

# **QUIZ APPLICATION**

## **A PROJECT REPORT**

*Submitted by*

**MOHAMMED RAASITH (2303811724321069)**

*in partial fulfillment of requirements for the award of the course*

**CGB1201 – JAVA PROGRAMMING**

*in*

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY**

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by  
AICTE, New Delhi)

**SAMAYAPURAM – 621 112**

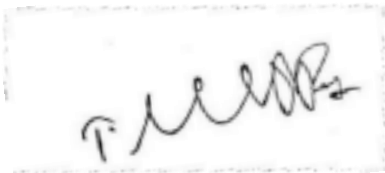
**DECEMBER, 2024**

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY (AUTONOMOUS)**

**SAMAYAPURAM – 621 112**

**BONAFIDE CERTIFICATE**

Certified that this project report on “ **QUIZ APPLICATION**” is the bonafide work of **MOHAMMED RAASITH (2303811724321069)** who carried out the project work during the academic year 2024 - 2025 under my supervision.



Signature

Dr.T.AVUDAIAPPAN M.E.,ph.D.,

**HEAD OF THE DEPARTMENT,**

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.

S

Submitted for the viva-voce examination held on 3.12.24



Signature

Mrs. S. GEETHA M.E.,

**HEAD OF THE DEPARTMENT,**

Department of Artificial Intelligence,

K. Ramakrishnan College of Technology,

Samayapuram, Trichy -621 112.



**INTERNAL EXAMINER**

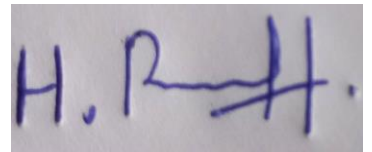


**EXTERNAL EXAMINER**

# DECLARATION

I declare that the project report on “**QUIZ APPLICATION**” is the result of original work done by me and best of my knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This project report is submitted on the partial fulfillment of the requirement of the award of the **CGB1201 – JAVA PROGRAMMING**.

**Signature**

A handwritten signature in blue ink, appearing to read 'H. R. H.', is written on a light-colored rectangular background.

**MOHAMMED RAASITH**

**Place:** Samayapuram

**Date:** 3/12/2024

## ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and indebtedness to our institution, **“K. Ramakrishnan College of Technology (Autonomous)”**, for providing us with the opportunity to do this project.

I extend our sincere acknowledgment and appreciation to the esteemed and honorable Chairman, **Dr. K. RAMAKRISHNAN, B.E.**, for having provided the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director, **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding our project and offering an adequate duration to complete it.

I would like to thank **Dr. N. VASUDEVAN, M.TECH., Ph.D.**, Principal, who gave the opportunity to frame the project to full satisfaction.

I thank **Dr.T.AVUDAIAPPAN, M.E.,Ph.D.**, Head of the Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for providing her encouragement in pursuing this project.

I wish to convey our profound and heartfelt gratitude to our esteemed project guide **Mrs.S.GEETHA M.E.**, Department of **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**, for her incalculable suggestions, creativity, assistance and patience, which motivated us to carry out this project.

I render our sincere thanks to the Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

## **VISION OF THE INSTITUTION**

To serve the society by offering top-notch technical education on par with global standards.

## **MISSION OF THE INSTITUTION**

- Be a centre of excellence for technical education in emerging technologies by exceeding the needs of industry and society.
- Be an institute with world class research facilities.
- Be an institute nurturing talent and enhancing competency of students to transform them as all- round personalities respecting moral and ethical values.

## **VISION AND MISSION OF THE DEPARTMENT**

To excel in education, innovation and research in Artificial Intelligence and Data Science to fulfill industrial demands and societal expectations.

Mission 1: To educate future engineers with solid fundamentals, continually improving teaching methods using modern tools.

Mission 2: To collaborate with industry and offer top-notch facilities in a conducive learning environment.

Mission 3: To foster skilled engineers and ethical innovation in AI and Data Science for global recognition and impactful research.

Mission 4: To tackle the societal challenge of producing capable professionals by instilling employability skills and human values.

## **PROGRAM EDUCATIONAL OBJECTIVES (PEOS)**

**PEO 1:** Compete on a global scale for a professional career in Artificial Intelligence and Data Science.

**PEO 2:** Provide industry-specific solutions for the society with effective communication and ethics.

**PEO 3:** Hone their professional skills through research and lifelong learning initiatives.

### **PROGRAM OUTCOMES**

Engineering students will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

### **PROGRAM SPECIFIC OUTCOMES (PSOs)**

- **PSO 1:** Capable of working on data-related methodologies and providing industry-focussed solutions.
- **PSO2:** Capable of analysing and providing a solution to a given real-world problem by designing an effective program.

## **ABSTRACT**

The proposed educational quiz application is designed to enhance both teaching and learning experiences by providing a seamless platform for quiz creation, student participation, and performance tracking. Educators can easily design quizzes using a range of question types, including multiple-choice, true/false, short answer, and even more complex formats like essay-type questions or fill-in-the-blank. Each question can be configured with correct answers, weighted scores, and time limits to ensure comprehensive assessment. Additionally, educators can set randomization for questions and answer options to create a more dynamic testing environment. Once a quiz is deployed, students can engage with the content by selecting or typing their responses. The application immediately evaluates answers against the correct ones and generates a score, allowing for instant feedback on the student's performance. In cases of incorrect answers, the app provides tailored explanations or hints to guide the student toward understanding the material. For more complex questions or assignments, like short answers or essays, teachers can manually review responses and offer detailed feedback.



## TABLE OF CONTENTS

<b>CHAPTER No.</b>	<b>TITLE</b>	<b>PAGE No.</b>
	<b>ABSTRACT</b>	<b>viii</b>
<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 INTRODUCTION	1
	1.2 OBJECTIVE	1
<b>2</b>	<b>PROJECT METHODOLOGY</b>	<b>2</b>
	2.1 PROPOSED WORK	2
	2.2 BLOCK DIAGRAM	3
<b>3</b>	<b>JAVA PROGRAMMING CONCEPTS</b>	<b>4</b>
	3.1 CORE BUILDING BLOCKS OF THE GUI	<b>4</b>
	3.2 USER INTERACTION FEEDBACK	<b>5</b>
<b>4</b>	<b>MODULE DESCRIPTION</b>	<b>6</b>
	4.1 AWT (ABSTRACT WINDOW TOOLKIT):	6
	4.2 EVENT HANDLING	7
	4.3 ARRAYS	7
	4.4 SCORE AND FEEDBACK:	8
<b>5</b>	<b>CONCLUSION</b>	<b>9</b>
	<b>REFERENCES</b>	<b>10</b>
	<b>APPENDICES</b>	<b>11</b>
	Appendix A – Source code	<b>12</b>
	Appendix B – Screen shots	<b>15</b>

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 INTRODUCTION**

The development of a quiz application for educational purposes aims to enhance the learning experience by integrating interactive assessments into the educational process. In today's digital age, quizzes are an effective tool for reinforcing learning, testing knowledge, and providing instant feedback. This application will enable educators to easily create and manage quizzes, ranging from simple multiple-choice questions to more complex assessments, while also ensuring that students receive immediate, personalized feedback based on their responses.

### **1.2 OBJECTIVE**

The objective of the quiz application is to provide an efficient and user-friendly tool for both educators and students by simplifying quiz creation, response handling, scoring, and feedback. Educators will be able to easily design customizable quizzes with various question types, such as multiple-choice and short answer, tailored to their teaching needs. Students can submit their answers through an intuitive interface that supports different response formats. The application will automatically grade quizzes in real-time and offer instant feedback on performance. Additionally, personalized feedback will help students understand their mistakes and improve their knowledge. The system will also track student performance over time, enabling both educators and students to identify areas for improvement.

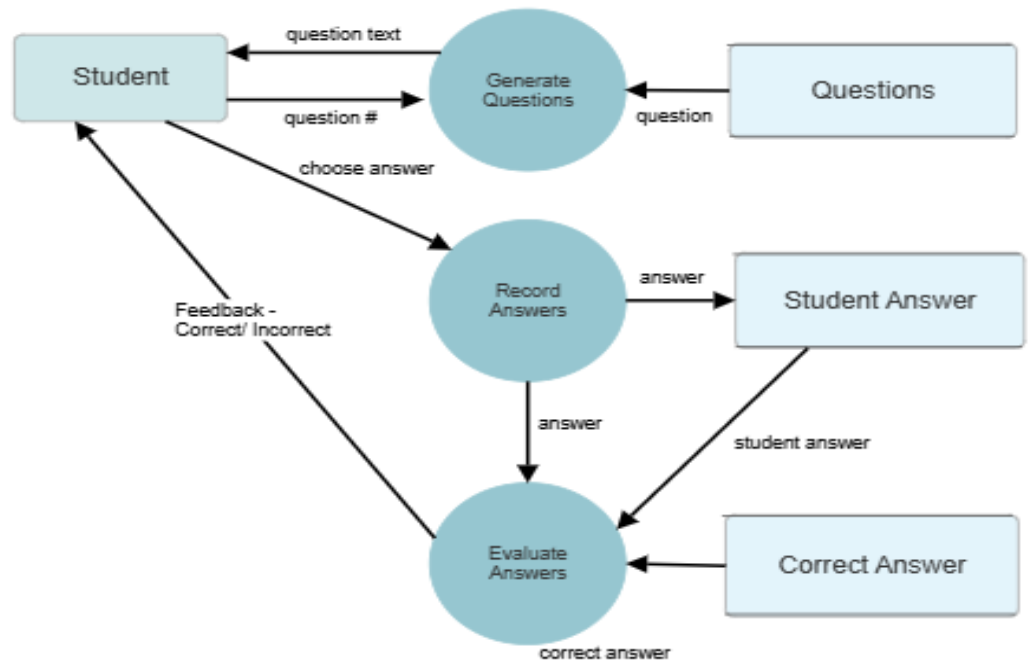
## **CHAPTER 2**

### **PROJECT METHODOLOGY**

#### **2.1 PROPOSED WORK**

1. Requirement Analysis: Gather input from educators and students on desired features like quiz creation, responses, scoring, and feedback.
2. Design: Create a user-friendly interface and system structure to support various question types and automated grading.
3. Development: Build the application to handle quizzes, user responses, scoring, and feedback.
4. Testing: Ensure usability and accuracy through testing, making improvements based on feedback.
5. Deployment and Maintenance: Launch the app and provide regular updates.

## 2.2 BLOCK DIAGRAM



## **CHAPTER 3**

### **JAVA PROGRAMMING CONCEPTS**

#### **3.2 Core Building Blocks of the GUI**

##### **1. Frame Class: Creating a Window for the GUI**

The Frame class (usually JFrame) is used to create the main window in a GUI application. It serves as the container for all user interface components like buttons, labels, and checkboxes. The frame's size, title, and visibility can be configured, and it holds the layout where all other elements are placed.

##### **2. ActionListener: Responding to User Actions**

The ActionListener interface is crucial for capturing user interactions, such as button clicks or menu selections. The actionPerformed() method is invoked whenever an action occurs, and it defines the response. For instance, in a quiz app, clicking a "Next" button could display the next question based on the user's action.

##### **3. Arrays: Storing Questions and Answers**

Arrays in Java allow you to store multiple data values of the same type. For a quiz application, arrays can hold questions (String[]), options (String[][]), and correct answers (int[]). This makes it easier to loop through questions and evaluate answers without repeating code.

##### **4. Checkboxes: Letting Users Select One Option from Multiple Choices**

Checkboxes are used to allow users to select answers from multiple options. In a quiz app, each checkbox represents an answer, and you can use a CheckboxGroup to ensure only one answer is selected at a time. The selected answer can then be checked against the correct options to evaluate the user's choice.

### **3.2 User Interaction and Performance Feedback**

#### **1. FlowLayout: Arranging the UI Components**

FlowLayout arranges components sequentially from left to right, and when space is exhausted, it moves components to the next row. This layout is useful for simple interfaces where precise positioning isn't necessary. It's ideal for dynamically resizing components as the window adjusts, like buttons, labels, and checkboxes.

#### **2. Labels: Displaying Text**

Labels display static or dynamic text in a GUI. In a quiz, labels show the current question, instructions, or feedback to the user. Labels can be easily updated to show new information as the user interacts with the quiz, offering clear communication without user interaction.

#### **3. Button: Triggers Actions Like Next or Submit**

A Button (JButton) is an interactive UI component that responds to user clicks. It can trigger actions such as moving to the next question or submitting answers. Buttons are linked to an ActionListener, which defines how the program should respond when the user clicks it, like updating the quiz content.

#### **4. GetFeedback: Method to Provide Feedback Based on Score**

The GetFeedback method provides the user with feedback after completing the quiz. It evaluates the user's score and delivers a response, such as "Excellent!" or "Better luck next time!" This feedback helps users understand their performance and motivates improvement. It can be displayed on-screen or in a dialog box.

## **CHAPTER 4**

### **MODULE DESCRIPTION**

#### **4.1 AWT (ABSTRACT WINDOW TOOLKIT);**

AWT (Abstract Window Toolkit) is a set of classes provided by Java to create graphical user interfaces (GUIs). It provides basic components for building windows, handling events, and interacting with users.

1. **Frame:** The Frame class in AWT is used to create the main window for the application. It serves as the container for other UI components like buttons, labels, and checkboxes. You define the window size, title, and layout using this class, and add all necessary components to the frame for user interaction.
2. **Label:** A Label is a non-interactive component that displays text to the user. In a quiz application, it is commonly used to display questions, feedback, and the current score. Labels are static by default but can be updated dynamically during the quiz to show new information.
3. **Checkbox:** A Checkbox is an interactive component that allows the user to select one or more options from a list. In a quiz, checkboxes are typically used to let the user select an answer choice. You can either allow multiple selections or restrict it to a single selection using a CheckboxGroup.
4. **Button:** A Button in AWT is a clickable component that triggers an action when pressed. It is commonly used in a quiz to move to the next question or submit answers. A button responds to user interactions, such as clicks, and is associated with an ActionListener to define the behavior upon clicking.
5. **CheckboxGroup:** A CheckboxGroup is a container that ensures only one checkbox from a group can be selected at a time. This is useful in scenarios where you want the user to select just one answer option from a list. By grouping the checkboxes together, it enforces the "single selection" rule.

6. **FlowLayout:** FlowLayout is a simple layout manager that arranges components in a left-to-right flow, one after the other. When the window size is adjusted, components automatically flow to the next row. This layout is easy to use for simple interfaces where precise control over the placement of components is not required.

## 4.2 Event Handling

Event handling is the mechanism that allows an application to respond to user interactions such as button clicks, mouse movements, or keyboard presses. It allows the application to remain dynamic and interactive.

1. **ActionListener:** The ActionListener interface is used to handle user actions like clicking a button. In a quiz application, an ActionListener is attached to buttons (like "Next") to define the action that should occur when the user clicks the button. For example, the "Next" button may trigger the display of the next question or update the user's score.
2. **ActionEvent:** An ActionEvent represents an event that occurs when a user interacts with a component that triggers an action, like pressing a button. When the button is clicked, an ActionEvent is fired, and the actionPerformed() method in the ActionListener is called. The event contains information about the source of the action, allowing the program to respond accordingly.

## 4.3 ARRAYS

Arrays are used to store multiple values of the same type in a single variable. In a quiz application, arrays help manage questions, options, and answers efficiently.

1. **String[] questions:** This array stores all the quiz questions as strings. Each question is stored as a separate element in the array, and the index of the array corresponds to the question number. This allows for easy access to each question during the quiz and enables the dynamic updating of the displayed question.
2. **String[][] options:** This two-dimensional array stores the answer options for each question. The first dimension represents the question number, and the second dimension stores the



multiple options for that question. For example, `options[0]` might store the options for the first question. This structure allows easy access to the answer choices when displaying questions.

3. `int[] answers`: This array stores the index of the correct answer for each question. The index corresponds to the question number, and each element holds the index of the correct answer from the options array. For example, if the correct answer to the first question is the second option, `answers[0]` would be set to 1.

## 4.4 SCORE AND FEEDBACK

The scoring mechanism is a crucial part of any quiz application, and feedback helps users understand their performance.

1. The score is updated based on correct answers: The score is calculated as the user progresses through the quiz. Every time the user selects the correct answer, the score is incremented. The score is stored in a variable and updated dynamically as the quiz is completed. At the end of the quiz, the total score represents the user's performance.
2. `getFeedback()` provides feedback based on the final score: The `getFeedback()` method is used to give the user feedback after they complete the quiz. It checks the score and provides appropriate feedback, such as "Excellent!" for a high score or "Try Again!" for a lower score. The feedback is displayed to the user, helping them understand how well they did and offering encouragement or suggestions.

## **CHAPTER 5**

### **CONCLUSION**

In conclusion, the educational quiz application is designed to offer a seamless and interactive learning experience by efficiently managing quiz creation, user responses, scoring, and feedback. With a modular structure, it organizes key components like question handling, quiz flow, user interaction, scoring, and feedback into separate, manageable sections. This ensures that the application runs smoothly and can be easily maintained or expanded in the future. The ability to handle multiple questions, options, and answers provides flexibility, allowing the quiz to be tailored to various educational purposes.

The immediate feedback provided to users after each question or at the end of the quiz not only reinforces learning but also highlights areas for improvement. This timely feedback is essential for user motivation and better retention of knowledge. The modular design further supports scalability, meaning new features—such as additional question formats, user profiles, or advanced scoring systems—can be integrated without disrupting the existing flow. Overall, this educational quiz application combines functionality and adaptability to create a dynamic, user-friendly learning tool that can evolve over time to meet changing educational needs.

## REFERENCES:

### Websites:

1. Oracle Java Documentation

The official Java documentation from Oracle provides comprehensive details on Java classes, methods, and libraries for GUI development, including AWT and Swing.

- [Oracle Java Documentation](#)

2. GeeksforGeeks - Java GUI Programming

A detailed tutorial on Java GUI programming, focusing on AWT (Abstract Window Toolkit) and Swing, which are key to building a quiz application.

- [GeeksforGeeks Java GUI Programming](#)

### YouTube Channels:

1. Programming with Mosh - Java Tutorial for Beginners

This channel offers a comprehensive Java tutorial for beginners, covering Java basics and GUI programming with AWT and Swing.

- [Programming with Mosh - Java Tutorial for Beginners](#)

2. BroCode - Java GUI Programming with AWT

BroCode provides tutorials on building GUI applications in Java, specifically using AWT components like buttons, labels, and checkboxes, which are key to developing the quiz app.

- [BroCode - Java GUI Programming with AWT](#)

## **APPENDICES**

### **APPENDIX A – SOURCE CODE**

```
import java.awt.*;
import java.awt.event.*;
public class QuizApp extends Frame implements ActionListener {
    // Question, options, and correct answer.
    String[] questions = {
        "What is the capital of France?",
        "Which planet is known as the Red Planet?",
        "Who wrote 'Romeo and Juliet'?"
    };
    String[][] options = {
        {"Berlin", "Madrid", "Paris", "Rome"},
        {"Earth", "Mars", "Jupiter", "Saturn"},
        {"Shakespeare", "Dickens", "Hemingway", "Austen"}
    };
    int[] answers = {2, 1, 0}; // Correct answers' indexes (0-based)
    int currentQuestion = 0;
    int score = 0;
    // UI components
    Label questionLabel;
    CheckboxGroup optionGroup;
    Checkbox option1, option2, option3, option4;
    Button nextButton;
    Label scoreLabel;
    Label feedbackLabel; // Label for feedback
```

```

public QuizApp() {
    setTitle("Quiz Application");
    setSize(400, 350); // Increased size for feedback message
    setLayout(new FlowLayout());
    questionLabel = new Label(questions[currentQuestion]);
    add(questionLabel);
    optionGroup = new CheckboxGroup();
    option1 = new Checkbox(options[currentQuestion][0], optionGroup, false);
    option2 = new Checkbox(options[currentQuestion][1], optionGroup, false);
    option3 = new Checkbox(options[currentQuestion][2], optionGroup, false);
    option4 = new Checkbox(options[currentQuestion][3], optionGroup, false)
    add(option1);
    add(option2);
    add(option3);
    add(option4);
    nextButton = new Button("Next");
    nextButton.addActionListener(this);
    add(nextButton);
    scoreLabel = new Label("Score: 0");
    add(scoreLabel);
    feedbackLabel = new Label(""); // Initial empty feedback
    add(feedbackLabel);
    setVisible(true);
}

@Override
public void actionPerformed(ActionEvent e) {
    // Check the answer and update the score
    int selectedOption = -1;

```

```

if (option1.getState()) {
    selectedOption = 0;
} else if (option2.getState()) {
    selectedOption = 1;
} else if (option3.getState()) {
    selectedOption = 2;
} else if (option4.getState()) {
    selectedOption = 3;
}
// Check the selected answer
if (selectedOption == answers[currentQuestion]) {
    score++;
}
// Update the score
scoreLabel.setText("Score: " + score);
// Go to the next question
currentQuestion++;
// If there are more questions, update the question and options
if (currentQuestion < questions.length) {
    questionLabel.setText(questions[currentQuestion]);
    option1.setLabel(options[currentQuestion][0]);
    option2.setLabel(options[currentQuestion][1]);
    option3.setLabel(options[currentQuestion][2]);
    option4.setLabel(options[currentQuestion][3]);
    // Clear previous selection
    optionGroup.setSelectedCheckbox(null);
} else {
    // End of quiz, show feedback
    nextButton.setEnabled(false);
}

```

```

        questionLabel.setText("Quiz Finished!");
        scoreLabel.setText("Final Score: " + score);
        feedbackLabel.setText(getFeedback(score));
    }
}

// Method to generate feedback based on the score
private String getFeedback(int score) {
    if (score == 3) {
        return "Excellent! You got all answers correct.";
    } else if (score == 2) {
        return "Good job! You got 2 out of 3 correct.";
    } else if (score == 1) {
        return "Needs improvement. You got 1 out of 3 correct.";
    } else {
        return "Better luck next time! You got 0 out of 3 correct.";
    }
}

public static void main(String[] args) {
    new QuizApp();
}
}

```

## APPENDIX B - SCREENSHOTS

