



COMSATS University Islamabad, Lahore Campus

Object Oriented Programming

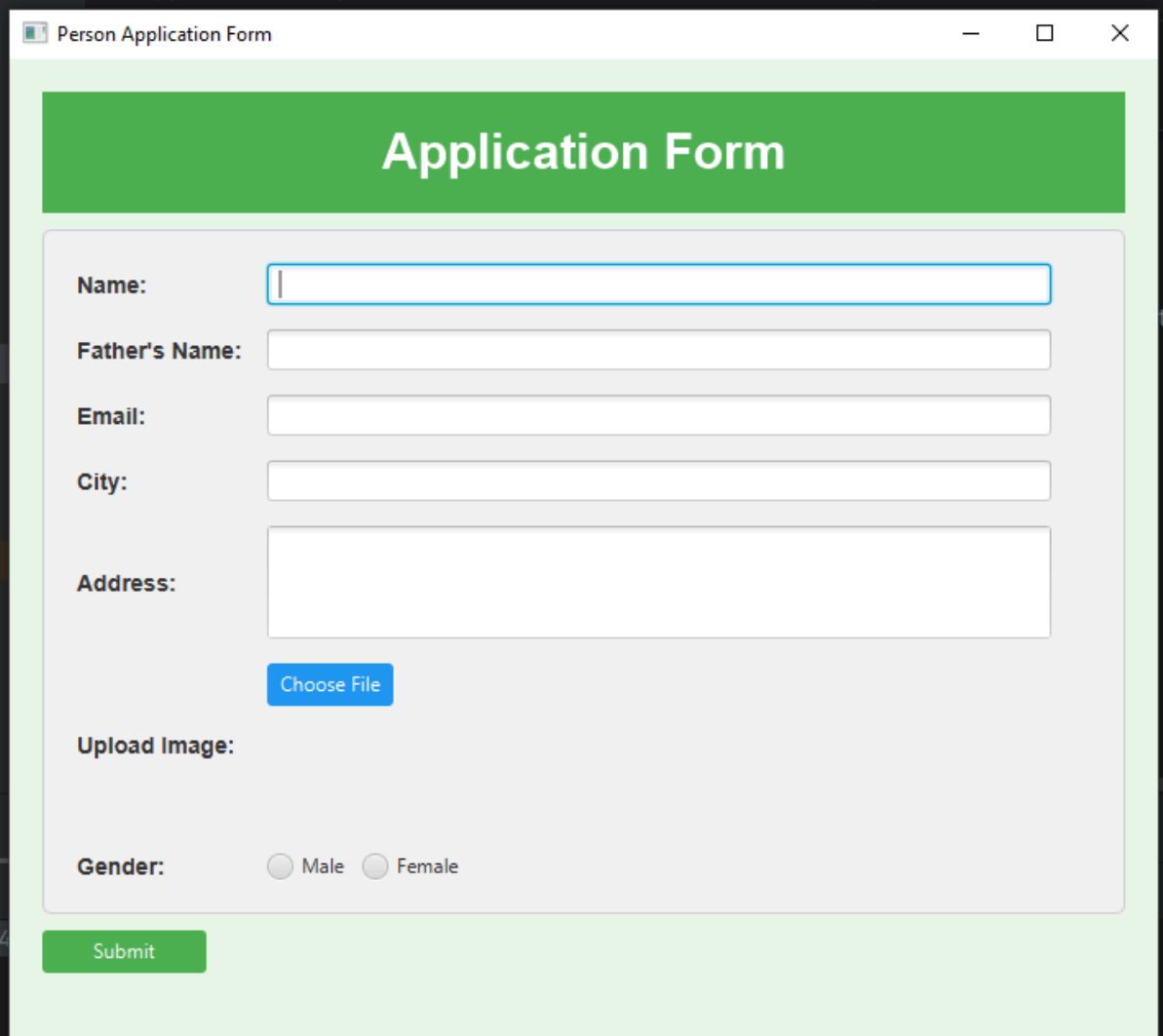
Assignment

Submitted to: Sir Shahid Bhatti

Submitted by: Khadija Noor - SP24-BSE-052

Program Overview

- The program uses JavaFX to create a GUI (Graphical User Interface).
- The main window (stage) has a banner at the top, a form layout in the middle, and a submit button at the bottom.
- When users fill out the form and press "Submit," the details are displayed in a new window.



Person Application Form

Application Form

Name:

Father's Name:

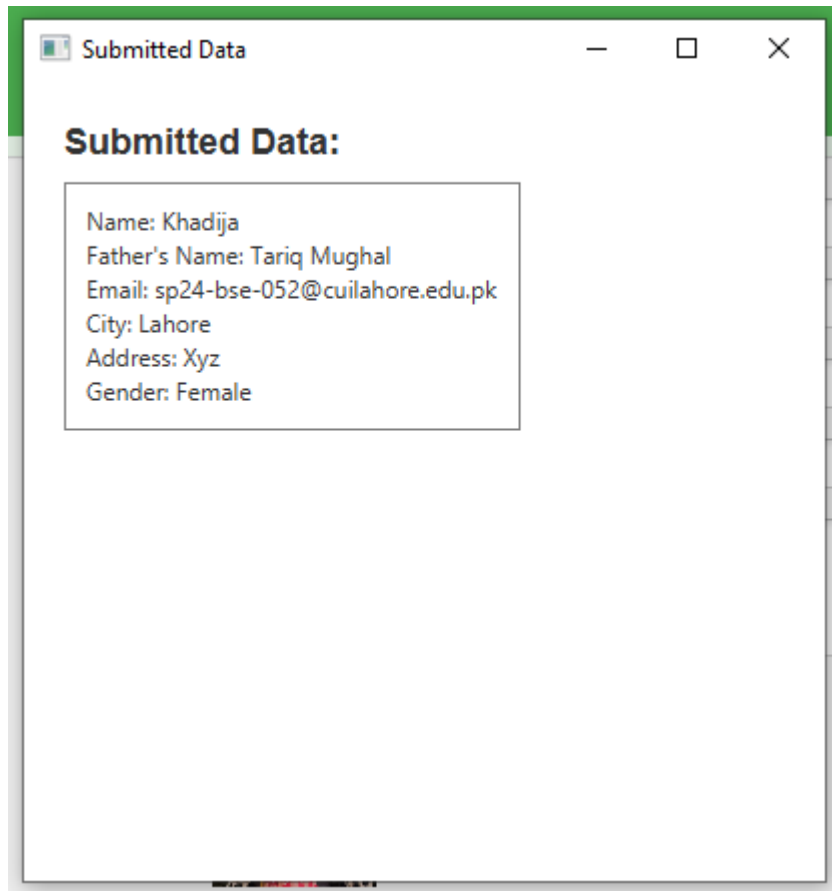
Email:

City:

Address:

Upload Image:

Gender: ☐ Male ☐ Female



Code Explanation:

1. Main Class:

This is the main class where the class Application is extended and JavaFX feature is used.

```
1 package com.example.dataform;
2
3 > import ...
23
24 ▶ public class ApplicationFormFX extends Application {
25
26     // List to store submitted data
27     private ObservableList<String[]> submittedData = FXCollections.observableArrayList(); 2 usages
28
29 ▶ > public static void main(String[] args) { launch(args); }
32
33     @Override
34     @@@ public void start(Stage primaryStage) {
35         primaryStage.setTitle("Person Application Form");
36     }
```

2. Banner Section

This is the colorful banner at the top of the form. It uses a `StackPane` for centering the text inside it. It creates a green rectangular banner with white text that says *Application Form*.

```
36
37 // Create a colorful banner
38 StackPane bannerPane = new StackPane();
39 bannerPane.setStyle("-fx-background-color: #4CAF50; -fx-padding: 20px;");
40
41 Text bannerText = new Text(s: "Application Form");
42 bannerText.setFont(Font.font(s: "Arial", FontWeight.BOLD, v: 30));
43 bannerText.setFill(Color.WHITE);
44 bannerPane.getChildren().add(bannerText);
45
```

3. Form Layout

The form is created using a `GridPane`, where each row contains a label and a corresponding input field. It creates a green rectangular banner with white text that says *Application Form*.

Example: Name Field

```
45
46 // Form elements
47 GridPane formGrid = new GridPane();
48 formGrid.setPadding(new Insets(v: 20));
49 formGrid.setHgap(15);
50 formGrid.setVgap(15);
51 formGrid.setStyle("-fx-background-color: #f4f4f4; -fx-border-color: #cccccc; -fx-border-radius: 5px;");
52
53 // Name
54 Label nameLabel = new Label(s: "Name:");
55 nameLabel.setFont(Font.font(s: "Arial", FontWeight.BOLD, v: 14));
56 TextField nameField = new TextField();
57 formGrid.add(nameLabel, i: 0, i1: 0);
58 formGrid.add(nameField, i: 1, i1: 0);
59
```

Example: Email Field

```
66
67 // Email
68 Label emailLabel = new Label(s: "Email:");
69 emailLabel.setFont(Font.font(s: "Arial", FontWeight.BOLD, v: 14));
70 TextField emailField = new TextField();
71 formGrid.add(emailLabel, i: 0, i1: 2);
72 formGrid.add(emailField, i: 1, i1: 2);
73
```

4. Image Upload

Users can upload an image using a FileChooser, and the selected image is displayed in an **ImageView**. It allows users to upload an image and displays the uploaded image in a preview box.

```
39 // Image Upload
40 Label imageLabel = new Label(s: "Upload Image:");
41 imageLabel.setFont(Font.font(s: "Arial", FontWeight.BOLD, v: 14));
42 Button uploadButton = new Button(s: "Choose File");
43 uploadButton.setStyle("-fx-background-color: #2196F3; -fx-text-fill: white;");
44 ImageView imageView = new ImageView();
45 imageView.setFitWidth(100);
46 imageView.setFitHeight(100);
47 imageView.setPreserveRatio(true);
48 imageView.setStyle("-fx-border-color: black;");
49
50 HBox imageBox = new HBox(v: 10, uploadButton, imageView);
51 formGrid.add(imageLabel, i: 0, i1: 5);
52 formGrid.add(imageBox, i: 1, i1: 5);
53
54 // File chooser for image upload
55 FileChooser fileChooser = new FileChooser();
56 uploadButton.setOnAction(e -> {
57     File selectedFile = fileChooser.showOpenDialog(primaryStage);
58     if (selectedFile != null) {
59         Image image = new Image(selectedFile.toURI().toString());
60         imageView.setImage(image);
61     }
62 });
```

5. Gender Selection

Users select their gender using radio buttons in a **ToggleGroup** to ensure only one option can be selected at a time. It provides radio buttons for "Male" and "Female." It also ensures only one gender can be selected.

```

13
14 // Gender
15 Label genderLabel = new Label(s: "Gender:");
16 genderLabel.setFont(Font.font(s: "Arial", FontWeight.BOLD, v: 14));
17 ToggleGroup genderGroup = new ToggleGroup();
18 RadioButton maleButton = new RadioButton(s: "Male");
19 RadioButton femaleButton = new RadioButton(s: "Female");
20 maleButton.setToggleGroup(genderGroup);
21 femaleButton.setToggleGroup(genderGroup);
22
23 HBox genderBox = new HBox(v: 10, maleButton, femaleButton);
24 formGrid.add(genderLabel, i: 0, i1: 6);
25 formGrid.add(genderBox, i: 1, i1: 6);
26
27 // Submit Button
28 Button submitButton = new Button(s: "Submit");
29 submitButton.setStyle("-fx-background-color: #4CAF50; -fx-text-fill: white;");
30 submitButton.setPrefWidth(100);
31
32 VBox mainLayout = new VBox(v: 10, bannerPane, formGrid, submitButton);
33 mainLayout.setPadding(new Insets(v: 20));
34 mainLayout.setStyle("-fx-background-color: #e8f5e9;");
35

```

6. Submit Button

A button is added to submit the form data. When clicked, it validates the input and either shows an error or saves the data. It does the following processes:

Checks if any field is empty.

Shows an error if validation fails.

Saves the data into the `submittedData` list if validation passes.

Opens a new window to display the submitted data.

```

// Submit button action
submitButton.setOnAction(e -> {
    String name = nameField.getText();
    String fatherName = fatherNameField.getText();
    String email = emailField.getText();
    String city = cityField.getText();
    String address = addressArea.getText();
    String gender = maleButton.isSelected() ? "Male" : (femaleButton.isSelected() ? "Female" : "No Gender");

    if (name.isEmpty() || fatherName.isEmpty() || email.isEmpty() || city.isEmpty() || address.isEmpty() || gender.isEmpty()) {
        Alert alert = new Alert(Alert.AlertType.ERROR, s: "Please fill all fields!");
        alert.showAndWait();
        return;
    }

    // Store data
    submittedData.add(new String[]{name, fatherName, email, city, address, gender});

    // Show submitted data in new window
    showDataScreen();
});

```

7. Displaying Submitted Data

A new window (**Stage**) shows the submitted data. It opens a new window and lists all submitted data with labels for clarity.

```

63
64 private void showDataScreen() { 1 usage
65     Stage dataStage = new Stage();
66     dataStage.setTitle("Submitted Data");
67
68     VBox dataLayout = new VBox(v: 10);
69     dataLayout.setPadding(new Insets(v: 20));
70     dataLayout.setStyle("-fx-background-color: #ffffff;");
71
72     Label dataLabel = new Label(s: "Submitted Data:");
73     dataLabel.setFont(Font.font(s: "Arial", FontWeight.BOLD, v: 18));
74     dataLayout.getChildren().add(dataLabel);
75
76     for (String[] entry : submittedData) {
77         String entryString = String.format(
78             "Name: %s\nFather's Name: %s\nEmail: %s\nCity: %s\nAddress: %s\nGender: %s\n",
79             entry[0], entry[1], entry[2], entry[3], entry[4], entry[5]
80         );
81         Label entryLabel = new Label(entryString);
82         entryLabel.setStyle("-fx-border-color: gray; -fx-padding: 10px;");
83         dataLayout.getChildren().add(entryLabel);
84     }
85
86     Scene dataScene = new Scene(dataLayout, v: 400, v1: 400);
87     dataStage.setScene(dataScene);
88     dataStage.show();
89 }
90 }

```

8. Putting It All Together

Finally, the main layout (**VBox**) combines the banner, form, and submit button. It displays everything together on the screen and sets the application window size to 700x600 pixels.

```
VBox mainLayout = new VBox(v: 10, bannerPane, formGrid, submitButton);
mainLayout.setPadding(new Insets(v: 20));
mainLayout.setStyle("-fx-background-color: #e8f5e9;");
```

```
// Set the scene
Scene scene = new Scene(mainLayout, v: 700, v1: 600);
primaryStage.setScene(scene);
primaryStage.show();
}
```