**Class HelloApplication**

**1. Package Declaration**

java

Copy code

package com.example.demo;

* Specifies the package name for organizing the code.

**2. Import Statements**

* The program imports necessary JavaFX classes and standard libraries to create a graphical user interface (GUI), handle images, manage layouts, and work with file selection.

**3. Class Declaration**

java

Copy code

public class HelloApplication extends Application {

* The HelloApplication class extends Application, making it a JavaFX application.

**4. Global Variables**

java

Copy code

ArrayList<Person> personList = new ArrayList<>();

* personList is used to store data about people entered in the form.

**5. Main Method**

java

Copy code

public static void main(String[] args) {

launch(args);

}

* The launch method starts the JavaFX application by calling the start method.

**6. start Method**

* This is where the JavaFX GUI is created.

**6.1. Layout Setup**

java

Copy code

BorderPane mainLayout = new BorderPane();

* BorderPane divides the layout into five areas (top, bottom, left, right, center).
* Used here to organize different sections of the form.

**6.2. Top Section (Banner)**

java

Copy code

Label banner = new Label("Entry Form");

banner.setStyle("-fx-font-size: 28px; ...");

* A label is created as the banner with styling (font size, weight, and color).
* It is set to the **top** of the layout.

**6.3. Form Grid (Left Section)**

java

Copy code

GridPane grid = new GridPane();

* A GridPane organizes form fields into rows and columns.
* The form includes:
  + **Name**, **Father Name**, **CNIC** (text fields).
  + **Date of Birth** (date picker).
  + **Gender** (radio buttons with a toggle group).
  + **City** (combo box for city selection).
  + **Image** (button to choose an image).
  + **Save** (button to save the form data).

**6.4. Adding Form Fields to Grid**

java

Copy code

grid.add(name, 0, 0);

grid.add(nameText, 1, 0);

* Form fields are added to specific rows and columns in the grid.

**6.5. Image Upload (Right Section)**

java

Copy code

ImageView imageView = new ImageView();

imageButton.setOnAction(e -> { ... });

* An ImageView displays the selected image.
* A FileChooser lets users upload an image file, which is displayed in the ImageView.

**6.6. Save Button Action**

java

Copy code

saveButton.setOnAction(e -> { ... });

* The **Save** button collects data from the form fields and creates a Person object:
  + Name
  + Father Name
  + CNIC
  + Date of Birth
  + Gender
  + City
* The Person object is added to the personList.

**6.7. Combining Sections**

java

Copy code

mainLayout.setLeft(grid);

mainLayout.setRight(rightSection);

* The GridPane (form) is placed on the **left**, and the image display section is placed on the **right** of the BorderPane.

**7. Scene Setup**

java

Copy code

Scene scene = new Scene(mainLayout, 800, 600);

stage.setTitle("Form Layout Example");

stage.setScene(scene);

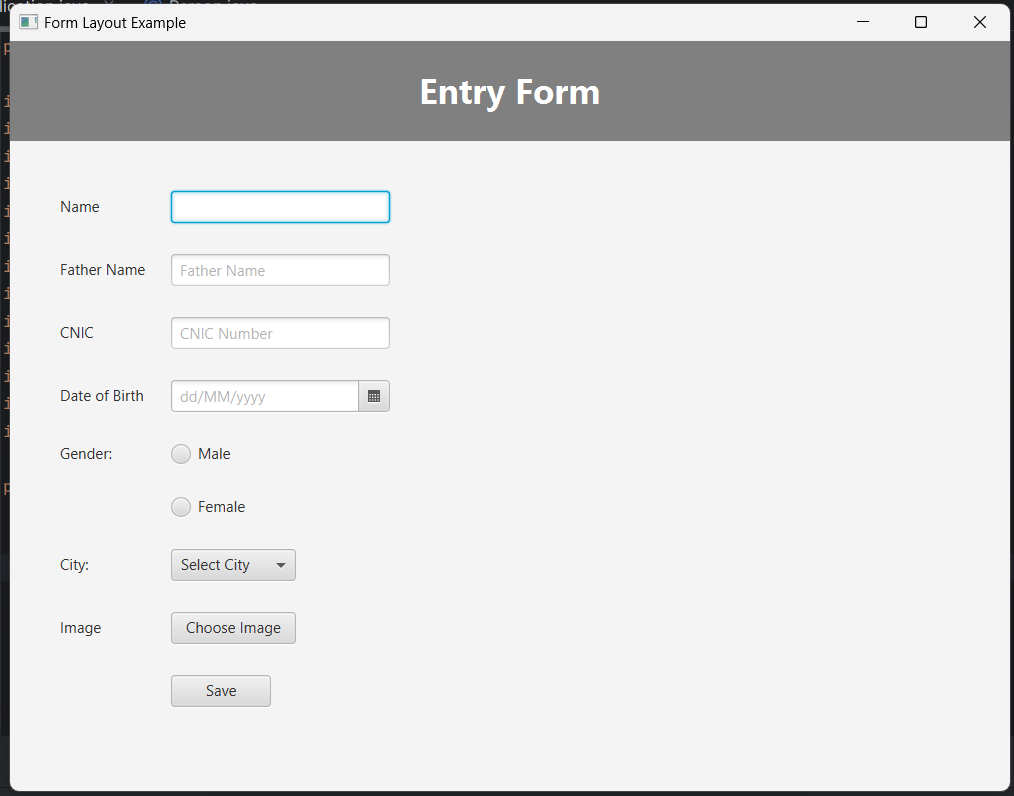
stage.show();

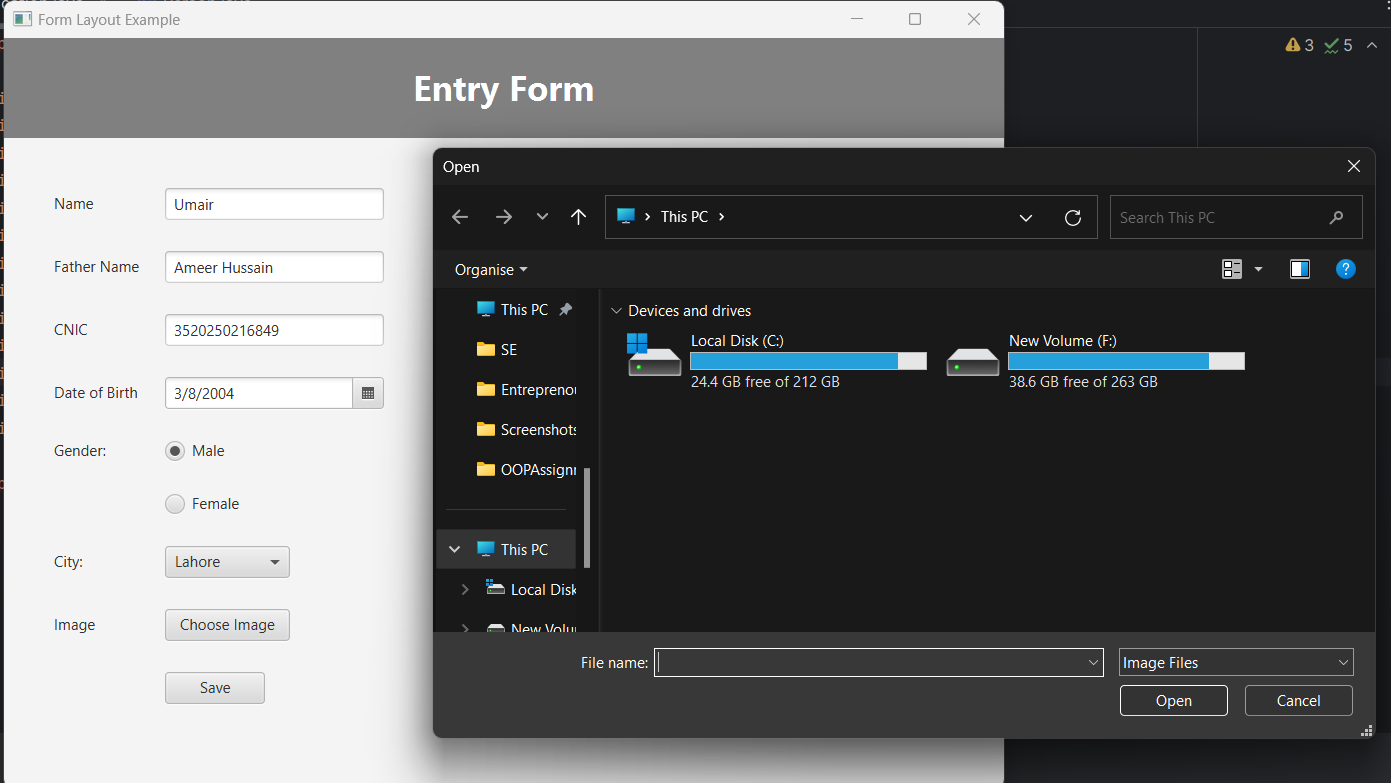
* A Scene is created with the BorderPane as the root layout.
* The Stage displays the window with the scene.

**Key Features of the Application**

* **GUI Components**: Labels, TextFields, ComboBoxes, RadioButtons, DatePicker, Buttons, and ImageView.
* **Event Handling**: Button clicks trigger file selection and data saving actions.
* **Layout Management**: BorderPane organizes sections, and GridPane organizes form fields.
* **Data Storage**: Uses an ArrayList<Person> to store form entries

**Output Screen**

****



**Person Class**

**Purpose**

* The Person class is a blueprint for creating objects that store information about a person. It encapsulates data such as name, father name, CNIC, date of birth, gender, and city.

**Attributes (Fields)**

java

Copy code

private String name;

private String fatherName;

private String cnic;

private String dateOfBirth;

private String gender;

private String city;

* These are private variables used to store information about a person.
* Declaring them private ensures encapsulation, so they can only be accessed or modified through methods in the class.

**Constructor**

java

Copy code

public Person(String name, String fatherName, String cnic, String dateOfBirth,

String gender, String city) {

this.name = name;

this.fatherName = fatherName;

this.cnic = cnic;

this.dateOfBirth = dateOfBirth;

this.gender = gender;

this.city = city;

}

* The constructor initializes a Person object with the provided values.
* Parameters are passed during object creation and assigned to the corresponding fields using the this keyword.

**Getter Methods**

java

Copy code

public String getName() { return name; }

public String getFatherName() { return fatherName; }

public String getCnic() { return cnic; }

public String getDateOfBirth() { return dateOfBirth; }

public String getGender() { return gender; }

public String getCity() { return city; }

* These methods allow controlled access to the private fields.
* Each getter method returns the value of a specific field.

**toString Method**

java

Copy code

@Override

public String toString() {

return "Name: " + name + " Father: " + fatherName + " CNIC: " + cnic +

" Date: " + dateOfBirth + " Gender: " + gender + " City: " + city;

}

* The toString method overrides the default implementation from the Object class.
* It provides a readable string representation of a Person object.
* When you print a Person object (e.g., System.out.println(person)), this method is automatically called.

**How It Works in the Application**

* In the HelloApplication class:
  1. A Person object is created when the **Save** button is clicked.
  2. Data from the form fields is passed to the Person constructor.
  3. The Person object is added to the personList for storage.
* Example:

java

Copy code

Person person = new Person(

nameText.getText(),

fatherNameText.getText(),

cnicText.getText(),

dobPicker.getValue() != null ? dobPicker.getValue().toString() : "",

selectedGender,

cityComboBox.getValue()

);

**Summary**

The Person class:

* Stores personal information.
* Provides getter methods for accessing this information.
* Offers a toString method for easy display.

This makes it a reusable and organized way to handle data related to individuals in your application.