Design and Development of an E-Learning Platform

**Case Study 3**

**Kian Chryslyr Q. Cadungog**

**Ruby Jean B. Solomon**

**Ralph Jade Omega**

Advised by:

**Prof. Joseph C. Lorilla**

**A yellow and green logo

Description automatically generated**

DECEMBER 2024

Description

The code represents a comprehensive **E-Learning Environment** system designed to manage students, instructors, courses, enrollments, assignments, grades, discussions, announcements, schedules, and admin operations. The system includes object-oriented programming principles, using multiple classes and methods to define the relationships and operations between different entities in an educational platform. Below is a detailed explanation of the system and its components:

**Overview:**

The system serves as a platform to:

* **Administrate educational resources** via instructors, students, and courses.
* Facilitate **student management** (e.g., enrollment, grade tracking).
* Allow instructors to **assign assignments**, manage courses, and maintain schedules.
* Support discussions between participants through **discussion threads**.
* Enable **administrative functionalities** for platform-level management, including user authentication and data persistence.

**Main Components:**

**1. Person Class (Abstract Base Class):**

* Acts as the base class for managing general information about a person associated with the system (students and instructors).
* Includes attributes like name, email, contact\_number, and address with property methods and validation.
* Contains an abstract method display\_info() which must be implemented by subclasses.
* Enforces data integrity by validating input (e.g., email format, contact number length).

**2. Student Class (Subclass of Person):**

* Represents students in the system.
* Attributes include student\_id, year\_level, program, GPA, enrolled\_courses, and grades.
* Provides methods to:
  + add\_grade: Add grades for courses.
  + enroll: Enroll in courses.
  + Data conversion (to\_dict, from\_dict).
  + Display student-specific information (display\_info).

**3. Instructor Class (Subclass of Person):**

* Represents instructors in the system.
* Attributes include instructor\_id, courses\_taught, and assignments.
* Methods to:
  + teach\_course: Associate instructors with courses.
  + assign\_assignment: Create and assign assignments to courses.
  + Convert data to a dictionary (to\_dict).
  + Display detailed instructor information (display\_info).

**4. Course Class:**

* Represents academic courses.
* Attributes include course\_name, course\_code, instructor (who teaches it), units, students, assignments, and grades.
* Provides methods to:
  + Enroll students (add\_student).
  + Assign grades and assignments.
  + Manage discussion threads (discussion\_threads).
  + Display course details (display\_info).
  + Convert between dictionary representation (to\_dict).

**5. Enrollment Class:**

* Handles course enrollment operations for students.
* Tracks associations between Student and Course objects with methods to:
  + Enroll and unenroll students.
  + Check enrollment status.
  + Retrieve lists of students and courses based on their enrollment.
  + Display current enrollments.

**6. Grade Class:**

* Represents the grades assigned to students for specific assignments in a course.
* Attributes for score, feedback, and associations with Student and Assignment.
* Methods for:
  + Score validation (must be between 0-100).
  + Conversion between dictionary representation and object (to\_dict, from\_dict).
  + Display of grade-related information.

**7. Schedule Class:**

* Associates courses with specific days and times.
* Used to manage weekly scheduling information.
* Provides methods to:
  + Display schedules for a given course.
  + Convert schedules to/from dictionary format.

**8. Assignment Class:**

* Represents course assignments.
* Attributes include title, description, due\_date, and associated Course.
* Methods include:
  + Grade management (add\_grade, display\_grades).
  + Calculating average grade for an assignment.
  + Checking overdue assignments.

**9. Announcement Class:**

* Represents announcements made on the platform, including the title, content, date, and recipient groups (e.g., Students, Instructors, Admins).
* Methods for:
  + Adding/removing recipient groups.
  + Checking group membership for the announcement.

**10. DiscussionThread Class:**

* Represents discussion threads in a course.
* Attributes include title, creator, posts (list of messages), and associated Course.
* Methods for:
  + Adding posts.
  + Displaying discussion details.

**11. PlatformAdmin Class:**

* Handles administrative features such as managing users, courses, and enrollments.
* Includes functionality for:
  + Admin authentication (admin\_login).
  + Loading and saving platform data from a JSON file.
  + Ensures courses, students, and instructors persist across sessions.

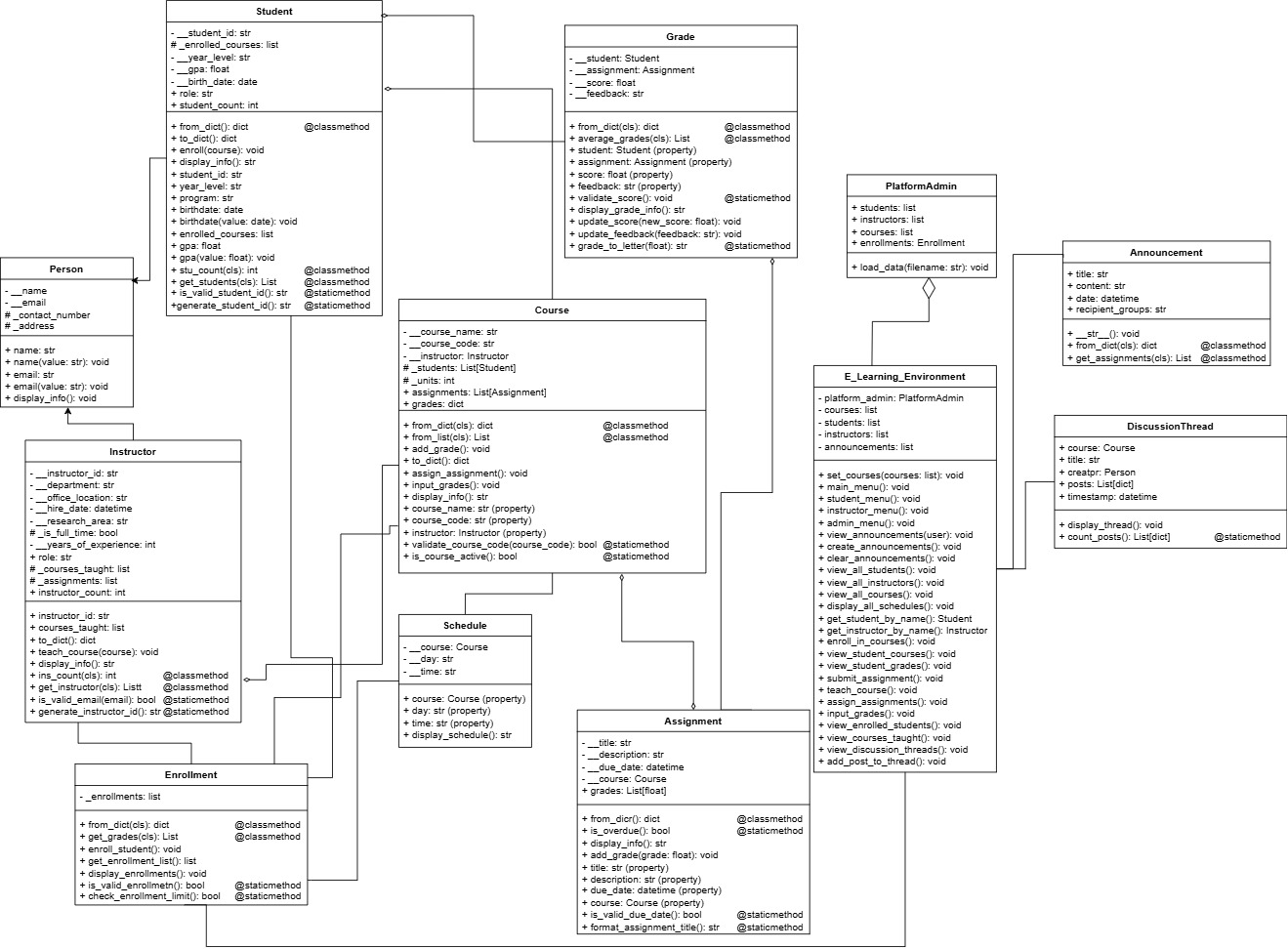
**12. E-Learning\_Environment Class:**

* Acts as the main interface for the system.
* Handles user interactions with separate menus for:
  + Students (view grades, enroll, submit assignments, etc.).
  + Instructors (manage courses, assign grades, view students, etc.).
  + Admins (view/manage users, make announcements).
* Features platform-level functionality such as mission/vision statements and system-wide data persistence.

**Key Features:**

1. **Data Persistence:**
   * Uses JSON to save and load data for students, instructors, courses, and enrollments.
2. **User Authentication:**
   * Provides authentication for admin users.
3. **Menus and Interaction:**
   * Includes command-line menus for different user roles (Student, Instructor, Admin).
4. **Validation:**
   * Validates input for email addresses, grades, GPA, unique enrollments, etc.
5. **Object-Oriented Design:**
   * Uses inheritance, encapsulation, and polymorphism to structure the system.
6. **Hierarchical Relationships:**
   * Establishes relationships among roles (e.g., Instructor teaches Courses, Students enroll in Courses, etc.).

**Class Diagram**





**1. Person Class Relationships**

**Inheritance:** Student and Instructor inherit from the Person class (generalization relationship). This is single inheritance, where both child classes reuse Person's common attributes (name, email, etc.) and methods (display\_info()).

**2. Student Class Relationships**

**Association with Course:** A student can enroll in many courses, and each course can have multiple students. This is a many-to-many association represented via the Enrollment class.

**Association with Grade:** Each Student is associated with multiple grades. Grades are specific to assignments in courses.

**3. Instructor Class Relationships**

**Inheritance from Person:** Instructors inherit attributes and behaviors from the Person class (e.g., name, email).

**Association with Course:** An instructor can teach multiple courses, but each course is taught by only one instructor. This is a one-to-many association.

**Association with Assignment:** Instructors create and manage assignments for their courses.

**4. Course Class Relationships**

**Aggregation:**

**Assignment:** A course is composed of multiple assignments, but an assignment can exist independently (aggregation relationship).

**Schedule:** Each course has a schedule, and a schedule can exist independently.

**Association with Student:** Students can enroll in a course via the Enrollment class.

**Association with Instructor:** A course is taught by one instructor.

**5. Enrollment Class Relationships**

**Association with Student and Course:** The Enrollment class serves as a linking table for the many-to-many relationship between Student and Course. A student can enroll in multiple courses, and each course can have multiple students.

**6. Assignment Class Relationships**

**Aggregation by Course:** Assignments belong to courses. A course can have many assignments, and an assignment can exist independently (aggregation relationship).

**Association with Grade:** Each assignment has grades for the students who complete it.

**7. Grade Class Relationships**

**Association with Student:** A grade is assigned to a student for completing a specific assignment.

**Association with Assignment:**Each grade is linked to a specific assignment.

**8. DiscussionThread Class Relationships**

**Association with Course:** Each discussion thread is part of a course, but a course can have many discussion threads.

**Association with Student (indirect):** Students create posts in discussion threads.

**9. Schedule Class Relationships**

**Aggregation by Course:** A course has a schedule, which specifies the time and day for the course. The schedule exists independently of the course.

**10. PlatformAdmin Class Relationships**

**Association with All Classes:** The PlatformAdmin manages all major components of the e-learning environment, including students, instructors, courses, and enrollments.

**11. E\_Learning\_Environment Class Relationships**

**Aggregation:**

**E\_Learning\_Environment** aggregates all major components:

students

instructors

courses

announcements

discussions

enrollments

It represents the entire system and ensures all components work together.

**12. Announcement Class Relationships**

**Association with Student and Instructor:** Announcements can be targeted to specific students or instructors via the recipient attribute.

Code Snippets

A computer screen shot of a code

Description automatically generatedThe `Person` class is an abstract parent class that provides the foundational structure for `Student` and `Instructor`. It ensures all "person-like" entities in the system have common attributes such as name, email, contact number, and address, along with a contract (`display\_info`) that must be implemented in subclasses.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Explanation:**

* **Purpose:** Represents shared data and behavior of Student and Instructor.
* **Attributes:** Name, email, contact number, and address are protected or private.
* **Validation:** Ensures values like name are non-empty.
* **Abstract Method:** display\_info forces subclasses to implement a custom display format.  
    
    
    
    
    
    
    
    
    
    
    
    
    
  This extends the Person class and provides a detailed representation of a student in the e-learning system. It includes attributes like student\_id, GPA, and enrolled courses. It also provides methods to enroll a student in a course or manage grades.

A screen shot of a computer code

Description automatically generated

A computer code with black text

Description automatically generated

**Explanation:**

* **Attributes:**
  + student\_id: Unique identifier for the student.
  + year\_level and program: Define the academic progress and program of study.
  + gpa: Tracks the grade point average of a student; validation ensures it stays between 0.0–4.0.
  + enrolled\_courses: Holds a list of courses a student is enrolled in.
* **Methods:**
  + enroll: Adds a course to the student's enrolled courses if it's not already present.
  + display\_info: Customizes the display of the student's details.

This class extends the Person class, representing instructors in the system. It includes the courses they teach and methods to manage these courses.

A computer screen shot of a program

Description automatically generated

**Explanation:**

* **Attributes:**
  + instructor\_id: Unique ID for an instructor.
  + \_courses\_taught: List of courses currently taught by the instructor.
* **Methods:**
  + teach\_course: Enables linking of a course with the instructor.
  + display\_info: Displays the instructor's details and assigned courses.

Defines the structure for academic courses, managing relationships with Instructor, Student, and Assignment.

A computer screen shot of a program code

Description automatically generated  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Explanation:**

* **Attributes:**
  + course\_name and course\_code: Define course details.
  + \_\_instructor: Links an instructor to the course.
  + \_students: Manages a list of enrolled students.
  + assignments: Holds a list of assignments for the course.
* **Methods:**
  + add\_student: Handles student enrollment in the course.
  + assign\_assignment: Creates assignments tied to the course.
  + display\_info: Displays course details.

Represents assignments within courses, including a title, description, and a due\_date.

A computer screen shot of a code

Description automatically generated  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Attributes:**

* + title, description, and due\_date: Define assignment details.
  + course: Links the assignment to a course.
  + grades: Stores grades for the assignment.
* **Methods:**
  + display\_info: Displays assignment details.

This serves as the central interface for all functionalities, including user menus and platform-wide tasks.

A screenshot of a computer program

Description automatically generated

**Explanation:**

* **Attributes:**
  + courses, students, instructors, and announcements: Manage system-wide entities.
* **Methods:**
  + main\_menu: Handles role-based access through menus for students, instructors, and admins.