**Project: Quiz System with OOP Concepts**

**Objective:** Implement a simple quiz system using Object-Oriented Programming (OOP) principles. This project will involve creating classes for quizzes, questions, and quiz takers. Students will implement actions like taking quizzes, answering questions, and receiving scores for correct answers.

**Classes:**

1. **Question Class:**
   * Create a **Question** class with the following attributes:
     + **questionText** (String): Text of the question.
     + **options** (List<String>): List of options for the question.
     + **correctOptionIndex** (int): Index of the correct option in the **options** list.
   * Implement methods:
     + **displayQuestion()**: Display the question and options.
     + **isCorrectAnswer(int answerIndex)**: Check if the given answer index is correct.
2. **Quiz Class:**
   * Create a **Quiz** class with the following attributes:
     + **questions** (List<Question>): List of questions in the quiz.
   * Implement methods:
     + **displayQuiz()**: Display all questions in the quiz.
     + **scoreQuiz(List<Integer> playerAnswers)**: Score the quiz based on player answers.
3. **QuizTaker Class:**
   * Create a **QuizTaker** class with the following attributes:
     + **name** (String): Name of the quiz taker.
   * Implement methods:
     + **takeQuiz(Quiz quiz)**: Take a quiz and record the score.
     + **getQuizScores()**: Retrieve the map of quiz scores.
4. **Main Class (QuizSystem):**
   * Create a main class to interact with the quiz system using a **Scanner**.
   * Allow users to:
     + Create quizzes with hardcoded questions.
     + Take quizzes, answer questions, and receive scores.
     + Display quiz results for all players.

**Instructions:**

1. **Question Class:**
   * Create a **Question** class with the specified attributes and methods.
   * Ensure that the **displayQuestion** method presents the question and options in a user-friendly format.
2. **Quiz Class:**
   * Implement a **Quiz** class with the specified attributes and methods.
   * The **displayQuiz** method should iterate through the list of questions and call the **displayQuestion** method for each question.
3. **QuizTaker Class:**
   * Create a **QuizTaker** class with the specified attributes and methods.
   * Implement the **takeQuiz** method to interact with the user using the **Scanner**. Allow the user to answer each question.

**Display Instructions:**

1. Print a welcome message to greet the quiz taker.
2. Inform the quiz taker that they will be presented with 10 random questions.

**Initialize Variables:**

1. Create a **List<Integer>** variable named **playerAnswers** to store the quiz taker's answers.
2. Create a **Scanner** instance for reading the quiz taker's input.

**Take Quiz:**

1. Iterate through each question in the quiz using a loop.
2. Display each question using the **display** method
3. Prompt the quiz taker to enter their answer using the scanner.
4. Add the quiz taker's answer to the **playerAnswers** list.

**Score Quiz:**

1. Calculate the quiz taker's score by calling the **scoreQuiz** method of the quiz instance.
2. Display the quiz taker's score.

**Record Score:**

1. Record the quiz taker's score in the **quizScores** map using the taker name as the key
2. **Main Class (QuizSystem):**
   * Create a global map with scores: key is the player name and value is the score.
   * Implement a **populateQuestions** method to create a list of hardcoded questions with answers.
   * Create an instance of the **Quiz** class and populate it with questions using the **populateQuestions** method.
   * In the main menu, allow users to:
     + Start a quiz by entering their name.
     + Display quiz results for all players.
     + Exit the quiz system.

**Initialize Variables:**

* + Create a **List<Question>** variable to store all available questions.
  + Call the **populateQuestions** method to populate the list with hardcoded questions.
  + **Create Quiz:**
    - Create an instance of the **Quiz** class using the list of questions.
    - Use a helper method, **generateRandomQuiz**, to obtain a subset of 10 random questions from the list of all questions.

**Generate Random Quiz Method:**

1. Implement a method named **generateRandomQuiz** to randomly select 10 questions from the list of all questions.

**Main Menu Loop:**

* + Implement a loop to display the main menu and handle user interactions until the user chooses to exit.
    - System.out.println("1. Start Quiz");
    - System.out.println("2. Display Results");
    - System.out.println("3. Exit");
    - System.out.print("Enter your choice: ");

**Display Results Method:**

* + Implement a method named **displayResults** to display quiz results for all players.

1. **Extensions (Optional):**
   * Add interfaces and try in main to work only with interfaces to get the abstraction.
   * Add exception handling for each wrong input or not correct format of answers.
   * Add a timer for quizzes.
   * Implement different types of questions (e.g., multiple-choice, true/false).
   * Allow the quiz system to save and load data to/from a file for persistence.
   * Implement a leaderboard to display top quiz scores.

Example of population the questions:

* **private static** List<Question> populateQuestions() {  
   List<Question> questions = **new** ArrayList<>();  
   *// Question 1* List<String> options1 = Arrays.*asList*(**"0.Berlin"**, **"1.Paris"**, **"2.Madrid"**);  
   questions.add(**new** Question(**"What is the capital of France?"**, options1, **1**));  
    
   *// Question 2* List<String> options2 = Arrays.*asList*(**"0.Mars"**, **"1.Jupiter"**, **"2.Saturn"**);  
   questions.add(**new** Question(**"Which planet is known as the Red Planet?"**, options2, **1**));  
    
   *// Question 3* List<String> options3 = Arrays.*asList*(**"0.New York"**, **"1.Los Angeles"**, **"2.Chicago"**);  
   questions.add(**new** Question(**"What is the most populous city in the United States?"**, options3, **1**));….