Software Development Exercise 1 (10%)

Scenario: Building on the UCMS concept discussed in Week 04 Lecture

In this assignment, you will design and implement classes for two key modules of the UCMS: Assessment and User Management, demonstrating key OOP principles: encapsulation, inheritance, polymorphism, abstraction, and composition.

Attachments: Some tasks require you to build on existing classes. A zip file named ***"ExerciseOneResources.zip"*** contains the following Java classes which you will use and extend (where necessary) to complete this exercise: *User.java, Lecturer.java, Student.java, Module.java, Course.java*.

Task 1: Designing the Assessment Module

Design and implement the module for managing students' assessments. Consider what data and behaviours are essential for different assessment types and how OOP principles can be applied effectively.

1. Design and implement suitable classes for three main assessment types: Final Exam, Test, and Assignment. Demonstrate appropriate use of OOP principles.
   - Your classes may include attributes such as:
     - Assessment ID
     - Module (consider using your Module class)
     - Student (consider using your Student class)
     - Total marks
     - Weight toward final module marks
     - Duration (for exams)
     - Number of questions
     - Date taken, released date, due date
     - Instructor/lecturer in charge

     Note: Some properties may not be relevant for some assessment types.

   - Behaviours for an assessment include:
     - Displaying assessment details.
     - Checking whether assessment is overdue.
     - Calculating scores.

Deliverables:

- UML diagrams feasible relationships between Assessment classes, Module, and Student.
- Java classes implementing your design.
- Include explanations in your report describing design decisions, highlighting how OOP principles are applied and how classes relate/interact.

Task 2: Implement Admin Class

Extend the User Management Module by adding an Admin class. Think critically about what attributes and responsibilities an admin class should have, how it interacts with students, lecturers, modules, and courses, and how to structure the class to demonstrate OOP principles.

1. Design and implement a blueprint for instantiating admin objects that captures inheritance, encapsulation, polymorphism, and composition.
   - Typical attributes to consider:
     - User ID, name, email, address, phone number, date of birth.
   - Some responsibilities include (but limited to):
     - Adding, viewing, updating, and removing students and lecturers
     - Creating and managing courses and modules. You may use classes included in the ***ExerciseOneResources.zip***.
     - Assign modules to lecturers.

Deliverables:

- UML diagram showing the Admin class and its relationships with other classes.
- Java implementation with at least three properly chained constructors.
- Document your design decisions and use of OOP principles in the report.

Task 3: Testing Classes

Create a Main class to demonstrate realistic system interactions and OOP Principles.

1. Instantiate at least two objects each for Student, Lecturer, and Admin, demonstrating constructor overloading and chaining.
2. Demonstrate encapsulation using getters and setters.
3. Demonstrate polymorphism in handling different assessment types.
4. Simulate Admin operations including add/remove users, assign modules, manage courses.
5. Simulate Lecturer interactions (view assigned modules, create/manage assessments, record marks).

Submission Instructions

- Save all your files in one folder and compress it into a single .zip file. Upload your completed work to Blackboard under the assignment link for this module.
- Please include the following items in your submission:
  - All your Java source files (.java). This should include your Assessment Module classes, Admin class, and Main test class.
  - Your UML diagrams (saved as .pdf, .png, or .jpg).
  - A short ReadMe or documentation file (.pdf or .docx) explaining:
    - How your classes are structured
    - How you applied the key OOP principles
- Submit your own original work. Copying or sharing code is considered plagiarism and will be reported according to the academic integrity policy.

Rubric Overview (10 %)

Criteria	Description	Marks
Assessment Module Design	- Design and implement assessment classes that demonstrate OOP principles: inheritance, abstraction, polymorphism, and composition. - Include UML diagrams (attach in report)	3%
Admin Class Design	- Correctly extend the User class. - Implement Admin responsibilities with appropriate attributes and methods. - Include UML diagram (attach in report)	3%
Testing and System Simulation	- Create a Main class that demonstrates realistic interactions among students, lecturers, admin, courses, modules, and assessments. - Meaningful outputs to demonstrate that your simulation works.	2%

Short documentation	- A short ReadMe or documentation (.pdf) about:
	  - How your classes are structured along with design decisions.
	  - How you applied the key OOP principles Attach UML diagrams
		2%