**Dilpreet Singh**

**(23100264)**

**(Lab2)**

**Section - A**

import javafx.application.Application;

import javafx.stage.Stage;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.\*;

import javafx.geometry.Insets;

import javafx.geometry.Pos;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.beans.property.SimpleStringProperty;

import javafx.beans.property.SimpleIntegerProperty;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

public class Main extends Application {

private static final String STUDENT\_NAME = "Dilpreet Singh";

private static final String STUDENT\_ID = "23100264";

private static final String CURRENT\_DATE = LocalDate.now().format(DateTimeFormatter.ofPattern("2025-07-02"));

private TableView<Person> tableView;

private ObservableList<Person> data;

private TextField studentNameField;

private TextField studentIdField;

private TextField dateField;

private Label statusLabel;

@Override

public void start(Stage primaryStage) {

data = FXCollections.observableArrayList();

BorderPane mainLayout = new BorderPane();

VBox header = createHeader();

mainLayout.setTop(header);

VBox inputForm = createInputForm();

HBox buttonPanel = createButtonPanel();

tableView = createTableView();

VBox leftPanel = new VBox(10);

leftPanel.getChildren().addAll(inputForm, buttonPanel);

leftPanel.setPadding(new Insets(10));

leftPanel.setPrefWidth(300);

VBox rightPanel = new VBox(10);

Label tableLabel = new Label("Data Table View");

tableLabel.setStyle("-fx-font-size: 14px; -fx-font-weight: bold;");

rightPanel.getChildren().addAll(tableLabel, tableView);

rightPanel.setPadding(new Insets(10));

mainLayout.setLeft(leftPanel);

mainLayout.setCenter(rightPanel);

statusLabel = new Label("Ready for CRUD operations");

statusLabel.setStyle("-fx-background-color: #f0f0f0; -fx-padding: 5px;");

mainLayout.setBottom(statusLabel);

addSampleData();

Scene scene = new Scene(mainLayout, 900, 600);

primaryStage.setTitle("JavaFX CRUD Application - " + STUDENT\_NAME);

primaryStage.setScene(scene);

primaryStage.show();

}

private VBox createHeader() {

VBox header = new VBox(5);

header.setAlignment(Pos.CENTER);

header.setPadding(new Insets(10));

header.setStyle("-fx-background-color: #2c3e50;");

Label studentInfoLabel = new Label("Student: " + STUDENT\_NAME + " | ID: " + STUDENT\_ID + " | Date: " + CURRENT\_DATE);

studentInfoLabel.setStyle("-fx-font-size: 14px; -fx-font-weight: bold; -fx-text-fill: white;");

header.getChildren().add(studentInfoLabel);

return header;

}

private VBox createInputForm() {

VBox form = new VBox(10);

form.setPadding(new Insets(10));

form.setStyle("-fx-border-color: #bdc3c7; -fx-border-width: 1px; -fx-border-radius: 5px;");

Label formTitle = new Label("CRUD Operations Form");

formTitle.setStyle("-fx-font-size: 14px; -fx-font-weight: bold;");

studentNameField = new TextField();

studentNameField.setPromptText("Enter student name");

studentIdField = new TextField();

studentIdField.setPromptText("Enter student ID");

dateField = new TextField();

dateField.setPromptText("Enter date (YYYY-MM-DD)");

GridPane grid = new GridPane();

grid.setHgap(10);

grid.setVgap(10);

grid.add(new Label("Student Name:"), 0, 0);

grid.add(studentNameField, 1, 0);

grid.add(new Label("Student ID:"), 0, 1);

grid.add(studentIdField, 1, 1);

grid.add(new Label("Date:"), 0, 2);

grid.add(dateField, 1, 2);

form.getChildren().addAll(formTitle, grid);

return form;

}

private HBox createButtonPanel() {

HBox buttonPanel = new HBox(10);

buttonPanel.setAlignment(Pos.CENTER);

buttonPanel.setPadding(new Insets(10));

Button insertBtn = new Button("Insert");

Button updateBtn = new Button("Update");

Button deleteBtn = new Button("Delete");

Button viewDataBtn = new Button("View Data");

Button clearBtn = new Button("Clear");

String buttonStyle = "-fx-min-width: 80px; -fx-min-height: 30px;";

insertBtn.setStyle(buttonStyle + "-fx-background-color: #27ae60; -fx-text-fill: white;");

updateBtn.setStyle(buttonStyle + "-fx-background-color: #3498db; -fx-text-fill: white;");

deleteBtn.setStyle(buttonStyle + "-fx-background-color: #e74c3c; -fx-text-fill: white;");

viewDataBtn.setStyle(buttonStyle + "-fx-background-color: #9b59b6; -fx-text-fill: white;");

clearBtn.setStyle(buttonStyle + "-fx-background-color: #95a5a6; -fx-text-fill: white;");

insertBtn.setOnAction(e -> insertData());

updateBtn.setOnAction(e -> updateData());

deleteBtn.setOnAction(e -> deleteData());

viewDataBtn.setOnAction(e -> viewData());

clearBtn.setOnAction(e -> clearFields());

buttonPanel.getChildren().addAll(insertBtn, updateBtn, deleteBtn, viewDataBtn, clearBtn);

return buttonPanel;

}

private TableView<Person> createTableView() {

TableView<Person> table = new TableView<>();

TableColumn<Person, Integer> idCol = new TableColumn<>("ID");

idCol.setCellValueFactory(new PropertyValueFactory<>("id"));

idCol.setPrefWidth(50);

TableColumn<Person, String> studentNameCol = new TableColumn<>("Student Name");

studentNameCol.setCellValueFactory(new PropertyValueFactory<>("studentName"));

studentNameCol.setPrefWidth(150);

TableColumn<Person, String> studentIdCol = new TableColumn<>("Student ID");

studentIdCol.setCellValueFactory(new PropertyValueFactory<>("studentId"));

studentIdCol.setPrefWidth(120);

TableColumn<Person, String> dateCol = new TableColumn<>("Date");

dateCol.setCellValueFactory(new PropertyValueFactory<>("date"));

dateCol.setPrefWidth(120);

table.getColumns().addAll(idCol, studentNameCol, studentIdCol, dateCol);

table.setItems(data);

table.getSelectionModel().selectedItemProperty().addListener((obs, oldSelection, newSelection) -> {

if (newSelection != null) {

populateFields(newSelection);

}

});

return table;

}

private void addSampleData() {

data.addAll(

new Person(1, "Roman Reigns", "S001", "2024-01-15"),

new Person(2, "John Cena", "S002", "2024-01-16"),

new Person(3, "Brock Lesnar", "S003", "2024-01-17")

);

}

**Section – D-G**

private void insertData() {

if (validateInput()) {

int newId = data.size() + 1;

Person newPerson = new Person(

newId,

studentNameField.getText(),

studentIdField.getText(),

dateField.getText()

);

data.add(newPerson);

clearFields();

updateStatus("Data inserted successfully!");

}

}

private void updateData() {

Person selectedPerson = tableView.getSelectionModel().getSelectedItem();

if (selectedPerson != null && validateInput()) {

selectedPerson.setStudentName(studentNameField.getText());

selectedPerson.setStudentId(studentIdField.getText());

selectedPerson.setDate(dateField.getText());

tableView.refresh();

clearFields();

updateStatus("Data updated successfully!");

} else {

showAlert("Warning", "Please select a row to update.");

}

}

private void deleteData() {

Person selectedPerson = tableView.getSelectionModel().getSelectedItem();

if (selectedPerson != null) {

Alert confirmAlert = new Alert(Alert.AlertType.CONFIRMATION);

confirmAlert.setTitle("Confirm Delete");

confirmAlert.setHeaderText("Delete Record");

confirmAlert.setContentText("Are you sure you want to delete this record?");

if (confirmAlert.showAndWait().get() == ButtonType.OK) {

data.remove(selectedPerson);

clearFields();

updateStatus("Data deleted successfully!");

}

} else {

showAlert("Warning", "Please select a row to delete.");

}

}

private void viewData() {

if (data.isEmpty()) {

showAlert("Information", "No data available to view.");

} else {

updateStatus("Displaying " + data.size() + " records in the table.");

tableView.getSelectionModel().selectFirst();

}

}

private void clearFields() {

studentNameField.clear();

studentIdField.clear();

dateField.clear();

tableView.getSelectionModel().clearSelection();

}

private void populateFields(Person person) {

studentNameField.setText(person.getStudentName());

studentIdField.setText(person.getStudentId());

dateField.setText(person.getDate());

}

private boolean validateInput() {

if (studentNameField.getText().trim().isEmpty() ||

studentIdField.getText().trim().isEmpty() ||

dateField.getText().trim().isEmpty()) {

showAlert("Validation Error", "Please fill in all fields.");

return false;

}

return true;

}

private void updateStatus(String message) {

statusLabel.setText(message);

}

private void showAlert(String title, String message) {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(title);

alert.setHeaderText(null);

alert.setContentText(message);

alert.showAndWait();

}

public static class Person {

private SimpleIntegerProperty id;

private SimpleStringProperty studentName;

private SimpleStringProperty studentId;

private SimpleStringProperty date;

public Person(int id, String studentName, String studentId, String date) {

this.id = new SimpleIntegerProperty(id);

this.studentName = new SimpleStringProperty(studentName);

this.studentId = new SimpleStringProperty(studentId);

this.date = new SimpleStringProperty(date);

}

public int getId() { return id.get(); }

public void setId(int id) { this.id.set(id); }

public SimpleIntegerProperty idProperty() { return id; }

public String getStudentName() { return studentName.get(); }

public void setStudentName(String studentName) { this.studentName.set(studentName); }

public SimpleStringProperty studentNameProperty() { return studentName; }

public String getStudentId() { return studentId.get(); }

public void setStudentId(String studentId) { this.studentId.set(studentId); }

public SimpleStringProperty studentIdProperty() { return studentId; }

public String getDate() { return date.get(); }

public void setDate(String date) { this.date.set(date); }

public SimpleStringProperty dateProperty() { return date; }

}

public static void main(String[] args) {

launch(args);

}

}

Output-

A screenshot of a computer

AI-generated content may be incorrect.