QuizApp

📌 **Abstract**

This project presents a beginner-friendly **Java-based Quiz Application** developed using the Swing framework. The application allows users to take a timed multiple-choice quiz with real-time feedback and score calculation. It demonstrates core programming concepts such as object-oriented design, event-driven programming, GUI development, and user input handling. The project is designed to be modular, scalable, and visually engaging, making it ideal for learners building their first portfolio-ready application.

📓 Introduction

In the digital learning era, quiz applications have become a popular tool for self-assessment and interactive education. This project aims to create a simple yet functional quiz app using Java, focusing on usability, performance, and clean code practices. The app features a graphical interface, a countdown timer, and a scoring system, making it both educational and engaging. It serves as a hands-on exercise in applying Java fundamentals and GUI development techniques.

□ Tools & Technologies Used

TOOLS	TECHNOLOGY
Java (JDK 8+)	Core programming language
Swing (javax.swing)	GUI development
AWT (java.awt)	Layout and event handling
IntelliJ IDEA	Integrated Development Environment
ImageIcon	Displaying banner image
JFrame, JLabel, JButton, JRadioButton	UI Components

🛠 Steps Involved in Building the Project

1. **Project Setup**

- Created a new Java project in IntelliJ IDEA named QuizApp-master.
- Organized source files under quiz/app package for modularity.

2. **Designing the GUI**

- Used JFrame to create the main window.
- Added components like JLabel for questions, JRadioButton for options, and JButton for navigation.
- Integrated a banner image using ImageIcon for visual appeal.

3. **Question Bank Initialization**

- Stored 10 questions in a 2D array with four options each.
- Defined a separate array for correct answers and user-selected answers.

4. **Timer Implementation**

- Added a static timer variable .
- Used Thread.sleep() and repaint () to update the countdown dynamically.

5. **Event Handling**

- Implemented ActionListener to respond to button clicks (Next, Submit).
- Captured user input and stored selected answers.

6. **Score Calculation**

- Compared user answers with correct answers.
- Displayed final score using a separate Score.java class.

7. **Navigation & Flow**

- Created Login.java for username input.
- Added Rules.java to display quiz instructions.
- Linked all screens for smooth user flow.

✅ Conclusion

The Java QuizApp project successfully demonstrates how core programming concepts can be applied to build an interactive and functional application. Through this project, I gained hands-on experience in GUI design, event handling, and modular coding practices. It also helped reinforce the importance of user experience and clean code structure. Future enhancements could include category-based quizzes, persistent leaderboards, and a JavaFX-based modern interface. This project marks a significant step in my journey toward becoming a confident and capable software developer.