Overview

The project is a gradebook system designed to manage student grades for a course. Its purpose is to demonstrate core OOP principles—encapsulation, inheritance, and composition—while working with dynamic data structures (like ArrayLists or custom ListADTs). The system reads student data (including a series of grade entries) from a file, allows modifications to individual grades or the entire roster, and calculates current scores as well as needed averages for target grades. Look for the To-Do in the Solution.java and fill those methods based on the projec t gradebook.

Class Design and Relationships

1. **Person Class**
    - **Purpose:**
      Serves as the base class containing common personal information.
    - **Attributes:**
        - `firstName`
        - `lastName`
        - `nationality`
        - `age`
    - **Design Principle:**
      Encapsulation is used to hide details about a person while providing public methods (like `toString()`) to access that information.
2. **Student Class (extends Person)**
    - **Purpose:**
      Represents a student enrolled in the course and adds course-specific information.
    - **Additional Attributes:**
        - `major` (the student's field of study)
        - `andrewID` (unique identifier)
        - A private list (or a custom ListADT) of `Grade` objects, representing the student's assignment grades.
    - **Methods:**
        - `addGrade(Grade g)` : Adds a new grade entry.
        - `changeGrade(String assignment, int newScore)` : Modifies a specific grade entry.
        - `getCurrentScore()` : Computes the weighted average of all grades.
        - `getCurrentLetterGrade()` : Determines the letter grade based on the computed

score.
        - `whatDoINeed(char targetGrade)` : Calculates the required average on remaining assignments to achieve a target letter grade.
    - **Design Principle:**
      Inheritance is used to extend the Person class, and *composition* is used by holding a collection of Grade objects.

3. **Grade Class**

    - **Purpose:**
      Encapsulates details about a single grade entry.
    - **Attributes:**
        - `assignmentName` (e.g., "HW1", "Q1", "MT", "F")
        - `score` (an integer value representing the grade)
        - `weight` (a double value computed based on the assignment type; fixed weights such as 0.07 for homework, 0.015 for quizzes, etc.)
    - **Methods:**
        - Getter methods for each attribute.
        - `setScore(int newScore)` : Updates the score.
    - **Design Principle:**
      Encapsulation and *abstraction* allow the Grade class to manage its own state and behavior related to weight calculation.

4. **GradeBook Class**

    - **Purpose:**
      Manages a collection of Student objects and provides methods for operations on the overall course data.
    - **Attributes:**
        - A dynamic list (or a custom ListADT) of `Student` objects.
    - **Key Methods:**
        - **Constructor:**
            - Reads a file (e.g., `S18grades.txt`) and creates Student objects along with their grade entries.
        - **toString():**
            - Returns a formatted string representation of the entire roster.
        - **printIndividualGrades(String id):**
            - Prints all grade entries for a given student identified by their Andrew ID.
        - **printGradesByMajor(String major):**

            - Prints the information and grades for all students in a specified major.
        - **removeStudent(String id):**
            - Removes a student from the list based on their Andrew ID.
        - **changeGrade(String id, String assignment, int newgrade):**
            - Finds the student by ID and updates the grade for a specific assignment.
        - **addGradeToAll():**
            - Prompts the user for a new assignment and then adds this assignment (with a user-specified grade) to every student.
        - **printCurrentGrade(String id):**
            - Computes and prints the current letter grade of a student based on their weighted average.
        - **whatDoINeed(String id, char grade):**
            - Determines what score a student needs on the remaining coursework to reach a desired letter grade.
        - **updateDatabase(String filename):**
            - Writes the current state of the gradebook back to a file so it can be reloaded later.
    - **Design Principle:**
      The GradeBook class demonstrates the *Single Responsibility Principle* by managing the collection of students and orchestrating operations on them, while delegating student-specific behavior to the Student class.

Testing and Main Method

- **Main Method:**
  The project includes a `main()` method that serves as a test harness. This method is responsible for:
    - Instantiating the GradeBook (by reading the provided data file).
    - Calling methods on the GradeBook to test each feature.
    - Using conditional logic (`if` / `else`) to verify that each operation works as expected.
    - Running a suite of around 30 test cases that cover:
        - File reading and GradeBook creation.
        - Individual and bulk operations on student grades.
        - Error handling for non-existent student IDs or assignments.
        - Grade calculations and database updates.