SECJ2154 Object Oriented Programming
Mini Project (10%)

Guidelines:

- Form groups of 2 or 3 members.
- Choose one topic and one concept to implement within the selected topic.
- Utilize concepts learned, such as encapsulation, inheritance, polymorphism using packages, arrays/vectors, etc.
- Evaluation will be based on the incorporation of concepts, creativity in programming, and the efficiency of your code.

- Your system MUST apply:

		Weightage (Marks)
1.	At least THREE related classes (10% x classes) with their constructors, mutators and accessors (10%)	40%
2.	Relationships between classes (at least one of them is inheritance)	10%
3.	Array of objects OR ArrayList OR Vector	10%
4.	Abstract class, Interface and implement them (at least ONE class must be abstract, ONE interface and the rest can be solid/concrete)	20%
5.	Polymorphism through overriding methods	10%
6.	Overloading methods	10%
7.	Package (option only)	10%
	Total	100% (+10%)

Important Dates:
Submit the proposed UML diagram that show relationship among the classes – **16 Dec 2023**
Demo & Presentation – **Meeting 9 (14 Jan 2024)**

Deliverables: Report & Source Code

The report must contain:
1. Introduction to your system
2. Class Diagrams with their relationships (association, composition, aggregation & inheritance)
3. Details about the class members (data attributes and methods)
4. Explanation about the concepts used. For example, show where you applied the concept of polymorphism, etc.
5. Program Listing

Topics

1. **Hotel reservation system**

The program should able to :

- add, edit and delete information of the rooms available in this hotel
- add, edit and delete personal information of the customer
- manage the details about the occupied rooms books such as startDate, endDate, customerId, etc.

Example of classes (example only, you can add more)

Class: **room**
Data member: type, rate, description, status(available, booked, occupied, etc), etc.
Methods: constructors, accessors, mutators, etc.

Class: **customer**
Data member: id, name, address, etc.
Methods: constructors, accessors, mutators, bookRoom, checkIn, checkOut, etc.

2. **Online shopping management system**

The program should able to :

- add, edit and delete information of the items available in this shop
- add, edit and delete personal information of the customers
- manage the details about the customers' online transations such as transactionId, date, customerId, itemId, unit, calcItemPrice, calcTotalPrice, etc.

Example of classes (example only, you can add more)

Class: **item**
Data member: itemId, itemName, itemPrice, unitItem, etc.
Methods: constructors, add, delete, accessors, mutators, etc.

Class: **customer**
Data member: id, name, address, etc.
Methods: constructors, accessors, mutators, buyItems, etc.

3. **Clinic management system**

The program should able to :

- add, edit and delete information of the medicine available in this clinic
- add, edit and delete personal information of the patients

- manage the details about the customers' visiting such as date, patientId, treatment, etc.

Example of classes (example only, you can add more)

Class: **patient**
Data member: id, name, address, etc.
Methods: constructors, accessors, mutators, register, etc.

Class: **medicine**
Data member: code, name, description, etc.
Methods: constructors, accessors, mutators, add, delete, etc.

4. Bus ticketing system

The program should able to :

- add, edit and delete information of the tickets available in this company
- add, edit and delete personal information of the customer
- manage the details about the booked/bought tickets such as date, time, customerId, etc.

Example of classes (example only, you can add more)

Class: **ticket**
Data member: type, rate, description, status(available, booked, paid, etc), etc.
Methods: constructors, accessors, mutators, etc.

Class: **passanger**
Data member: id, name, address, etc.
Methods: constructors, accessors, mutators, book, pay, etc.

5. Staff management system

The program should able to :

- add, edit and delete personal information of the staff and calculate their salary etc.
- manage the details about the customers' duty such as date, startTime, endTime, staffId, etc.

Example of classes (example only, you can add more)

Class: **staff**
Data member: staffId, name, address, contact no., etc.
Methods: add, delete, etc.

Class: **task**
Data member: taskId, name, description, ratePerHour, etc.
Methods: add, edit, delete, etc.

6. **Course registration system**

The program should able to :

- add, edit and delete information of the courses offered by this company
- add, edit and delete personal information of the participants
- manage the details about the registered participants such as courseId, date, participantId, itemId, unit, calcItemPrice, calcTotalPrice, etc.

Example of classes (example only, you can add more)

Class: **course**
Data member: coursed, courseName, courseFee, instructorId, etc.
Methods: constructors, add, delete, accessors, mutators, etc.

Class: **participants**
Data member: participantId, participantName, address, etc.
Methods: constructors, accessors, mutators, registerCourse, payCourse, etc.

And also, can consider the following topics with the related classes accordingly

7. Car rental management system
8. Student management system
9. Meeting Room reservation system

Example

10. Mini library system

The program should be able to :

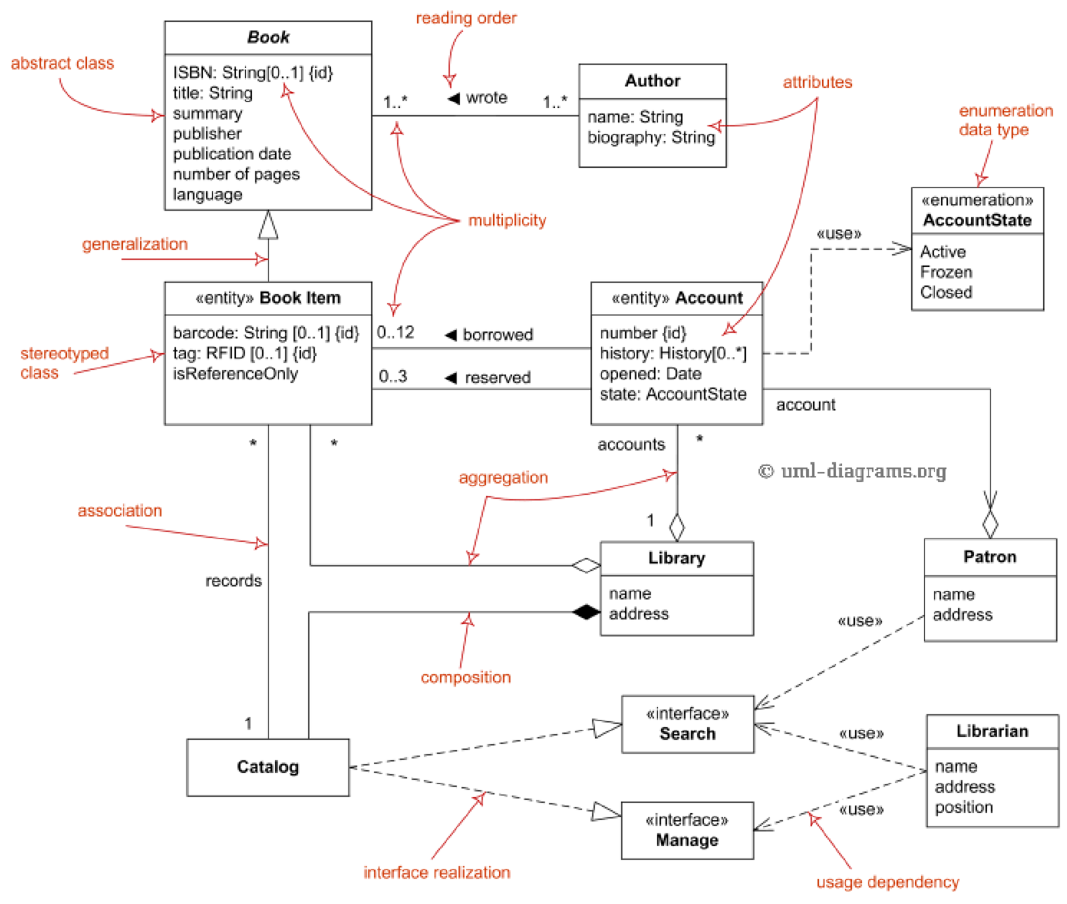
- add, edit and delete information of the books available in this library
- add, edit and delete personal information of the borrowers
- manage the details about the borrowed books such as startDate, endDate, borrowerId, etc.

Example of classes (example only, you can add more)

Class: **Book**
Data member: id, name, description, isbn, subject, status, etc.
Methods: constructors, accessors, mutators , etc.

Class: **Borrower**
Data member: id, name, etc.
Methods: constructors, accessors, mutators, borrowBook, returnBook, etc.

Class: **Author**
Data member: id, name, etc.
Methods: constructors, accessors, mutators, writeBook, etc.



reading order
abstract class
Book
ISBN: String[0..1] {id}
title: String
summary
publisher
publication date
number of pages
language
1..*
◀ wrote
1..*
Author
name: String
biography: String
attributes
enumeration data type
«enumeration» AccountState
Active
Frozen
Closed
«use»
multiplicity
generalization
«entity» Book Item
barcode: String [0..1] {id}
tag: RFID [0..1] {id}
isReferenceOnly
stereotyped class
0..12
◀ borrowed
0..3
◀ reserved
«entity» Account
number {id}
history: History[0..*]
opened: Date
state: AccountState
account
accounts
*
*
*
© uml-diagrams.org
aggregation
association
1
Library
name
address
Patron
name
address
records
«use»
composition
1
Catalog
«interface» Search
«interface» Manage
«use»
«use»
Librarian
name
address
position
interface realization
usage dependency