CECOS UNIVERSITY

Name	Muhammad Shehriyar
Student ID	CU-4786-2023 (D)
Task Title	Semester Project Summary Structures, Classes, Relations, Aggregations
Assignment #	03
Submitted To	Ma'am Arshi Parvaiz
Subject	Object Oriented Programming

Note: This Assignment contain more than enough details and wordings from ChatGPT

It also contains some assumed classes, structures, and functions

Cecos University of IT and Emerging Sciences Peshawar Hayatabad

Analysis and Design for Student Information and Management System

Struct student :

Attributes:

- std_name
- std_id
- std_marks
- std_grade
- std_per
- username
- Password

Represents a student with their personal and academic information.

Class Student_Info_System:

Attributes:

- admin_username
- admin_password

Methods:

- system_authorization()
- admin_authorization()
- student_login()
- management_system()
- add_records() update_records()
- delete_records()
- search_records()
- display_student_record()
- show_student_records()

Handles various functionalities such as system authorization, student login, and management operations (add, update, delete, search, display, and show records).

Global Variable :

- vector<student> student_data : Stores the list of student records.

Classes and Their Relationships:

Class Student :

Attributes:

- std_name : string
- std_id : int
- std_marks : int
- std_grade : string
- std_per : float
- Username : string
- password : int

Methods :

None in the struct, but potentially:

- calculate_grade()
- update_details()

Class Admin (New Class):

Attributes:

- username : string
- Password : int

Methods:

- authorize() : Check admin credentials

Class Course (New Class):

Attributes:

- course_id : int
- course_name : string
- Credits : int
- students: vector<Student>

Methods:

- add_student(Student)
- remove_student(Student)
- get_course_details()

Class Department (New Class):

Attributes:

- department_id : int
- department_name : string
- Courses : vector<Course>

Methods:

- add_course(Course)
- remove_course(Course)
- list_courses()

Class Student_Info_System:

Attributes:

- Admin Admin
- Students vector<Student>
- Departments vector<Department>

Methods:

- system_authorization()
- admin_authorization()

- student_login()
- management_system()
- add_records() update_records()
- delete_records()
- search_records()
- display_student_record()
- show_student_records()

Potentially: assign_student_to_course (Student, Course)

Relationships and Aggregation Between Classes:

Class Relationships and Aggregation:

Class Student_Info_System:

Aggregation with Student:

The 'Student_Info_System' manages a collection of Student objects through operations such as adding, updating, deleting, and displaying student records. It aggregates these Student objects in the 'student_data' vector to maintain and manipulate student information within the system.

Aggregation with Course:

The system interacts with multiple `Course` objects to manage student enrollments, course details, and academic records. It can add, update, and delete course information, demonstrating an aggregation of `Course` objects within its management operations.

Aggregation with `Department`:

The system oversees various academic departments, maintaining a collection of `Department` objects. This aggregation allows it to manage department-specific activities, such as course offerings, professor assignments, and departmental policies.

Class `Student`:

Aggregation with `Course`:

A `Student` can enroll in multiple `Course` objects. While a `Student` exists independently of any specific course, they are aggregated within the context of courses they are enrolled in.

Composition with `Admin`:

Each `Student` interacts with the system through an `Admin` entity during authentication and administrative processes. This composition ensures that each student's access and system interactions are mediated by administrative controls.

Class `Admin`:

Aggregation with `Student_Info_System`:

The `Admin` entity is integral to the functioning of the `Student_Info_System`. It manages system-wide administrative tasks, including user authentication, data management, and system configuration. The `Admin` aggregates control over system operations and interacts with both `Student` and `Course` objects to maintain academic and administrative integrity.

Class `Course`:

Aggregation with `Department`: A `Course` is associated with a specific `Department` within the institution. This aggregation signifies that courses are organized and administered within the context of their respective departments, influencing curriculum design, faculty assignments, and academic planning.

Class `Department`:

Aggregation with `Professor`:

Each `Department` manages a faculty comprising `Professor` objects who teach courses and contribute to departmental activities. This aggregation ensures that professors are associated with specific departments, collaborating on academic programs, research initiatives, and student mentorship.

=====The End=====