur VBox
        Scene scene = new Scene(vbox, 800, 600);

        // Association de la scène à la fenêtre principale
        primaryStage.setScene(scene);

        // Affichage de la fenêtre
        primaryStage.show();
    }
}
```

L'étape 3 consiste essentiellement à créer une scène, définir ses dimensions et à l'associer à la fenêtre principale.

# Étape 4 : Ajout de composants graphiques

4.1 Ajout de Composants :

1. **Boutons et Étiquettes :**
   - Dans la méthode `start(Stage primaryStage)`, ajoutez des composants tels que des boutons et des étiquettes à votre fenêtre principale. Par exemple :

```
Button bouton = new Button("Cliquez-moi !");
Label etiquette = new Label("Bienvenue dans JavaFX !");
```

4.2 Organisation avec des Conteneurs :

1. **Utilisation de VBox (Vertical Box) :**
   - Créez un conteneur `VBox` pour organiser vos composants de manière verticale. Ajoutez les composants au conteneur.

```
VBox vbox = new VBox();
vbox.getChildren().add(etiquette);
vbox.getChildren().add(bouton);
```

2. **Positionnement du Conteneur dans la Scène :**
   - Ajoutez le `VBox` à la scène pour qu'il soit affiché dans la fenêtre principale.

```
Scene scene = new Scene(vbox, 300, 200);
primaryStage.setScene(scene);
```

3. **Personnalisation du Conteneur et des Composants :**
   - Vous pouvez personnaliser le `VBox` en ajoutant des marges, des espaces, ou en définissant d'autres propriétés CSS.

```
vbox.setSpacing(10); // Ajoute un espace de 10 pixels entre les composants
vbox.setPadding(new Insets(20, 20, 20, 20)); // Ajoute des marges autour du conteneur
```

4. **Ajout de Styles CSS :**
   - Expérimentez avec l'ajout de styles CSS pour modifier l'apparence des composants.

```
scene.getStylesheets().add("styles.css"); // Assurez-vous d'avoir un fichier CSS dans votre projet
```

5. **Adaptation à la Taille de la Scène :**
   - Pour que les composants s'adaptent à la taille de la scène, utilisez des classes telles que `Region` pour les composants.

```
etiquette.setMaxWidth(Double.MAX_VALUE);
bouton.setMaxWidth(Double.MAX_VALUE);
VBox.setVgrow(etiquette, Priority.ALWAYS);
VBox.setVgrow(bouton, Priority.ALWAYS);
```

## 4.3 Exemple Complet de l'Étape 4 :

```
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.VBox;
import javafx.stage.Stage;

public class MainApp extends Application {

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

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Ma Première Application JavaFX");

        // Ajout de composants
        Button bouton = new Button("Cliquez-moi !");
        Label etiquette = new Label("Bienvenue dans JavaFX !");

        // Organisation avec VBox
        VBox vbox = new VBox();
        vbox.getChildren().add(etiquette);
        vbox.getChildren().add(bouton);

        // Personnalisation du VBox
        vbox.setSpacing(10);
        vbox.setPadding(new Insets(20, 20, 20, 20));

        // Adaptation à la taille de la scène
        etiquette.setMaxWidth(Double.MAX_VALUE);
        bouton.setMaxWidth(Double.MAX_VALUE);
        VBox.setVgrow(etiquette, Priority.ALWAYS);
        VBox.setVgrow(bouton, Priority.ALWAYS);

        // Ajout du VBox à la scène
        Scene scene = new Scene(vbox, 300, 200);

        // Association de la scène à la fenêtre principale
        primaryStage.setScene(scene);

        // Affichage de la fenêtre
        primaryStage.show();
    }
}
```

## Étape 5 : Gestion des événements

5.1 Ajout de Gestionnaires d'Événements :

1. **Ajout de Gestionnaire pour un Bouton :**
    - Dans la méthode `start(Stage primaryStage)`, ajoutez un gestionnaire d'événements pour un bouton. Par exemple, pour afficher une boîte de dialogue lorsqu'un bouton est cliqué :

```
bouton.setOnAction(e -> {
    Alert alert = new Alert(AlertType.INFORMATION);
```

```
        alert.setTitle("Information");
        alert.setHeaderText(null);
        alert.setContentText("Le bouton a été cliqué !");
        alert.showAndWait();
    });
```

2. **Autres Gestionnaires d'Événements :**
    - Les gestionnaires d'événements peuvent également être ajoutés à d'autres composants, tels que les zones de texte, les listes déroulantes, etc.

5.2 Gestion des Événements de Clavier :

1. **Gestion des Touches Clavier :**
    - Pour gérer les événements du clavier, ajoutez un écouteur d'événements à la scène.

```
scene.setOnKeyPressed(e -> {
    System.out.println("Touche pressée : " + e.getCode());
});
```

5.3 Gestion des Événements de Souris :

1. **Gestion des Clics de Souris :**
    - Pour gérer les événements de souris, ajoutez des écouteurs d'événements à des composants spécifiques.

```
bouton.setOnMouseClicked(e -> {
    System.out.println("Clic de souris sur le bouton !");
});
```

2. **Détection de la Position de la Souris :**
    - Vous pouvez également obtenir les coordonnées de la position de la souris lors d'un clic.

```
scene.setOnMouseClicked(e -> {
    double x = e.getSceneX();
    double y = e.getSceneY();
    System.out.println("Clic de souris à la position : (" + x + ", "
+ y + ")");
});
```

5.4 Exemple Complet de l'Étape 5 :

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class MainApp extends Application {

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

    @Override
```

```
    public void start(Stage primaryStage) {
        primaryStage.setTitle("Ma Première Application JavaFX");

        // Création d'un bouton (étape 4)
        Button bouton = new Button("Cliquez-moi !");

        // Ajout d'un gestionnaire d'événements pour le bouton
        bouton.setOnAction(e -> {
            Alert alert = new Alert(AlertType.INFORMATION);
            alert.setTitle("Information");
            alert.setHeaderText(null);
            alert.setContentText("Le bouton a été cliqué !");
            alert.showAndWait();
        });

        // Création d'un conteneur VBox (étape 4)
        VBox vbox = new VBox();
        vbox.getChildren().add(bouton);

        // Création d'une scène et association au conteneur VBox (étape 3)
        Scene scene = new Scene(vbox, 800, 600);

        // Ajout d'un gestionnaire d'événements pour les touches du clavier
        scene.setOnKeyPressed(e -> {
            System.out.println("Touche pressée : " + e.getCode());
        });

        // Ajout d'un gestionnaire d'événements pour un clic de souris
        bouton.setOnMouseClicked(e -> {
            System.out.println("Clic de souris sur le bouton !");
        });

        // Ajout d'un gestionnaire d'événements pour obtenir la position du
clic de souris
        scene.setOnMouseClicked(e -> {
            double x = e.getSceneX();
            double y = e.getSceneY();
            System.out.println("Clic de souris à la position : (" + x + ", "
+ y + ")");
        });

        // Association de la scène à la fenêtre principale (étape 3)
        primaryStage.setScene(scene);

        // Affichage de la fenêtre
        primaryStage.show();
    }
}
```

Cette étape permet d'illustrer la manière dont les événements, qu'ils soient liés au clavier, à la souris ou à d'autres interactions, peuvent être gérés dans une application JavaFX.

## Étape 6 : Style et mise en page

1. Appliquez des styles CSS à vos composants graphiques.
2. Expérimentez avec différentes mises en page pour organiser vos éléments.

```
// Ajout de styles CSS

scene.getStylesheets().add(MainApp.class.getResource("application.css").toExter
nalForm());
```

# Conclusion :

À la fin de ce TP, vous devriez être capables de créer une application simple en utilisant JavaFX, comprendre la structure de base d'un projet JavaFX et connaître les concepts de base tels que les scènes, les composants graphiques et les gestionnaires d'événements.