Detailed Code Explanation for QuizApp

1. Overview

The **QuizApp** is a Java Swing application designed to test high school students (grades 10–12) in four subjects: Literature, Biology, Chemistry, and Mathematics. The quiz consists of 25 questions, with each grade level having its own unique set of questions and a specified time limit per question (Grade 10: 60 seconds, Grade 11: 45 seconds, and Grade 12: 30 seconds). The application uses a modern, colorful interface to make the quiz visually appealing and user friendly.

---

2. Class Structure and Main Components

2.1 The `QuizApp` Class

- **Extends JFrame:**
  The `QuizApp` class extends `JFrame`, making it a windowed application. It sets up the main window properties (size, title, close operation, and centering).

Inner `Question` Class:

```
 static class Question {
    String subject;
    String questionText;
    String[] options;
    int correctAnswer; // index of correct option
    int timeLimit; // in seconds

    public Question(String subject, String questionText, String[] options, int correctAnswer, int
timeLimit) {
        this.subject = subject;
        this.questionText = questionText;
        this.options = options;
        this.correctAnswer = correctAnswer;
        this.timeLimit = timeLimit;
    }
}
```

- This inner class encapsulates all necessary data for each quiz question:
  - **subject:** The category (Literature, Biology, etc.).

- **questionText:** The text of the question.
- **options:** An array of four answer choices.
- **correctAnswer:** The index (0–3) of the correct option.
- **timeLimit:** The number of seconds allocated to answer the question.

---

3. User Interface Layout

The application uses a **CardLayout** to switch between different screens (start, quiz, result).

3.1 Main Panel and CardLayout

- **CardLayout:**
  The `mainPanel` uses a `CardLayout` which allows easy switching between panels by their names (e.g., "startPanel", "quizPanel", "resultPanel").

- **Adding Panels:**
  The start panel and quiz panel are created and added to the `mainPanel` in the constructor.

---

4. Start Panel

4.1 Purpose

The **start panel** is where the user selects their grade level. Only grades 10, 11, or 12 are allowed. The user then clicks the "Start Quiz" button to begin.

4.2 Components and Layout

- **Title Label:**
  A large, centered title welcomes the user.

- **Grade Selection:**
  A `JComboBox` lets the user choose between grades 10, 11, and 12.

- **Start Button:**
  A button that, when clicked, stores the selected grade, loads the corresponding questions, resets counters (such as score and question index), and switches the view to the quiz panel.

4.3 Code Snippet

```
// Grade selection components
JLabel gradeLabel = new JLabel("Select Your Grade Level (10, 11, or 12):");
gradeLabel.setFont(new Font("Verdana", Font.PLAIN, 22));
// ...
Integer[] grades = {10, 11, 12};
gradeComboBox = new JComboBox<>(grades);
// ...
JButton startButton = new JButton("Start Quiz");
// When the start button is clicked:
startButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        selectedGrade = (Integer) gradeComboBox.getSelectedItem();
        loadQuestions();
        currentQuestionIndex = 0;
        score = 0;
        showQuestion();
        cardLayout.show(mainPanel, "quizPanel");
    }
});
```

---

5. Quiz Panel

5.1 Purpose

The **quiz panel** is where each question is displayed along with:

- A label showing the current subject.
- The question text (formatted with HTML for proper wrapping).
- Four multiple-choice options (using radio buttons).
- A countdown timer.
- A "Next" button to proceed to the next question.

5.2 Layout

- **Top Panel:**
  Displays the subject and the timer. Uses a dark blue background and white text for contrast.

- **Center Panel:**
  Contains the question and its four answer options. The question is displayed in a larger font and the options are spaced vertically.

- **Bottom Panel:**
  Contains the "Next" button. When clicked, it checks the answer and advances to the

next question.

5.3 Code Snippet

```
// Top panel for subject and timer:
subjectLabel = new JLabel("Subject: ", SwingConstants.LEFT);
subjectLabel.setFont(new Font("Verdana", Font.BOLD, 26));
timerLabel = new JLabel("Time: ", SwingConstants.RIGHT);
timerLabel.setFont(new Font("Verdana", Font.BOLD, 26));
// ...

// Center panel for question text and options:
questionLabel = new JLabel("Question", SwingConstants.CENTER);
questionLabel.setFont(new Font("Verdana", Font.PLAIN, 24));
optionsGroup = new ButtonGroup();
optionButtons = new JRadioButton[4]; // four options
for (int i = 0; i < optionButtons.length; i++) {
    optionButtons[i] = new JRadioButton();
    optionButtons[i].setFont(new Font("Verdana", Font.PLAIN, 22));
    optionsGroup.add(optionButtons[i]);
}

// Next button action:
nextButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        checkAnswer();
        nextQuestion();
    }
});
```

---

6. Loading Questions

The `loadQuestions()` method selects the correct set of questions based on the grade level. It calls one of the following methods:

- **`loadGrade10Questions()`**
  Loads 25 questions for Grade 10 with a time limit of 60 seconds per question.
  The questions are divided into:
    - Literature (7 questions)
    - Biology (6 questions)
    - Chemistry (6 questions)
    - Mathematics (6 questions)

- **`loadGrade11Questions()`**
  Loads a different set of 25 questions for Grade 11 with a 45-second time limit per question.
  The distribution across subjects remains the same.

- **`loadGrade12Questions()`**
  Loads a unique set of 25 questions for Grade 12 with a 30-second time limit per question.
  Again, the distribution among the four subjects is similar.

Each of these methods creates an `ArrayList<Question>`, adds sample questions (which you can replace with your own content), and returns the list.

Example for Grade 10

```
private ArrayList<Question> loadGrade10Questions() {
    ArrayList<Question> list = new ArrayList<>();
    int timeLimit = 60;
    // Literature questions
    list.add(new Question("Literature", "Grade 10 Literature Q1: Who wrote 'Romeo and Juliet'?",
            new String[]{"William Shakespeare", "Charles Dickens", "Mark Twain", "Jane Austen"}, 0, timeLimit));
    // ... (other questions)
    return list;
}
```

---

7. Quiz Functionality

7.1 `showQuestion()` Method

- **Purpose:**
  Displays the current question on the quiz panel.

- **How It Works:**

  1. Checks if there are more questions to display.
  2. Retrieves the current `Question` object.
  3. Sets the subject and question text (wrapped in HTML for formatting).
  4. Sets each option text on the radio buttons.
  5. Clears any previous selection.
  6. Initializes and starts a countdown timer using a `Timer` object. The timer updates every second, and if the time runs out, it automatically calls `nextQuestion()`.

Code Snippet

```
private void showQuestion() {
    if (currentQuestionIndex < questions.size()) {
        Question q = questions.get(currentQuestionIndex);
        subjectLabel.setText("Subject: " + q.subject);
        questionLabel.setText("<html><body style='width:800px;'>" + q.questionText +
"</body></html>");
        for (int i = 0; i < optionButtons.length; i++) {
            optionButtons[i].setText(q.options[i]);
            optionButtons[i].setSelected(false);
        }
        optionsGroup.clearSelection();

        remainingTime = q.timeLimit;
        timerLabel.setText("Time: " + remainingTime + " sec");

        if (timer != null) { timer.stop(); }
        timer = new Timer(1000, new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                remainingTime--;
                timerLabel.setText("Time: " + remainingTime + " sec");
                if (remainingTime <= 0) {
                    timer.stop();
                    nextQuestion();
                }
            }
        });
        timer.start();
    } else {
        showResult();
    }
}
```

7.2 `checkAnswer()` Method

- **Purpose:**
  Checks whether the selected answer matches the correct answer for the current question.

- **How It Works:**

    1. Loops through the radio buttons to find which option is selected.
    2. Compares the selected option's index to the `correctAnswer` index in the current `Question` object.
    3. Increments the score if the answer is correct.

```
private void checkAnswer() {
    if (currentQuestionIndex < questions.size()) {
        Question q = questions.get(currentQuestionIndex);
        int selectedIndex = -1;
        for (int i = 0; i < optionButtons.length; i++) {
            if (optionButtons[i].isSelected()) {
                selectedIndex = i;
                break;
            }
        }
        if (selectedIndex == q.correctAnswer) {
            score++;
        }
    }
}
```

7.3 `nextQuestion()` Method

- **Purpose:**
  Advances the quiz to the next question.

- **How It Works:**

  1. Stops the current timer.
  2. Increments the `currentQuestionIndex`.
  3. If there are more questions, calls `showQuestion()` to display the next one.
  4. If there are no more questions, calls `showResult()` to display the final score.

```
private void nextQuestion() {
    if (timer != null) { timer.stop(); }
    currentQuestionIndex++;
    if (currentQuestionIndex < questions.size()) {
        showQuestion();
    } else {
        showResult();
    }
}
```

7.4 `showResult()` Method

- **Purpose:**
  Displays the final result of the quiz, including the score and an option to restart.

- **How It Works:**

  1. Creates a new result panel.
  2. Displays the score in a large font.
  3. Provides a "Restart Quiz" button that, when clicked, switches back to the start panel.

```
private void showResult() {
    resultPanel = new JPanel(new BorderLayout());
    resultPanel.setBackground(new Color(240, 248, 255));
    JLabel resultLabel = new JLabel("Your Score: " + score + " / " + questions.size(),
SwingConstants.CENTER);
    resultLabel.setFont(new Font("Verdana", Font.BOLD, 36));
    resultLabel.setForeground(new Color(25, 25, 112));
    resultPanel.add(resultLabel, BorderLayout.CENTER);

    JButton restartButton = new JButton("Restart Quiz");
    restartButton.setFont(new Font("Verdana", Font.BOLD, 26));
    restartButton.setBackground(new Color(65, 105, 225));
    restartButton.setForeground(Color.WHITE);
    restartButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            cardLayout.show(mainPanel, "startPanel");
        }
    });
    resultPanel.add(restartButton, BorderLayout.SOUTH);

    mainPanel.add(resultPanel, "resultPanel");
    cardLayout.show(mainPanel, "resultPanel");
}
```

---

8. Application Entry Point

Main Method

- **Purpose:**
  Launches the application on the Swing Event Dispatch Thread (EDT) to ensure thread safety.

- **How It Works:**
  The `main` method uses `SwingUtilities.invokeLater` to create and display an

instance of `QuizApp`.

```
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            new QuizApp().setVisible(true);
        }
    });
}
```

---

9. Summary

- **Structure:**
  The application is organized into a main class (`QuizApp`) that extends `JFrame` with an inner `Question` class to encapsulate question data.

- **User Interface:**
  Uses a CardLayout to manage three screens: start, quiz, and result. Each panel is designed with modern, appealing colors and fonts.

- **Functionality:**
  - **Start Panel:** Allows the user to select a grade (10–12) and then loads the corresponding questions.
  - **Quiz Panel:** Displays each question with a countdown timer; if time runs out, the quiz automatically advances.
  - **Answer Checking:** The app verifies the selected option against the correct answer.
  - **Result Panel:** Shows the final score and provides an option to restart the quiz.