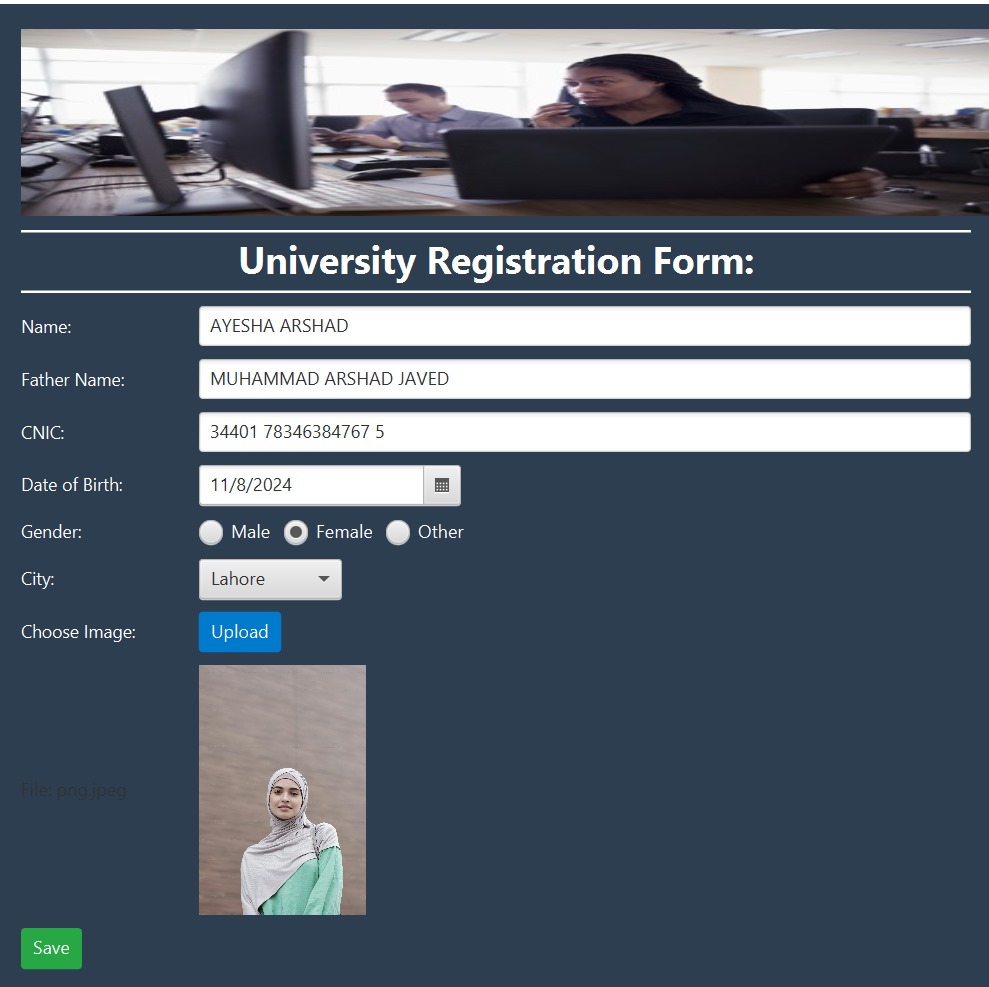


|  |  |
| --- | --- |
| **Name:** | Ayesha Arshad |
| **Reg No:** | SP24-BSE(B)-019 |
| **Course title:** | Object Oriented Programming |
| **Instructor:** | Prof Shahid Bhatti |
| **Assignment no:** | Assignment 2 |
| **Date:** | 11/22/24 |

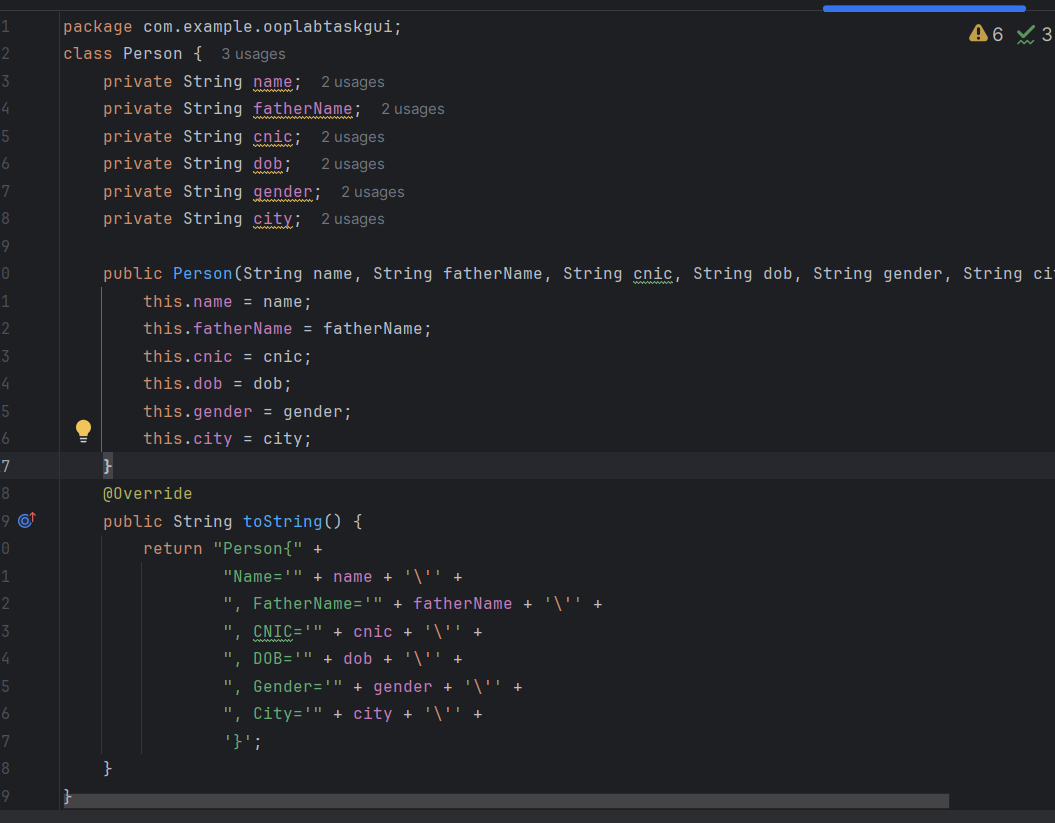
Here is the output demonstration of my GUI code form:



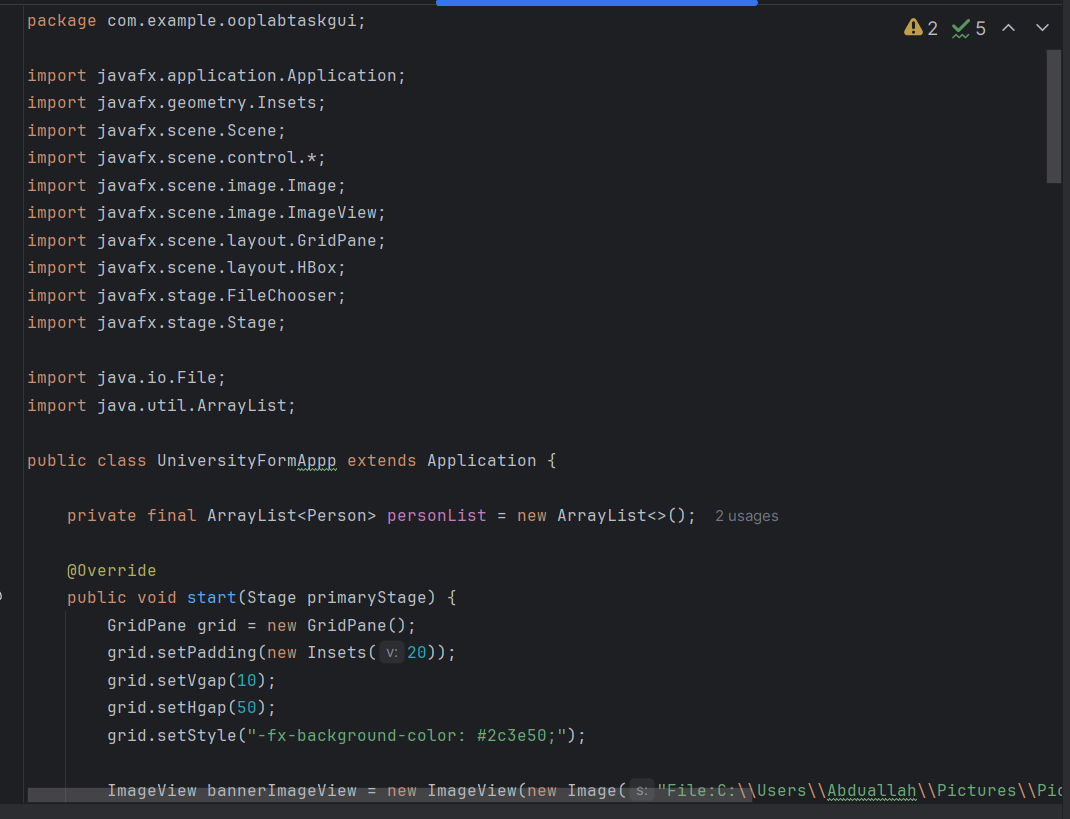
The output of this program is an elegant and user-friendly GUI-based University Registration Form designed using JavaFX. The form features a clean and modern layout with a professional appearance. At the top, a banner image is displayed, followed by a bold heading: *"University Registration Form"*. Below this, a series of input fields and interactive components allow users to enter and select various details required for registration, including:

1. **Personal Information:** Name, Father Name, CNIC, and Date of Birth fields.
2. **Gender and City Selection:** Dropdown menus for selecting gender and city.
3. **Image Upload:** A button to upload a profile picture, with a live preview displayed upon selection.
4. **Save Button and Feedback:** A save button to submit the form, with a feedback label that informs users of any missing fields or successful form submission.

**Here goes my code:**

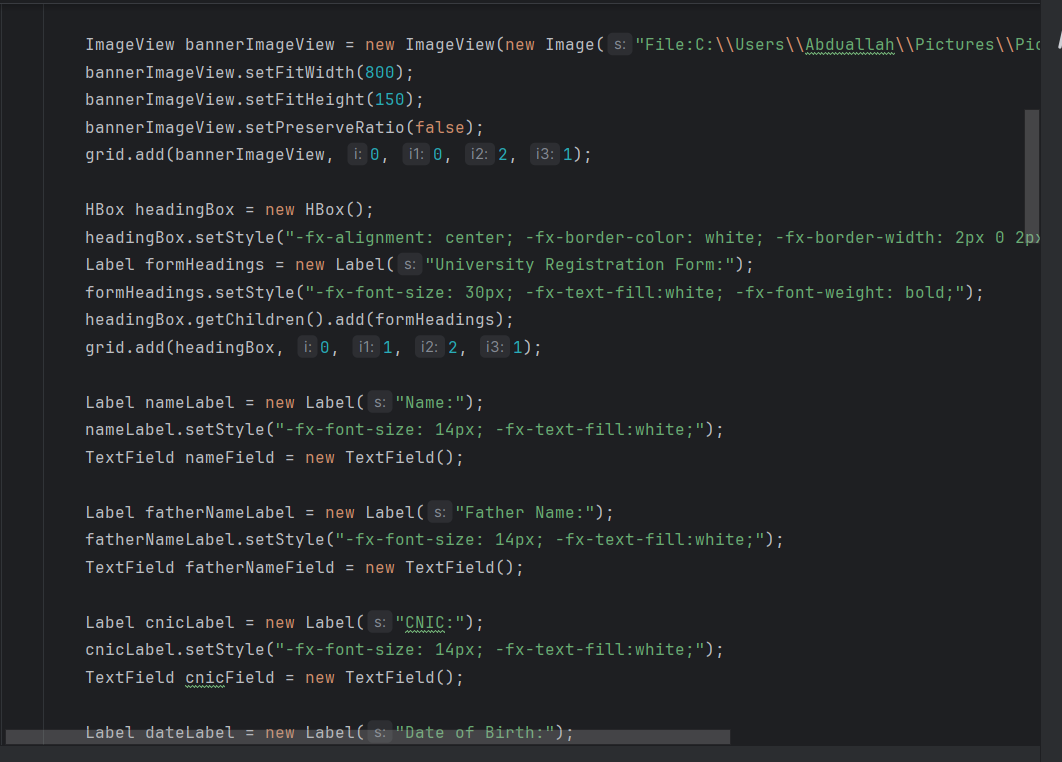
**PERSON CLASS:**

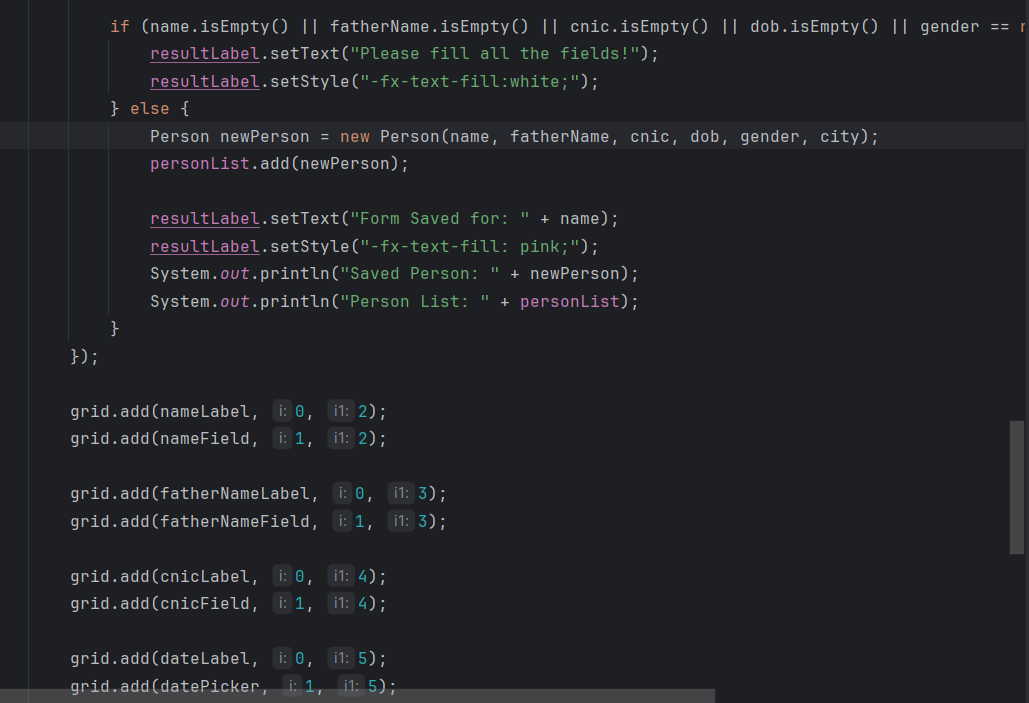
The Person class serves as a blueprint for storing user information collected from the registration form. It contains attributes like name, fatherName, cnic, dob, gender, city, and imagePath to represent personal details. The class provides a constructor for initializing these attributes and getter/setter methods to access and modify them. It ensures data encapsulation and supports creating objects to represent individual registrants.

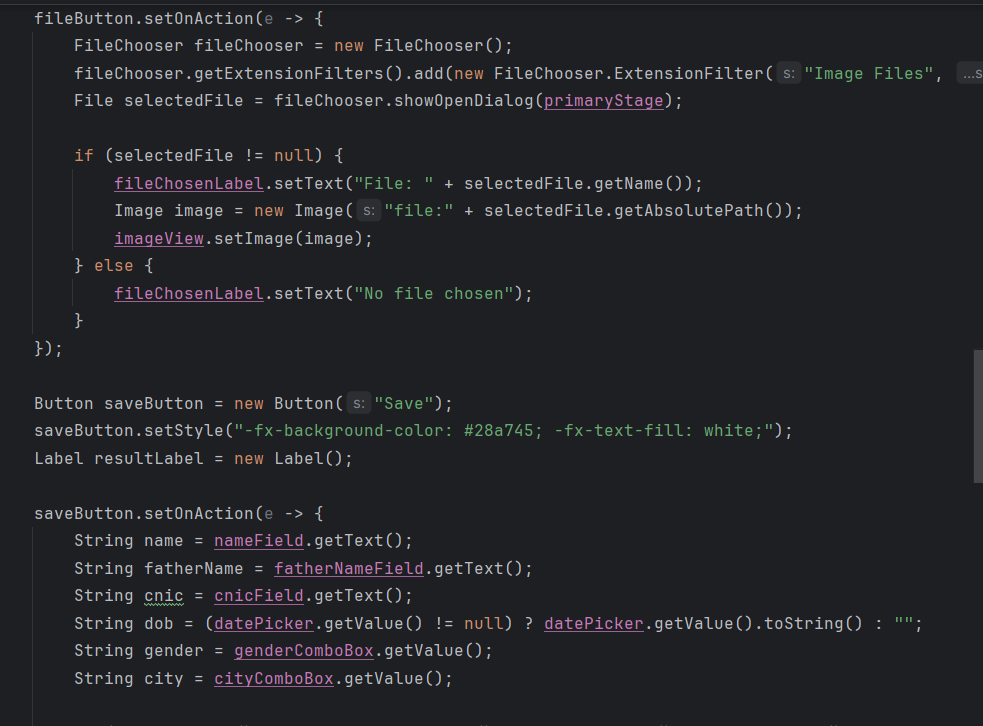
**UniversityFormAppp Class:**

he UniversityFormApp class is the main application for creating a university registration form using JavaFX.

It features a user-friendly interface with fields to input personal details like name, father's name, CNIC, date of birth, gender, city, and an option to upload an image.

The class uses a GridPane layout to organize components and applies CSS styles for an enhanced appearance.





This code creates a FileChooser to allow the user to select an image file from their system. The getExtensionFilters() method is used to restrict the file types to images (such as .png, .jpg, .jpeg). Once the file is selected, the file’s path is displayed, and an Image object is created with the chosen file path.



**Class Declaration**: The UniversityFormAppp class extends Application to create a JavaFX application.

**Person List**: A personList of type ArrayList<Person> is used to store the form data of each person submitted through the form.

**GridPane Layout**: The form layout uses GridPane to organize the UI components, such as text fields and buttons, in a grid format with custom padding and gaps.

**Banner Image**: A bannerImageView is used to display a university banner at the top of the form. The image is loaded from the local file system.

**Heading Box**: An HBox is used for the form heading, which is styled with bold, white text and aligned to the center.

**Form Fields**: The form includes labels and corresponding input fields for personal information like **Name**, **Father's Name**, **CNIC**, **Date of Birth**, **Gender**, and **City**.

**ComboBoxes**: Two ComboBox controls are used for selecting **Gender** (Male, Female, Other) and **City** (Lahore, Karachi, Islamabad, Other).

**DatePicker**: A DatePicker is used to select the **Date of Birth** from a calendar popup.

**File Upload Button**: A button labeled "Upload" triggers a file chooser dialog to select an image file, which is displayed in the ImageView.

**Image Display**: The selected image file is displayed in an ImageView that is sized and maintained in proportion.

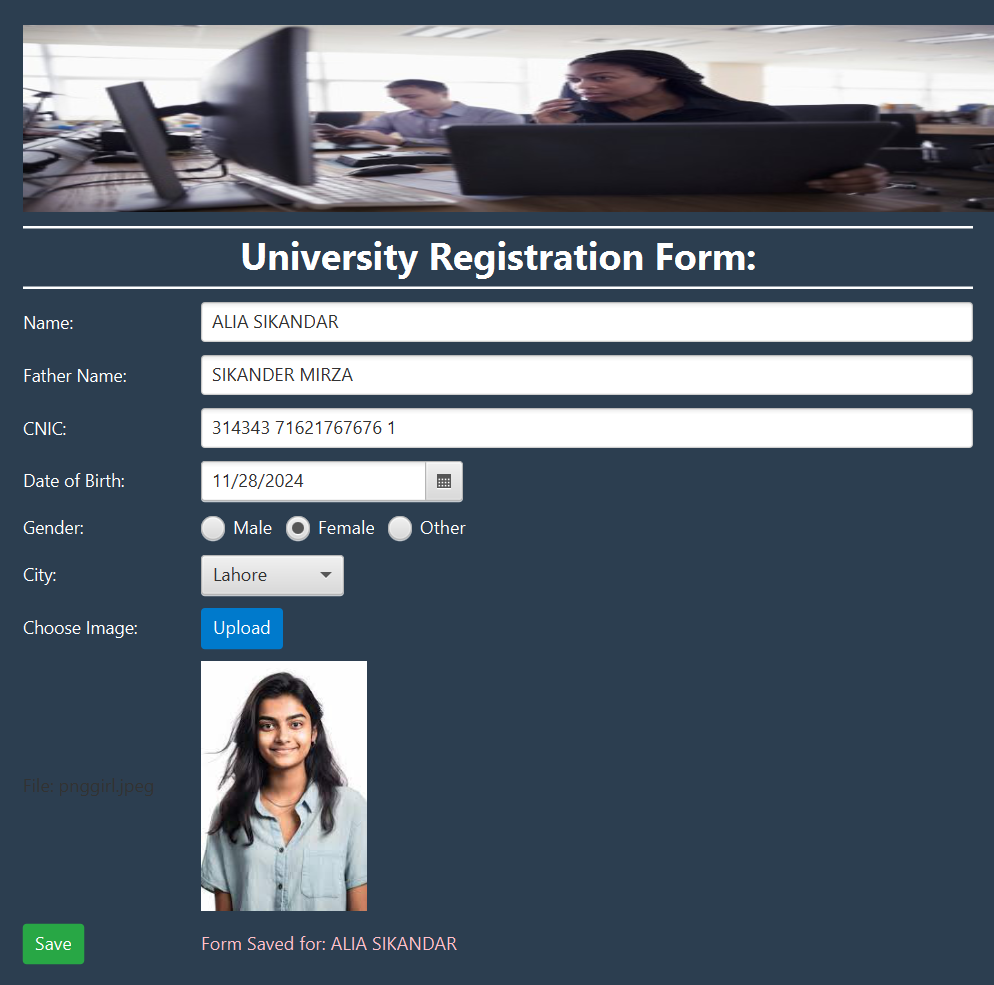
**Save Button**: A "Save" button allows the user to submit the form. The entered data is validated, and if all fields are filled, it adds the data to the personList.

**Validation**: If any field is left empty, a message is displayed to prompt the user to fill all the fields.

**Person Object**: When the form is valid, a new Person object is created with the form data and added to the personList.

**Result Label**: After saving the form data, a result label displays a confirmation message with the saved person’s name.

**UI Styling**: The application applies custom styles using CSS, including background color, text color, button styles, and borders to make the form visually appealing.

**OUTPUT:  
**

When the user fills in the form and clicks the Save button, the application validates the input data to ensure all required fields are completed. If any fields are missing, a message prompts the user to fill all fields. Once the form is valid, the data is used to create a new Person object, which contains the user's details, such as name, father’s name, CNIC, date of birth, gender, and city. This object is then added to the personList, an ArrayList<Person>, which stores all submitted form entries. The application displays a confirmation message in the form of a label, indicating that the data has been successfully saved for the user. This process not only ensures that the data is captured and stored but also allows the program to keep track of all the users who have submitted the form, making the list of people dynamically expandable as more submissions are made.

