Java Week 1 Task: Quiz Generator - Documentation

Overview

The task is to build a command-line quiz generator application. This application allows users to create quizzes, add multiple-choice questions with correct answers, and take quizzes to test their knowledge. The system should store the questions, options, and answers using suitable data structures, provide a user-friendly interface for taking quizzes, and display the results with scores and feedback.

Features

Create a Quiz:

Allow users to create quizzes by adding questions.

Each question has multiple-choice options, and the user can specify the correct answer.

Take a Quiz:

- Users can take quizzes to test their knowledge.
- The system will display one question at a time and present multiple-choice options.
- Users select an option, and at the end of the quiz, they get feedback on their performance.
  Store Questions and Answers:
- Use appropriate data structures (like lists or maps) to store questions, options, and correct answers.

Display Results:

After the quiz, display the total score and feedback based on the user's performance.
Data Structures

Question Class:

- Stores individual quiz questions.
- Contains properties like the question text, multiple options, and the correct answer.

```
class Question {

  String questionText;

  String[] options;  // Array of options (4 choices per question)

  String correctAnswer;  // Correct answer for the question

}
```

Quiz Class:

- Contains a list of Question objects.
- Manages the addition of questions to the quiz and facilitates the quiz-taking process.

```
class Quiz {
   List<Question> questions;
   // Method to add question to the quiz
   void addQuestion(Question question) {
      questions.add(question);
   }
}
```

Result Class:

- Stores the score and feedback after the quiz.
- Tracks the total number of correct answers.

```
class Result {
   int score;  // Number of correct answers
   String feedback;
}
```

Commands to Use

Create a Quiz:

Command: createQuiz

Description: Starts the process of creating a new quiz.

```
public void createQuiz() {
   // Prompt user to enter questions and options
   // Store the question and correct answer
}
```

Add Question to Quiz:

Command: addQuestion

Description: Adds a new question to the current quiz.

```
public void addQuestion(String questionText, String[] options, String correctAnswer) {
   // Create a new Question object and add it to the quiz
}
```

Take a Quiz:

Command: takeQuiz

Description: Starts the quiz-taking process where questions are shown one by one.

```
public void takeQuiz() {
   // Loop through each question and display options
   // Get user input for answers
}
```

Submit Answers:

Command: submitAnswers

Description: Submits the answers after taking the quiz and calculates the score.

```
public void submitAnswers(String[] answers) {
   // Compare answers and calculate score
   // Show feedback based on performance
}
```

Display Results:

Command: showResults

Description: Displays the total score and feedback after the quiz is completed.

```java
public void showResults(int score) {
   // Display score and provide feedback based on performance
}
```

<u>Example Usage</u>

Create a Quiz and Add Questions:

User: createQuiz

User: addQuestion("What is the capital of France?", ["Paris", "London", "Berlin", "Madrid"], "Paris")

User: addQuestion("What is 2 + 2?", ["3", "4", "5", "6"], "4")

<u>Take the Quiz:</u>

User: takeQuiz

System: "Question 1: What is the capital of France?"

System: "1. Paris 2. London 3. Berlin 4. Madrid"

User: 1

Submit and Show Results:

User: submitAnswers([1, 2])

System: "Your score is: 2/2. Great job!"

Feedback and Score Calculation

Provide feedback based on the number of correct answers:

<u>Example:</u>

0-2 correct answers: "Better luck next time!"

3-4 correct answers: "Good job!"

5 correct answers: "Excellent work!"

In this application I used Eclipse IDE.

Step 1: Install and Set Up Eclipse IDE

1. **Download Eclipse IDE** :

    - Go to Eclipse Downloads and download the appropriate version for your operating system.
    - Follow the installation instructions.

2. **Install Java :**

- Download and install JDK (Java Development Kit).
- Make sure to set up the **JAVA_HOME** environment variable on your system.

3. **Open Eclipse**:

- Once installed, open Eclipse IDE.
- It will ask you to select a workspace (this is where your project files will be saved).

Step 2: Create a New Java Project

1. **Open Eclipse IDE**.
2. **Create a new Java project**:

    - Go to `File` > `New` > `Java Project`.
    - Give your project a name (e.g., `QuizGenerator`).
    - Click `Finish`.

Step 3: Create a New Java Class

1. **Create a new class**:

    - Right-click on the `src` folder in the Project Explorer.
    - Select `New` > `Class`.
    - Name the class `QuizGenerator`.
    - Check the option `public static void main(String[] args)` to generate the main method.
    - Click `Finish`.

Step 4: Build the Structure of the Quiz Application

1. Define the Data Structure for Quiz

We need to store questions, options, and correct answers. This can be done using a class for each question.

- **Create a Question class**: This will hold data about each quiz question (question text, options, and correct answer).

Step 5: Documentation and Improvement

1. **Add documentation**: Document the functionality of your program, what each class does, and how to use the commands.
2. **Enhance the project**:

   - You can add more functionality, such as saving the quiz questions in a file, allowing users to create their own quizzes, or improving the user interface.

Step 6: Submit

- After completing your code and documentation, make sure to submit it as required by your task.

By following these steps, you should be able to create a basic command-line quiz generator application using Java in Eclipse.

Important Notes:

- **Packages** are special types of folders in Java, and they help organize classes. In Eclipse, creating a **package** is similar to creating a folder but with a specific structure.
- If you want to store **non-code files** (like `.txt`, `.xml`, or `.properties` files), you can create regular folders within the project or outside the `src` folder.
- You can also create **bin** and **lib** folders to manage compiled code and libraries in your project.

Conclusion

This application helps users create and take quizzes in a simple and interactive manner. The design utilizes basic data structures to manage quiz questions, answers, and results. The provided commands and their functionality allow users to manage quizzes and assess their knowledge effectively.