JavaFX & Swing

JavaFX and Swing are both Java libraries used for building desktop applications with graphical user interfaces (GUIs). Each has its own set of advantages and disadvantages, and the choice between them depends on factors such as project requirements, personal preference, and the target platform.

Below is the brief comparison between both.

Graphics and Animation:

*Swing***:** Swing lacks built-in support for advanced graphics and animation. While it's possible to create basic animations and custom graphics using Swing, it requires more effort and is not as efficient as JavaFX.

*JavaFX***:** JavaFX provides built-in support for high-quality graphics, animations, and effects. It offers features such as transitions, transformations, and timeline-based animations out of the box, making it well-suited for creating visually appealing UIs.

Styling and Theming:

Swing: Swing supports basic styling and theming through Look and Feel (L&F) mechanisms. However, customizing the appearance of Swing components can be challenging and often requires creating custom UI delegates.

JavaFX: JavaFX offers extensive support for styling and theming through CSS (Cascading Style Sheets). You can easily apply CSS styles to JavaFX components to customize their appearance, making it easier to create visually consistent and attractive UIs.

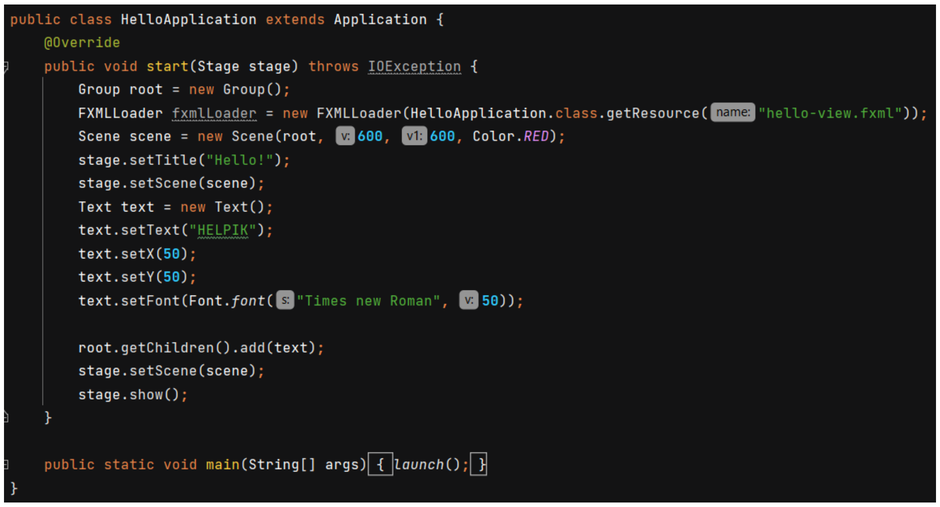
Performance:

Swing: Swing applications are generally lightweight and have good performance for most use cases. However, performance may degrade when dealing with complex UIs or heavy graphics.

JavaFX: JavaFX applications tend to have better performance compared to Swing, especially when it comes to rendering complex graphics and animations. JavaFX leverages hardware acceleration for rendering, resulting in smoother and more responsive UIs.

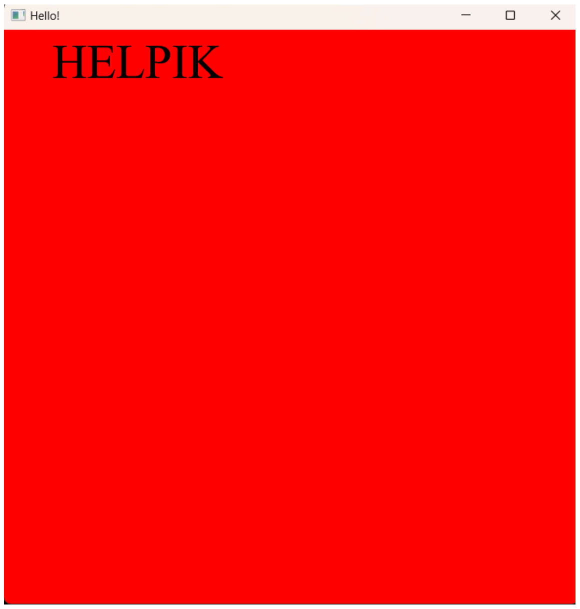
In a nutshell, JavaFX seems to be a better fit for our project. It provides the necessary features and capabilities to create a modern and visually appealing chatbot UI integrated with a weather API

Demo of JavaFx:

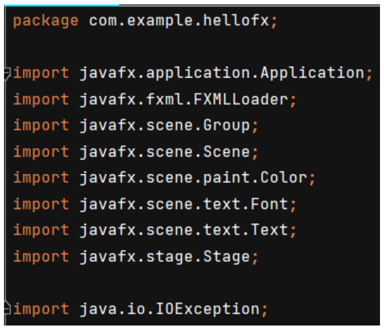


```
public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        Group root = new Group();
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource( name: "hello-view.fxml"));
        Scene scene = new Scene(root,  v: 600,  v1: 600, Color.RED);
        stage.setTitle("Hello!");
        stage.setScene(scene);
        Text text = new Text();
        text.setText("HELPIK");
        text.setX(50);
        text.setY(50);
        text.setFont(Font.font( s: "Times new Roman",  v: 50));

        root.getChildren().add(text);
        stage.setScene(scene);
        stage.show();
    }

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

The above code generates the output below.



The below are some of the packages and classes that JavaFx import



```
package com.example.hellofx;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.paint.Color;
import javafx.scene.text.Font;
import javafx.scene.text.Text;
import javafx.stage.Stage;

import java.io.IOException;
```