OOP project: E-learning Platform.

Concept:

Develop an E-learning platform for online courses and tutorials. The platform will include various classes to represent different aspects of the system, such as Users, Courses , Student , Admin , Professor , and ElearningPlatform. Each class will demonstrate key OOP concepts like inheritance, polymorphism ,Exception Handling , Collections , Object Composition , and encapsulation.

##Classes:

Task1: Abstract Class User

Attributes:

name: Name of the user.

email: Email address of the user.

Password : password of the user.

Methods:

Default constructor and parameter constructor.

abstract void displayInfo(): Abstract method to display user information.

Public void login(): interface methods to login user.

Public void logout(): interface methods to logout user.

toString(): Returns a string representation of the user.

equals(Object obj): Compares this user to another object.

getters and setters: For each attribute.

Task 2 : Student **(extends** User**)**

Attributes:

Id : Unique identifier for the student.

enrolledCourses: List of courses the student is enrolled in.

Methods:

Default constructor and parameter constructor.

displayInfo(): Displays student information.

Public void login(): interface methods to displays Student login.

Public void logout(): interface methods to displays Student logout.

enrollCourse(Course course): Enrolls the student in a course.

dropCourse(Course course): Drops the student from a course.

toString(): Returns a string representation of the student.

equals(Object obj): Compares this student to another object.

getters and setters: For each attribute.

AdditionalMethods: viewGrades(), submitAssignment(), participateInDiscussion()

Task 3 : Professor (extends User**)**

Attributes:

Id : Unique identifier for the professor.

taughtCourses: List of courses the Professor teaches.

Methods:

Default constructor and parameter constructor.

displayInfo(): Displays Professor information.

Public void login(): interface methods to displays Professor login.

Public void logout(): interface methods to displays Professor logout.

addCourse(Course course): Adds a course to the Professor list.

removeCourse(Course course): Removes a course from the Professor list.

toString(): Returns a string representation of the Professor.

equals(Object obj): Compares this Professor to another object.

getters and setters: For each attribute.

Additional Methods: gradeAssignment(), createLecture(), scheduleOfficeHour

Task 4: Course

Attributes:

courseId: Unique identifier for the course.

courseName: Name of the course.

Professor : object of professor teaching the course.

students: List of students enrolled in the course.

Duration : Course duration.

Methods:

Default constructor and parameter constructor

addStudent(Student student): Adds a student to the course.

removeStudent(Student student): Removes a student from the course.

toString(): Returns a string representation of the course.

equals(Object obj): Compares this course to another object.

getters and setters: For each attribute.

Additional Methods: startCourse(), endCourse(), getCourseDetails

.Task 5 : Admin **(extends** User**)**

Attributes:

Id : Unique identifier for the admin.

managedCourses: List of courses managed by the admin.

Methods:

Default constructor and parameter constructor

displayInfo(): Displays admin information.

Public void login(): interface methods to displays admin login.

Public void logout(): interface methods to displays admin logout.

addCourse(Course course): Adds a course to the admin's list.

removeCourse(Course course): Removes a course from the admin's list.

toString(): Returns a string representation of the admin.

equals(Object obj): Compares this admin to another object.

getters and setters: For each attribute.

Additional Methods: manageUsers(), generateReports(), updateCourseDetails

Task 6: Main Class

Create Objects: Instantiate objects for Student, Professor, Course, and Admin.

Array of Objects: Use an array or list to manage multiple Course objects.

User Input: Read information from the user to fill object attributes.

()Print Information: Display object information using toString.

Special Tasks: Implement tasks like enrolling a student in a course, displaying course details, and managing user logins and logouts.

Applying OOP Concepts

Polymorphism: Use method overriding in Student, Professor, and Admin classes.

Inheritance: Student, Professor, and Admin inherit from User.

Exception Handling: Handle exceptions for invalid inputs or operations.

Collections: Use lists to manage students and courses.

Object Composition: Course contains Professor and Student objects.