**Online Quiz Application**

**Features:**

**1. Dynamic Questions:**

- The code supports dynamic addition of questions, providing flexibility for different quizzes.

- Sample questions cover various topics related to Java programming and computer science.

**2. User-Friendly Interaction:**

- The quiz interface guides the user through each question and validates input.

- Clear messages and formatting enhance the user experience.

**3. Result Display:**

- The `Result` class produces a structured and informative display of the user's performance.

- Performance categories help interpret the percentage result easily.

**Design Choices:**

**1. Question Class:**

**- Text, Options, and Correct Option Index:**

- The `Question` class encapsulates a question's text, options, and the index of the correct option.

- Using a list for options allows flexibility in the number of choices.

**- Private Access Modifiers:**

- Data members are private, promoting encapsulation and ensuring controlled access.

- Getter methods provide read-only access to the class's internal state.

**2. Quiz Class:**

- **List of Questions:**

- Accepts a list of questions as a parameter in the constructor, allowing flexibility and reuse.

- Separation of Concerns:

- The `takeQuiz` method is responsible for presenting questions and recording results.

- A private `getUserAnswer` method handles user input validation.

- **Result Object:**

- Creates a `Result` object to display and calculate quiz results.

- Utilizes composition to achieve modularity.

**3. Result Class:**

**- Performance Thresholds:**

- Performance thresholds (e.g., >80% for "Good") are hardcoded for simplicity.

- Provides a clear and concise summary of the user's performance.

- **Percentage Calculation:**

- The `showPercentage` method calculates the percentage of correct answers.

- Ensures accurate representation of the user's performance.

**4. Main Class:**

**- Sample Questions:**

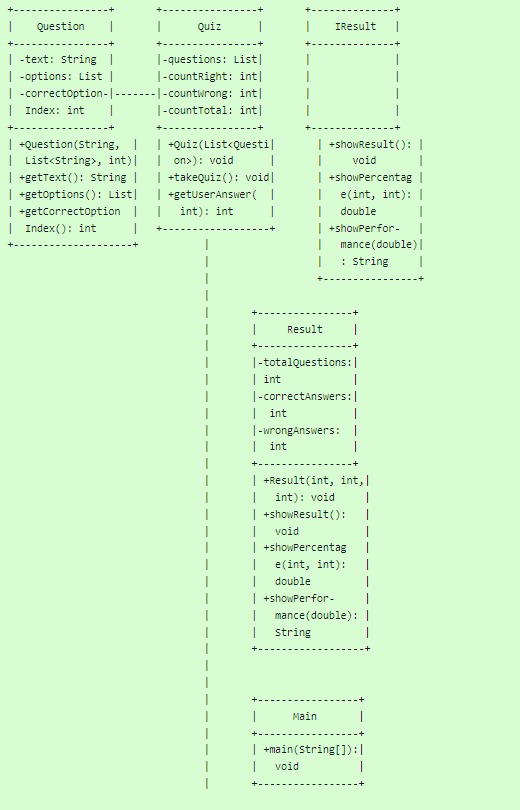
- Demonstrates the usage of the quiz with a set of sample questions.

- Questions cover a range of topics, promoting variety and testing different knowledge areas.

- **Percentage Thresholds:**

- The threshold for categorizing performance is set based on common grading standards.

**Class Diagram :**



**Challenges:**

**1. User Input Validation:**

- Validating user input (e.g., ensuring the user enters a number within the given range) can be challenging. The code addresses this with a do-while loop and exception handling.

**2. Hardcoding Thresholds:**

- Hardcoding performance thresholds may not be suitable for all scenarios. A more flexible solution could involve external configuration or user-defined thresholds.

**3. Scalability:**

- While the current design handles a moderate number of questions, scalability considerations might be necessary for larger quizzes.

**4. Code Reusability:**

- The code promotes reusability by encapsulating functionality within classes. However, further refactoring could enhance modularity and make components even more reusable.

**Conclusion:**

The provided code establishes a foundation for a console-based quiz application. It balances simplicity with key features and demonstrates object-oriented design principles. Enhancements could include greater configurability, improved user feedback, and increased modularity for scalability.