import javafx.application.Application;  
import javafx.geometry.Insets;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.control.Label;  
import javafx.scene.layout.VBox;  
import javafx.stage.Stage;  
import javafx.animation.FadeTransition;  
import javafx.util.Duration;  
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
  
public class Assignment\_5 extends Application {  
 private QuizModel quizModel;  
 private Label questionLabel;  
 private List<Button> optionButtons;  
 private Button submitButton;  
 private Label scoreLabel;  
 private int currentQuestionIndex;  
 private int score;  
   
 public void start(Stage primaryStage) {  
 initializeComponents(primaryStage);  
 startQuiz();  
 }  
  
 private void initializeComponents(Stage primaryStage) {  
 quizModel = new QuizModel();  
  
 questionLabel = new Label();  
 optionButtons = createOptionButtons();  
 submitButton = createSubmitButton();  
 scoreLabel = new Label("Score: 0");  
  
 VBox layout = createLayout();  
 Scene scene = new Scene(layout, 400, 300);  
  
 configureStage(primaryStage, scene);  
 }  
  
 private VBox createLayout() {  
 VBox layout = new VBox(10);  
 layout.setPadding(new Insets(10));  
 layout.setStyle("-fx-background-color: lightblue; -fx-padding: 20px;");  
  
 Label titleLabel = new Label("Welcome to the Quiz!");  
 configureTitleLabel(titleLabel);  
  
 layout.getChildren().addAll(titleLabel, scoreLabel, questionLabel);  
 layout.getChildren().addAll(optionButtons);  
 layout.getChildren().add(submitButton);  
  
 return layout;  
 }  
  
 private List<Button> createOptionButtons() {  
 List<Button> buttons = new ArrayList<>();  
 for (int i = 0; i < 4; i++) {  
 buttons.add(new Button());  
 }  
 return buttons;  
 }  
  
 private Button createSubmitButton() {  
 Button button = new Button("Submit");  
 button.setOnAction(e -> submitAnswer());  
 return button;  
 }  
  
 private void startQuiz() {  
 currentQuestionIndex = 0;  
 score = 0;  
 updateScoreLabel(0);  
 nextQuestion();  
 }  
  
 private void nextQuestion() {  
 if (currentQuestionIndex < quizModel.getQuestionCount()) {  
 Question question = quizModel.getQuestion(currentQuestionIndex);  
 displayQuestion(question);  
 } else {  
 finishQuiz();  
 }  
 }  
  
 private void displayQuestion(Question question) {  
 questionLabel.setText(question.getQuestion());  
 List<Integer> optionIndices = createShuffledOptionIndices(question);  
  
 for (int i = 0; i < optionIndices.size() && i < optionButtons.size(); i++) {  
 int optionIndex = optionIndices.get(i);  
 Button optionButton = optionButtons.get(i);  
 optionButton.setText(question.getOption(optionIndex));  
 configureOptionButton(optionButton);  
 }  
  
 disableUnusedOptionButtons(optionIndices.size());  
 }  
  
 private List<Integer> createShuffledOptionIndices(Question question) {  
 List<Integer> optionIndices = new ArrayList<>();  
 for (int i = 0; i < question.getOptionCount(); i++) {  
 optionIndices.add(i);  
 }  
 Collections.shuffle(optionIndices);  
 return optionIndices;  
 }  
  
 private void configureOptionButton(Button optionButton) {  
 optionButton.setDisable(false);  
 optionButton.setOnAction(e -> submitAnswer());  
 }  
  
 private void disableUnusedOptionButtons(int usedButtonCount) {  
 for (int i = usedButtonCount; i < 4 && i < optionButtons.size(); i++) {  
 optionButtons.get(i).setText("");  
 optionButtons.get(i).setDisable(true);  
 optionButtons.get(i).setOnAction(null);  
 }  
 }  
  
 private void submitAnswer() {  
 if (currentQuestionIndex < quizModel.getQuestionCount()) {  
 Question currentQuestion = quizModel.getQuestion(currentQuestionIndex);  
  
 if (currentQuestion.isCorrectOption(2)) {  
 score++;  
 updateScoreLabel(score);  
 }  
 currentQuestionIndex++;  
 nextQuestion();  
 } else {  
 finishQuiz();  
 }  
 }  
  
 private void finishQuiz() {  
 questionLabel.setText("Quiz complete. Your score: " + score);  
  
 FadeTransition fadeTransition = new FadeTransition(Duration.seconds(2), questionLabel);  
 fadeTransition.setFromValue(0);  
 fadeTransition.setToValue(1);  
 fadeTransition.play();  
  
 submitButton.setDisable(true);  
 }  
  
 private void updateScoreLabel(int updatedScore) {  
 scoreLabel.setText("Score: " + updatedScore);  
 }  
  
 private void configureTitleLabel(Label titleLabel) {  
 titleLabel.setStyle("-fx-font-size: 24px; -fx-font-weight: bold;");  
 }  
  
 private void configureStage(Stage primaryStage, Scene scene) {  
 primaryStage.setTitle("Quiz App");  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 }  
  
 public static class QuizModel {  
 private final List<Question> questions;  
  
 public QuizModel() {  
 questions = new ArrayList<>();  
 questions.add(new Question("What is the capital of Japan?", new String[]{"Beijing", "Seoul", "Tokyo", "Shanghai"}, 2));  
 questions.add(new Question("Which gas do plants absorb from the atmosphere?", new String[]{"Oxygen", "Carbon Dioxide", "Nitrogen", "Hydrogen"}, 1));  
 questions.add(new Question("How many continents are there on Earth?", new String[]{"5", "6", "7", "8"}, 0));  
 questions.add(new Question("What is the largest animal on Earth?", new String[]{"African Elephant", "Giraffe", "Blue Whale", "Lion"}, 2));  
 }  
  
 public Question getQuestion(int index) {  
 return questions.get(index);  
 }  
  
 public int getQuestionCount() {  
 return questions.size();  
 }  
 }  
  
 public static class Question {  
 private final String question;  
 private final String[] options;  
 private final int correctOptionIndex;  
  
 public Question(String question, String[] options, int correctOptionIndex) {  
 this.question = question;  
 this.options = options;  
 this.correctOptionIndex = correctOptionIndex;  
 }  
  
 public String getQuestion() {  
 return question;  
 }  
  
 public String getOption(int index) {  
 return options[index];  
 }  
  
 public boolean isCorrectOption(int index) {  
 return index == correctOptionIndex;  
 }  
  
 public int getOptionCount() {  
 return options.length;  
 }  
 }  
  
 public static void main(String[] args) {  
 launch(args);  
 }  
}

A screenshot of a quiz

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a quiz

Description automatically generated

A screenshot of a computer quiz

Description automatically generated