Semester Project

(CLO-5: Develop a GUI based project for a real-world problem in a team environment)

Project Overview:

The **Student Result Management System** is a Java-based application that manages the academic performance of students in multiple programs (e.g., Science, Arts, Engineering). The system persists data using binary .dat files (serialization) and provides a user-friendly Swing GUI for managing students, courses, and grades.

This system allows:

- Adding students and their transcripts
- Assigning courses and instructors
- Recording results for each course
- Calculating GPA, total marks, and grades
- Viewing reports per student or course
- Tracking global metrics (total students, total courses, pass marks)

Classes and Their Relationships

1. Student (Abstract Class)

- Attributes: studentId, name, program
- Composition: **has-a Transcript**
- Static: static int totalStudents
- Methods: addCourse(), calculateGPA(), displayResults()
- Subclasses: ScienceStudent, ArtsStudent, EngineeringStudent

2. Transcript

- Composition: **has-many ResultEntry objects**
- Attributes: List<ResultEntry> results
- Methods: addResultEntry(ResultEntry r), getGPA(), getTotalMarks()

3. ResultEntry

- Attributes: Course course, double marksObtained
- Exists only as part of a **Transcript** (composition)

4. Course

- Attributes: courseCode, title, creditHours
- Composition: **has-a CourseInstructor**
- Static: static int totalCourses
- Methods: displayCourseDetails()

5. CourseInstructor

- Attributes: name, qualification
- Exists only as part of a **Course**

6. RecordList<T> (Generic Class)

- Attributes: List<T> items
- Methods: add(T item), remove(String id), getAll()
- Usage: stores Students, Courses, or Transcripts

7. DataStore<T> (Generic Class)

- Methods: saveToFile(String fileName, List<T> items), loadFromFile(String fileName)
- Handles saving/loading objects via .dat files

8. ResultCalculator (Interface)

- Static Member: static final double passMarks = 50;
- Methods: calculateTotal(), calculatePercentage(), calculateGrade()
- Implemented by: all Student subclasses

Hint:

1. **Composition**
   - Each **Student** *has-a* **Transcript** (cannot exist independently).
   - Each **Transcript** *has-many* **ResultEntry** objects (results of individual courses).
   - Each **Course** *has-a* **CourseInstructor** (cannot exist independently of the course).
2. **Generics**
   - **RecordList<T>** stores heterogeneous objects such as Students, Courses, and Transcripts in type-safe lists.
   - **DataStore<T>** provides generic methods for saving and loading any type of records from .dat files.
3. **Static Members**
   - Student.totalStudents tracks the total number of students created.
   - Course.totalCourses tracks the total number of courses created.
   - ResultCalculator.passMarks is a static constant used across all result calculations.
4. **Interfaces**
   - **ResultCalculator** defines methods calculateTotal(), calculatePercentage(), calculateGrade().
   - All Student subclasses implement this interface to ensure polymorphic result calculation.

Deliverables:

Code, UML Diagram, Documentation

<u>How to present (example):</u>

1. Create a new **ScienceStudent** and increment Student.totalStudents.
2. Assign a **Transcript** to the student.
3. Add **ResultEntry** objects for courses such as Physics, Chemistry, and Mathematics.
4. Each **Course** has a **CourseInstructor**.
5. Calculate total marks, percentage, GPA, and grade using **ResultCalculator** interface methods.
6. Save all records to a .dat file using **DataStore<Student>**.
7. Load records from the .dat file when the application restarts.
8. Display the student's results in the GUI table.

PS: Add some records in each table when appear for presentation