**MINI PROJECT**

**ABSTRACT:**

The code implements a simple quiz application using Swing components in Java.Using java programming language we have created a quiz application. It will be very useful to school and college students.The questions will be asked and marks will be provided based upon the respective answers.

It manages a list of questions, allows the user to select answers, and calculates the final score at the end.The framework class extends JFrame and handles the GUI-related functionalities.

The Question class encapsulates information about individual questions.This application provides a basic structure for a quiz but can be expanded by adding more features like a timer, randomizing questions, or allowing users to save their scores.

**INTRODUCTION:**

The main method initializes the program. It invokes the EventQueue.invokeLater method to ensure the creation of the Swing components in the event-dispatching thread.It creates an instance of the framework class (which extends JFrame) and sets it visible, starting the application.

framework Class (Extending JFrame):This class represents the main frame of the quiz application.

**Attributes:**

private List<Question>questions;: Holds a list of Question objects.private int currentQuestionIndex;: Keeps track of the current question being displayed.

private int score;: Tracks the user's score.

**Components:**

JLabelquestionLabel;: Displays the question.JRadioButton[] optionButtons;: Array of radio buttons for answer choices.

JButtonsubmitButton;: Button to submit answers.

**Constructor framework():**

Initializes the Swing components and sets up the layout using GroupLayout.Calls initializeQuiz() to set up the list of questions and starts the quiz by displaying the first question.

**initializeQuiz() Method:**

Creates a list of Question objects with predefined questions, options, and correct answer indices.

Initializes the currentQuestionIndex and score variables.Calls showQuestion() to display the first question.

**showQuestion() Method:**

Retrieves the current question from the list and updates the question label and answer options on the GUI.

submitButtonActionPerformed(ActionEventevt) Method:Handles the action when the submit button is clicked.Checks the selected answer against the correct answer and updates the score accordingly.Moves to the next question if available or shows the final score.

**showResults() Method:**

Displays a message dialog showing the user's score after completing all the questions.Terminates the program using System.exit(0).

**Question Class:**

Represents a single question in the quiz.

**Attributes:**

private final String text;: Stores the question text.private final String[] options;: Array containing answer options.private final int correctOptionIndex;: Index of the correct answer in the options array.Constructor Question():Initializes the question text, answer options, and correct answer index.

**MODULE DESCRIPTION:**

**Overview:**

The "Quiz Application" module is a Java-based interactive program designed to present multiple-choice questions to users and evaluate their responses. It employs Java Swing components to create a user interface for displaying questions, accepting user input, and computing quiz scores.

**Components:**

framework Class (Extending JFrame): Manages the main frame of the application.Controls the display of questions, answer options, and submission of responses.

Question Class: Represents individual questions in the quiz.Stores question text, answer options, and correct answer indices.

**Functionality:**

Initialization:Loads a predefined set of questions with answer choices and correct answers.

Initializes the GUI elements and layout using Swing components.

Quiz Interaction:Displays one question at a time, allowing the user to select an answer.Tracks the user's score based on the correctness of the responses.

User Interface:Presents questions and answer options using labels and radio buttons.Provides a submit button to submit answers and progress through the quiz.Displays the final score in a message dialog after completing all questions.

Potential Enhancements:Addition of a timer for each question.Randomization of question order.Integration with a database to store user scores.Support for multiple quiz categories or difficulty levels.Improvement of GUI aesthetics and user experience.

**Dependencies:**

Java Swing for GUI components.Java Event Handling for user interactions.

**Future Scope:**

The module can be extended to incorporate more interactive features, a broader question bank, and enhanced user engagement to make it suitable for educational or entertainment purposes.

**PROGRAM:**

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;

import java.util.List;

import javax.swing.\*;

public class framemiletokilo {

public static void main(String[] args) {

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new framework().setVisible(true);

}

});

}

}

class framework extends JFrame

{

private List<Question> questions;

private int currentQuestionIndex;

private int score;

private JLabelquestionLabel;

private JRadioButton[] optionButtons;

private JButtonsubmitButton;

framework(){

questionLabel = new JLabel();

optionButtons = new JRadioButton[4];

ButtonGroupoptionGroup = new ButtonGroup();

for (int i = 0; i< 4; i++) {

optionButtons[i] = new JRadioButton();

optionGroup.add(optionButtons[i]);

}

submitButton = new JButton("Submit");

submitButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

submitButtonActionPerformed(e);

}

});

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

GroupLayout layout = new GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addComponent(questionLabel,GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(optionButtons[0],GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(optionButtons[1],GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(optionButtons[2],GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(optionButtons[3],GroupLayout.DEFAULT\_SIZE, GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(submitButton))

.addContainerGap()) );

layout.setVerticalGroup(

layout.createParallelGroup(GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addContainerGap()

.addComponent(questionLabel)

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(optionButtons[0])

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(optionButtons[1])

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(optionButtons[2])

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(optionButtons[3])

.addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)

.addComponent(submitButton)

.addContainerGap(GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

);

pack();

initializeQuiz();

setSize(300,300);

setVisible(true);

}

private void initializeQuiz() {

questions = new ArrayList<>();

questions.add(new Question("What is the capital of France?", new String[]{"Paris", "Berlin", "London", "Madrid"}, 0));

questions.add(new Question("Which planet is known as the Red Planet?", new String[]{"Earth", "Mars", "Venus", "Mercury"}, 1));

questions.add(new Question("What is the largest mammal?", new String[]{"Elephant", "Blue Whale", "Giraffe", "Lion"}, 1));

currentQuestionIndex = 0;

score = 0;

showQuestion();

}

private void showQuestion() {

Question currentQuestion = questions.get(currentQuestionIndex);

questionLabel.setText(currentQuestion.getText());

String[] options = currentQuestion.getOptions();

for (int i = 0; i< 4; i++) {

optionButtons[i].setText(options[i]);

optionButtons[i].setSelected(false);

}

}

private void submitButtonActionPerformed(ActionEventevt) {

int selectedOption = -1;

for (int i = 0; i< 4; i++) {

if (optionButtons[i].isSelected()) {

selectedOption = i;

break;

}

}

if(selectedOption!=1&&selectedOption==questions.get(currentQuestionIndex).getCorrectOptionIndex()) {

score++;

}

currentQuestionIndex++;

if (currentQuestionIndex<questions.size()) {

showQuestion();

} else {

showResults();

}

}

private void showResults() {

JOptionPane.showMessageDialog(this, "Quiz completed!\nYour Score: " + score + " out of " + questions.size());

System.exit(0);

}

}

class Question {

private final String text;

private final String[] options;

private final int correctOptionIndex;

public Question(String text, String[] options, int correctOptionIndex) {

this.text = text;

this.options = options;

this.correctOptionIndex = correctOptionIndex;

}

public String getText() {

return text;

}

public String[] getOptions() {

return options;

}

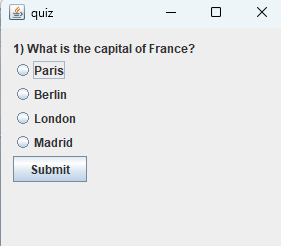
public int getCorrectOptionIndex() {

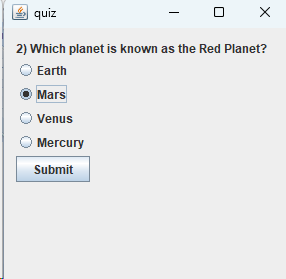
return correctOptionIndex;

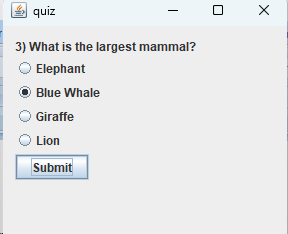
}

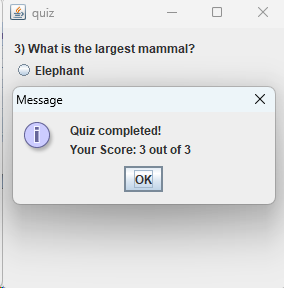
}

**OUTPUT:**









**Git link of project:** [**https://github.com/17MatchaRaniJ/oops-lab/blob/main/miniproject**](https://github.com/17MatchaRaniJ/oops-lab/blob/main/miniproject)**.**

**CONCLUSION:**

In conclusion, the provided Java code implements a basic quiz application using the Swing framework. The application is structured around two main classes: framework, which represents the main frame and user interface, and Question, which encapsulates individual quiz questions.

The application allows users to navigate through a predefined set of multiple-choice questions, selecting answers via radio buttons and submitting their responses. The program computes and displays the user's score at the end of the quiz.

While the code serves as a foundation for a quiz application, there are opportunities for further enhancement. Possible improvements include the addition of features such as timers, randomization of questions, database integration for score tracking, and improvements to the graphical user interface.

The modular design of the code promotes maintainability and extensibility. The framework class handles the GUI components and user interaction, while the Question class encapsulates the properties of each question, fostering a clear separation of concerns.

In summary, this quiz application code provides a starting point for creating interactive quizzes in Java, and its modular structure facilitates future expansion and customization to meet specific requirements.

**REFERENCE:**

For instance, if you've learned Java Swing from a book like "Head First Java" by Kathy Sierra and Bert Bates or from online tutorials on platforms like Codecademy, GeeksforGeeks, or Oracle's official Java documentation, you can reference those resources as part of your learning process. However, if this code is entirely your creation based on your understanding and knowledge, it can be referenced as an original work