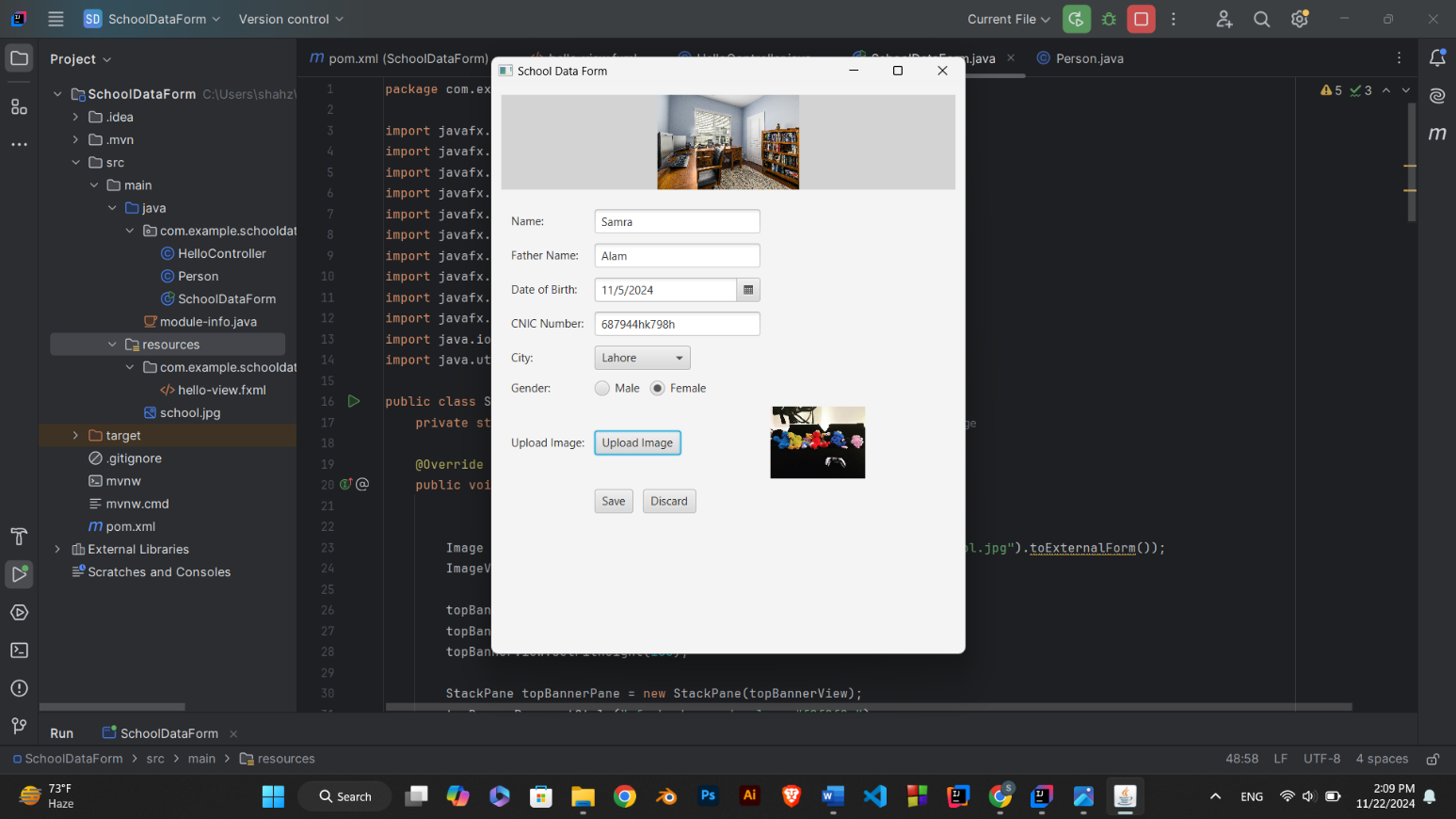
# School Data Entry Form



A screenshot of a computer

Description automatically generated

This program consists of two main components:

1. **A SchoolDataForm GUI Application**  
   This is a JavaFX application that provides a form to capture user details (like name, father's name, date of birth, etc.) and store them in a list.
2. **A Person Class**  
   The Person class represents an individual with specific attributes, such as name, father's name, date of birth, CNIC, city, and gender.

**Detailed Working**

**1. Application Initialization**

* The SchoolDataForm class extends Application and overrides the start(Stage primaryStage) method.
* A Stage (window) and Scene (content inside the window) are set up to create the GUI.

**2. UI Components**

The user interface contains:

1. **Header Image**:
   * Displays an image (school.jpg) at the top of the form using an ImageView.
2. **Input Fields**:
   * **TextFields**:
     + To input the Name, Father Name, and CNIC.
   * **DatePicker**:
     + To select the date of birth.
   * **ComboBox**:
     + Dropdown to select the city.
   * **RadioButtons**:
     + Grouped using a ToggleGroup to select the gender (Male/Female).
   * **FileChooser**:
     + Used to upload an image. The image is displayed in a ImageView.
3. **Buttons**:
   * **Save**:
     + Captures the data entered in the form, creates a Person object, adds it to an ArrayList, and displays the saved details in an alert.
     + It also clears the form after saving.
   * **Discard**:
     + Clears all form inputs and resets the view to its initial state.

**3. Data Handling**

* **When Save Button is Clicked**:
  1. The data from the input fields are retrieved.
  2. A Person object is created with the captured details.
  3. The Person object is added to the ArrayList<Person> named persons.
  4. An Alert dialog is shown with the saved person's details, using the toString() method of the Person class.
  5. The form inputs are cleared for a fresh start.
* **When Discard Button is Clicked**:
  1. All form fields are cleared, and the selected image is reset.

**4. The Person Class**

* This class represents the model (data structure) for storing individual user data.
* It has private fields (encapsulation) for the person's attributes.
* A constructor initializes these attributes.
* Getter methods allow access to these attributes.
* The toString() method provides a formatted string representation of the person's details.

**5. Data Storage**

* The program stores all Person objects in a static ArrayList<Person>. This list serves as an in-memory data store while the application runs.

**Features**

1. **User-Friendly GUI**:
   * A form-based UI makes it easy to input data.
2. **In-Memory Data Storage**:
   * Captured data is stored in a list for potential future processing.
3. **Reusable Person Class**:
   * Encapsulates all details of a person and provides a clean interface for accessing and displaying the data.
4. **Image Uploading**:
   * Users can upload and preview an image for the person.

**How It Works (Flow)**

1. The application launches, displaying the form.
2. The user enters data (name, father’s name, etc.), selects options (gender, city), and uploads an image.
3. The user clicks:
   * **Save**:
     + Saves the data into the ArrayList, shows confirmation, and clears the form.
   * **Discard**:
     + Clears the form inputs.
4. The cycle repeats until the user closes the application.