

**LAB PROBLEM 1: Class Diagram for Library System (Any Four)**

**Topic:** UML Class Diagram – Structure Overview

**Problem Statement:**

Draw a UML **Class Diagram** for a simple Library Management System with classes like Book, Member, and Librarian. Show relationships such as association and aggregation.

**Hints:**

● Use **attributes and methods** inside each class.

● Show **1-to-many** association between Member and Book.

● Mark the relationship direction using arrows.

| @startuml class Book {  -bookId: int  -title: String  -author: String  +getDetails(): void }  class Member {  -memberId: int  -name: String  +borrowBook(): void  +returnBook(): void }  class Librarian {  -librarianId: int  -name: String  +addBook(): void  +removeBook(): void }  Member "1" -- "many" Book : borrows > Librarian "1" o-- "many" Book : manages > @enduml |
| --- |

**LAB PROBLEM 2: Object Diagram for Student-Teacher Relationship**

**Topic:** UML Object Diagram – Runtime Instances

**Problem Statement:**

Draw an **Object Diagram** representing real instances of classes Student and Teacher where each teacher guides two students.

**Hints:**

● Show object names (e.g., teacher1:Teacher, student1:Student). ● Indicate **object links** (runtime relationships).

● Keep attribute values simple (e.g., name = "Karthik").

| @startuml object teacher1:Teacher {  name = "Anita" }  object student1:Student {  name = "Karthik" }  object student2:Student {  name = "Diya" }  teacher1 -- student1 teacher1 -- student2 @enduml |
| --- |

1



**LAB PROBLEM 3: Sequence Diagram for Online Order Process**

**Topic:** UML Sequence Diagram – Method Interaction Over Time

**Problem Statement:**

Draw a **Sequence Diagram** showing the process of placing an order on an e-commerce website. Include Customer, OrderService, PaymentGateway, and InventoryService.

**Hints:**

● Show the **flow of method calls** from customer to services.

● Include **return arrows** to indicate responses.

● Use **activation boxes** for ongoing operations.

| @startuml actor Customer participant OrderService participant PaymentGateway participant InventoryService  Customer -> OrderService: placeOrder() OrderService -> InventoryService: checkStock() InventoryService --> OrderService: stockAvailable() OrderService -> PaymentGateway: processPayment() PaymentGateway --> OrderService: paymentSuccess() OrderService --> Customer: orderConfirmed() @enduml |
| --- |

**LAB PROBLEM 4: Use Case Diagram for ATM System**

**Topic:** UML Use Case Diagram – User Interaction

**Problem Statement:**

Draw a **Use Case Diagram** showing the main user actions in an ATM system such as Withdraw Money, Check Balance, and Deposit Money.

**Hints:**

● Use **actor symbols** for users.

● Connect actors to use cases with lines.

● Optionally use **include** or **extend** relationships.

| @startuml actor User rectangle ATMSystem {  usecase "Withdraw Money" as UC1  usecase "Check Balance" as UC2  usecase "Deposit Money" as UC3 }  User --> UC1 User --> UC2 User --> UC3 @enduml |
| --- |

2



**LAB PROBLEM 5: Activity Diagram for Student Registration Process**

**Topic:** UML Activity Diagram – Workflow Representation

**Problem Statement:**

Draw an **Activity Diagram** that represents the steps involved in student registration — from form filling to admin approval and confirmation.

**Hints:**

● Start with an **initial node** (solid circle).

● Show decision branches (e.g., valid data / invalid data).

● End with a **final node** (solid circle with outer ring).

**LAB PROBLEM 6: State Diagram for Order Lifecycle**

**Topic:** UML State Diagram – State Transitions

**Problem Statement:**

Draw a **State Diagram** for an order showing transitions from New → Processing → Shipped → Delivered → Cancelled.

**Hints:**

● Use **arrows** to show transitions.

● Label transitions with **event names**.

● Include **initial and final states**.

3