Daily Student Attendance Tracker

**1. Introduction**

The "Student Attendance Tracker" is a pure Java application prepared to manage and display student attendance and lecture information. The system has functionalities for three user roles: Student, Lecturer, and Admin. This report outlines the system’s design, key functionalities, and implementation details.

**2. System Overview**

**2.1 Purpose**

The primary goal of the Student Attendance Tracker is to streamline the process of tracking attendance and grades while enhancing administrative control over the system. This project ensures data integrity and simplifies user interactions for students, lecturers, and administrators.

**2.2 Features**

* Students can view their attendance, grades, and lecture schedules.
* Lecturers can add or modify lectures, update attendance records, and manage grades.
* Admin

**3. Diagrams**

**3.1 Use Case Diagram**

The Use Case Diagram depicts the interactions between the user roles and system functionalities:

* **Student**: View attendance and view lectures.
* **Lecturer**: Modify attendance, modify grades, add student, withdraw student from lectures.
* **Admin**: Add lecturer, modify lecturer.

**3.2 Class Diagram**

The Class Diagram illustrates the structure and relationships between the system’s core components:

* **Classes**:
  + Student: Attributes include id, name, email, and enrolled courses. Methods allow viewing attendance and lectures.
  + Lecturer: Attributes include id, name, email, and assigned courses. Methods facilitate managing attendance and lectures.
  + Admin: Attributes include id and name. Methods enable adding and modifying lecturer details.
  + Course: Attributes include id, name, and lecturer. Methods handle enrolled students.

Overview

The provided code simulates a basic **Attendance Management System**. It uses object-oriented programming principles like inheritance, encapsulation, polymorphism, and interfaces in Java. The system allows users of different roles (**Admin**, **Lecturer**, **Student**) to log in, view, and manage attendance-related data.

The project is modularized into several classes, each with a distinct purpose. The main roles and functionality include:

* **Admin**: Manages users and their data.
* **Lecturer**: Manages courses and marks attendance.
* **Student**: Views their own attendance and enrolled courses.
* **DataStore**: Generic data handling for storing records.
* **Main**: Entry point for the application.

**2. Class-by-Class Explanation**

**Admin.java**

* **Purpose**: Represents an admin user with extra privileges.
* **Inheritance**: Extends User and overrides the viewDetails method to display admin-specific details.
* **Unique Methods**:
  + addUser(User user): Adding a user.
  + modifyUser(User user): Modifying user details.

**AttendanceRecord.java**

* **Purpose**: Tracks attendance information for a specific **student** in a **course**.
* **Attributes**:
  + course: The course associated with this attendance record.
  + student: The student this record belongs to.
  + isPresent: Boolean indicating whether the student was present or absent.
* **Methods**:
  + Standard getters and setters for encapsulation.

**Course.java**

* **Purpose**: Represents a university course.
* **Attributes**:
  + id, name: Basic course identifiers.
  + lecturer: Lecturer teaching the course.
  + enrolledStudents: A Set storing students enrolled in the course.
* **Methods**:
  + Standard getters and setters.
  + viewEnrolledStudents(): Prints a list of students enrolled in the course using a lambda function.

**DataStore.java**

* **Purpose**: A generic class to store any type of records.
* **Attributes**:
  + records: A List storing generic type T.
* **Methods**:
  + addRecord(T record): Adds a record to the list.
  + getRecords(): Retrieves all records.
  + removeRecord(T record): Removes a specific record from the list.

**Lecturer.java**

* **Purpose**: Represents a lecturer, inheriting from User.
* **Attributes**:
  + courses: A list of courses taught by the lecturer.
* **Methods**:
  + modifyAttendance(): Placeholder to modify attendance.
  + modifyGrades(): Placeholder to modify grades.
  + viewDetails(): Overrides to display lecturer-specific details.
  + addCourse(Course course): Adds a course to the lecturer's list.
  + viewCourses(): Placeholder to view all courses.

**Student.java**

* **Purpose**: Represents a student in the system.
* **Attributes**:
  + email: Email address of the student.
  + enrolledCourses: List of courses the student is enrolled in.
  + attendanceStatus: A map tracking attendance for each course.
* **Methods**:
  + enrollInCourse(Course course): Adds a course to the student's list.
  + updateAttendanceStatus(Course course, boolean isPresent): Updates attendance for a specific course.
  + viewDetails(): Displays student-specific details, including courses and attendance.

**User.java**

* **Purpose**: Abstract base class for all types of users.
* **Attributes**:
  + id, name, email: Basic user details.
* **Methods**:
  + Abstract viewDetails(): Forces subclasses to define their specific details view.
  + Default methods for unsupported operations (e.g., viewLectures(), markAttendance()).

**UserActions.java**

* **Purpose**: Interface defining actions a user can perform.
* **Methods**:
  + Abstract viewDetails(): Must be implemented by all subclasses.
  + Default methods for common user actions, with unsupported operations throwing exceptions.

**Main.java**

* **Purpose**: Entry point for the application. Manages user authentication, navigation, and role-specific menus.
* **Key Functionalities**:
  + **Login System**: Validates user credentials and redirects them to role-specific menus.
  + **Menus**:
    - **Student Menu**: Allows viewing details (attendance and courses) or logging out.
    - **Lecturer Menu**: Allows marking attendance and viewing details.
    - **Admin Menu**: Limited functionality to view details or logout.
  + **Attendance Marking**:
    - Lecturers can mark attendance for their courses.
    - Attendance is stored in a DataStore<AttendanceRecord>.

**3. Key Concepts Used**

**a. Object-Oriented Programming**

* **Inheritance**: Admin, Lecturer, and Student inherit from User.
* **Polymorphism**: The viewDetails() method is overridden in each subclass.
* **Encapsulation**: Attributes are private with public getters and setters.

**b. Generics**

* DataStore<T> is a generic class used for storing attendance records.

**c. Functional Programming**

* Lambda functions (e.g., courses.forEach(course -> ...)).

**d. Collections**

* **Set**: Used in Course to store enrolled students.
* **Map**: Used in Main for credential storage and in Student for attendance tracking.
* **List**: Used in multiple classes for storing records, courses, and users.

**5. Conclusion**

The code demonstrates a well-structured use of Java programming principles to implement a role-based system for managing attendance. While functional, it can be enhanced with better error handling, database integration, and a more user-friendly interface for production-level use.

Source

<https://stackoverflow.com/questions/55200665/application-that-manages-the-students-in-a-faculty>

<https://stackoverflow.com/questions/34781845/how-to-make-sure-that-the-user-is-not-able-to-take-the-attendance-of-the-same-cl>

<https://medium.com/@smagid_allThings/uml-class-diagrams-tutorial-step-by-step-520fd83b300b>

<https://www.geeksforgeeks.org/polymorphism-in-java/>