Project Title: Online Quiz/Exam System with Student Participation and Leaderboard

Overview:

The project is designed to simulate a quiz system where various types of questions are defined and managed dynamically. Students (represented as objects) take the quiz using pre-defined (simulated) answers, and their scores are recorded on a leaderboard. The system demonstrates core object-oriented programming (OOP) concepts such as encapsulation, inheritance, method overriding, and dynamic collections.

Class Descriptions:

1. Question (Base Class)

- **Purpose:**
  Encapsulates a single quiz question with its text, answer options, and the correct answer.
- **Attributes:**
  - **questionText**
    - **Type:**
      - Java: `String`
      - Python: `str`
    - **Description:** Holds the text of the question.
  - **options**
    - **Type:**
      - Java: `ArrayList<String>`
      - Python: `list`
    - **Description:** Contains a list of possible answer choices. For certain question types, this list may be empty.
  - **correctAnswer**
    - **Type:**
      - Java: `String`
      - Python: `str`
    - **Description:** The correct answer for the question.
- **Constructor:**

- **Signature:**
  - Java: `public Question(String questionText, ArrayList<String> options, String correctAnswer)`
  - Python: `def __init__(self, question_text: str, options: list, correct_answer: str) -> None`
- **Parameters:**
  - `questionText` (or `question_text`): Text of the question.
  - `options`: List of answer choices.
  - `correctAnswer` (or `correct_answer`): The correct answer.

- **Methods:**
  - **getQuestionText()** / **get_question_text()**
    - **Return Type:** `String` (Java) / `str` (Python)
    - **Description:** Returns the question text.
  - **getOptions()** / **get_options()**
    - **Return Type:** `ArrayList<String>` (Java) / `list` (Python)
    - **Description:** Returns the list of answer options.
  - **getCorrectAnswer()** / **get_correct_answer()**
    - **Return Type:** `String` (Java) / `str` (Python)
    - **Description:** Returns the correct answer.
  - **setQuestionText(String questionText)** / **set_question_text(question_text: str) -> None**
    - **Parameters:** New question text.
    - **Return Type:** `void` (Java) / `None` (Python)
    - **Description:** Updates the question text.
  - **setOptions(ArrayList options)** / **set_options(options: list) -> None**
    - **Parameters:** New list of answer options.
    - **Return Type:** `void` / `None`
    - **Description:** Updates the answer options.
  - **setCorrectAnswer(String correctAnswer)** / **set_correct_answer(correct_answer: str) -> None**
    - **Parameters:** New correct answer.
    - **Return Type:** `void` / `None`
    - **Description:** Updates the correct answer.
  - **validateAnswer(String answer)** / **validate_answer(answer: str) -> bool**

        - **Parameters:**
            - `answer` : The answer to be checked.
        - **Return Type:** `boolean` (Java) / `bool` (Python)
        - **Description:** Compares the provided answer with the stored correct answer and returns `true` if they match.

2. Derived Classes from Question:

a. MultipleChoiceQuestion

- **Inheritance:**
  Extends the `Question` class.
- **Purpose:**
  Handles questions with multiple answer choices. It overrides the `validateAnswer` method to perform a case-insensitive comparison.
- **Constructor:**
    - **Signature:**
        - Java: `public MultipleChoiceQuestion(String questionText, ArrayList<String> options, String correctAnswer)`
        - Python: `def __init__(self, question_text: str, options: list, correct_answer: str) -> None`
- **Method Override:**
    - **validateAnswer(String answer) / validate_answer(answer: str) -> bool**
        - **Description:** Checks the answer after trimming and converting both the input and the correct answer to lower case.

b. TrueFalseQuestion

- **Inheritance:**
  Extends the `Question` class.
- **Purpose:**
  Represents questions with only two possible answers ("True" or "False"). The constructor automatically sets the options to `["True", "False"]` .
- **Constructor:**
    - **Signature:**
        - Java: `public TrueFalseQuestion(String questionText, String correctAnswer)`
        - Python: `def __init__(self, question_text: str, correct_answer: str) -> None`
- **Method Override:**

- **validateAnswer(String answer) / validate_answer(answer: str) -> bool**
  - **Description:** Performs a case-insensitive check against the preset correct answer.

c. FillInTheBlankQuestion

- **Inheritance:**
  Extends the `Question` class.
- **Purpose:**
  For questions where the answer is entered freely (no options required).
- **Constructor:**
  - **Signature:**
    - Java: `public FillInTheBlankQuestion(String questionText, String correctAnswer)`
    - Python: `def __init__(self, question_text: str, correct_answer: str) -> None`
- **Method Override:**
  - **validateAnswer(String answer) / validate_answer(answer: str) -> bool**
    - **Description:** Compares the input answer (after trimming and case normalization) with the correct answer.

3. Quiz

- **Purpose:**
  Manages a dynamic collection of `Question` objects.
- **Attributes:**
  - **questions**
    - **Type:**
      - Java: `ArrayList<Question>`
      - Python: `list`
    - **Description:** Stores all quiz questions.
- **Constructor:**
  - **Signature:**
    - Java: `public Quiz()`
    - Python: `def __init__(self) -> None`
- **Methods:**
  - **addQuestion(Question question) / add_question(question: Question) -> None**
    - **Parameters:** A `Question` object.

        - **Return Type:** `void` / `None`
        - **Description:** Adds a new question to the quiz.
    - **removeQuestion(Question question)** / **remove_question(question: Question) -> None**
        - **Description:** Removes a specified question.
    - **getQuestions()** / **get_questions() -> ArrayList (Java)** / **list (Python)**
        - **Description:** Returns the list of questions.
    - **shuffleQuestions()** / **shuffle_questions() -> None**
        - **Description:** Randomizes the order of the questions.
    - **getTotalQuestions()** / **get_total_questions() -> int**
        - **Description:** Returns the total number of questions.

4. Person

- **Purpose:**
  Serves as a base class for individuals involved in the system.
- **Attributes:**
    - **name**
        - **Type:**
            - Java: `String`
            - Python: `str`
    - **age**
        - **Type:**
            - Java: `int`
            - Python: `int`
- **Constructor:**
    - **Signature:**
        - Java: `public Person(String name, int age)`
        - Python: `def __init__(self, name: str, age: int) -> None`
- **Methods:**
    - **getName()** / **get_name() -> String/str**
    - **getAge()** / **get_age() -> int**

5. Student

- **Inheritance:**

Extends the `Person` class.

- **Purpose:**
  Represents a student taking the quiz.
- **Attributes:**
  - **studentId**
    - **Type:**
      - Java: `String`
      - Python: `str`
  - **score**
    - **Type:**
      - Java: `int`
      - Python: `int`
- **Constructor:**
  - **Signature:**
    - Java: `public Student(String name, int age, String studentId)`
    - Python: `def __init__(self, name: str, age: int, student_id: str) -> None`
- **Methods:**
  - **simulateQuiz(Quiz quiz, ArrayList simulatedAnswers) / simulate_quiz(quiz: Quiz, simulated_answers: list) -> None**
    - **Parameters:**
      - `quiz` : The `Quiz` object containing the questions.
      - `simulatedAnswers` (Java) / `simulated_answers` (Python): A list of answers corresponding to each question.
    - **Return Type:** `void` / `None`
    - **Description:** Iterates through each question in the quiz, compares the simulated answer using `validateAnswer` , updates the score, and prints the result.
  - **getScore() / get_score() -> int**
    - **Description:** Returns the student's score.

6. Leaderboard

- **Purpose:**
  Maintains a record of students and their quiz scores.
- **Attributes:**
  - **students**
    - **Type:**

        - Java: `ArrayList<Student>`
        - Python: `list`
    - **Description:** Holds student objects with their respective scores.
- **Constructor:**
    - **Signature:**
        - Java: `public Leaderboard()`
        - Python: `def __init__(self) -> None`
- **Methods:**
    - **addStudent(Student student) / add_student(student: Student) -> None**
        - **Parameters:** A `Student` object.
        - **Return Type:** `void` / `None`
        - **Description:** Adds a student to the leaderboard.
    - **displayLeaderboard() / display_leaderboard() -> None**
        - **Description:** Prints the name and score of each student. (No sorting is performed.)

Main Method / Main Function:

The main entry point of the program should:

1. **Test the Question Classes:**
    - Create instances of `Question`, `MultipleChoiceQuestion`, `TrueFalseQuestion`, and `FillInTheBlankQuestion` with hardcoded data.
    - Use the getter methods to print question details.
    - Validate answers using simulated correct responses.
2. **Build and Test the Quiz:**
    - Instantiate a `Quiz` object.
    - Add all created questions to the quiz.
    - Optionally, simulate a quiz run by iterating through each question, displaying the question text and options, and using the correct answer as the simulated input.
    - Test removal and shuffling of questions.
3. **Simulate Student Quiz Participation and Leaderboard:**
    - Create a `Leaderboard` object.
    - Hardcode data for one or more `Student` objects.

    - For each student, supply a list of simulated answers corresponding to the quiz questions.
    - Call the `simulateQuiz` (Java) or `simulate_quiz` (Python) method on each student to simulate quiz participation and update their score.
    - Add each student to the leaderboard.
    - Finally, display the leaderboard showing each student's name and score.

Usage for Students:

When students review this project description, they should:

- Understand how to define classes with private attributes and public methods.
- Know how inheritance allows the creation of specialized question types.
- Learn how to use dynamic collections to manage quiz questions.
- Implement methods with correct parameters and return types.
- Write a main method (or function) that creates objects, simulates quiz participation, and displays the leaderboard.

This detailed description should serve as a clear guideline for students to write the complete code in both Java and Python, ensuring that all necessary components are implemented and can be rigorously tested using the main method.

Happy coding!