**Comsats University Islamabad, Lahore campus**

**Assignment #2**

**OOP**

**Submitted To:**

**Sir, Shahid Bhatti**

**Submitted by:**

**Muhammad Aqib Javed**

**Sp24-BSE-150**

**Section: A**

**Code Explanation for Application Form**

This project is a **JavaFX-based GUI application** that simulates an application form. It collects user details such as name, father's name, CNIC, date of birth, gender, and city, and allows uploading a profile picture. The data is stored in a Person object and displayed in the console for simplicity.

**App.java (Main Application Class)**

**1. Imports**

**The program imports essential Java and JavaFX classes:**

* javafx.application.Application: Base class for JavaFX applications.
* javafx.scene.Scene, javafx.scene.control.\*, javafx.scene.image.\*, etc.: Components and layout utilities for building the GUI.
* java.io.File, java.util.ArrayList: Standard Java utilities for file handling and managing a list of Person objects.

**2. Attributes**

* ArrayList<Person> personList: Stores the submitted form details.
* ImageView imageView: Displays the uploaded image.

**3. Application Start Method**

This method initializes and sets up the application interface.

**Banner Setup**

* A Label styled to display the title "Application Form".
* Placed in an HBox for horizontal alignment and styled with a dark background.

**Form Fields**

Several input fields allow users to fill in their details:

1. **Text Fields**:
   * nameField for the user's name.
   * fatherNameField for the father's name.
   * cnicField for CNIC.
   * cityField for city.
2. **Date Picker**: datePicker for selecting a date.
3. **ComboBox**: genderComboBox for gender selection.
4. **Image Upload**:
   * A FileChooser is used to select an image file (JPG/PNG).
   * Selected image is displayed in imageView. (learned about this online )

**Submit Button**

* Creates a new Person object with the entered details.
* Adds the Person object to personList.
* Clears the form fields and the displayed image after submission.
* Logs the submitted Person object using System.out.println.

**4. Layout**

The layout is built using JavaFX layout components:

1. **GridPane**: Arranges the form fields in a grid structure.
2. **VBox**: Used to combine the banner and form into a vertical layout.

Each component is styled using CSS-like properties for a modern, sleek design.

**5. Helper Method**

* createStyledLabel(String text): Creates a label with consistent styling for form labels.

**6. Main Method**

Calls the launch() method to start the JavaFX application.

**Person.java (Data Model Class)**

**Represents an individual user in the application.**

**1. Attributes**

* **String name: User's name.**
* **String fatherName: User's father's name.**
* **String cnic: User's CNIC.**
* **String date: User's date of birth.**
* **String gender: User's gender.**
* **String city: User's city.**

**2. Constructor**

Initializes the attributes with data from the form.

**3. toString() Method**

Returns a formatted string representing the Person object. Useful for logging the user's details in the console.

**How It Works**

1. The user fills in the form fields and uploads an image.
2. Clicking the **Submit** button:
   * Creates a Person object.
   * Clears the form and resets the image view.
   * Logs the Person details to the console.
3. Uploaded images are displayed next to the **Name** field.

**Features**

* **Dynamic UI Elements**: Supports text input, dropdowns, date picking, and file selection.
* **Real-Time Image Display**: Selected images are displayed in the GUI.
* **Object-Oriented Design**: Separates the form logic (App.java) and data model (Person.java).

**Potential Enhancements**

1. **Data Persistence**: Store Person objects in a file or database.
2. **Validation**: Add checks for empty fields or invalid input.
3. **Display Saved Data**: Display submitted data in a table view.
4. **Enhanced Styling**: Add more CSS-based themes for a polished look.

# GUI Image:

# Information filled:

# Selecting image:

# Image Display:

# Console Output: