
Library Management System

Scenario-Based Assignment (Strict Task Instructions)

TASK 1: Abstract Class & Abstract Methods

What the student MUST create

1. **ONE abstract base class** named:
    - `LibraryItem`
2. The abstract class MUST contain:
    - **Exactly 3 properties**
        - Title
        - Author
        - ItemID
    - **Exactly 2 abstract methods**
        - One method to display item details
        - One method to calculate late fee
3. **TWO concrete child classes**:
    - `Book`
    - `Magazine`
4. Each child class MUST:
    - Inherit from `LibraryItem`
    - Override **both abstract methods**
5. Late fee rules (MANDATORY):
    - Book → 1 unit per day
    - Magazine → 0.5 units per day

What the student MUST show as proof

- Create **1 Book object** and **1 Magazine object**
- Display:
    - Title
    - Author

    - ItemID
    - Late fee for **exactly 3 days**

TASK 2: Interfaces & Multiple Inheritance

What the student MUST create

1. **ONE interface** named:
    - `IReservable`
    - Must contain **one method** for reserving an item
2. **ONE interface** named:
    - `INotifiable`
    - Must contain **one method** that accepts a message
3. The `Book` class MUST:
    - Implement **both interfaces**
    - Implement **all interface methods**

What the student MUST show as proof

- Call:
    - The reservation method
    - The notification method
- Output MUST clearly indicate:
    - Reservation success
    - Notification message sent

TASK 3: Dynamic Polymorphism

What the student MUST create

1. **ONE collection** that stores:
    - Objects of type `LibraryItem`
2. Add:
    - **1 Book**
    - **1 Magazine**
3. Use:
    - A loop that accesses items **only using the parent type**

What the student MUST show as proof

- The correct `DisplayItemDetails()` method must execute:
  - Book version for Book
  - Magazine version for Magazine
- Student MUST write **1–2 lines explanation** stating:
  - "Method selection happens at runtime"

TASK 4: Explicit Interface Implementation

What the student MUST create

1. Modify `Book` class so that:
   - Interface methods are **explicitly implemented**
2. The class MUST NOT allow:
   - Direct calling of reservation or notification methods using Book object

What the student MUST show as proof

- Call interface methods using:
  - `IReservable` reference
  - `INotifiable` reference
- Student MUST write **1 clear sentence** explaining:
  - Why direct access is restricted

TASK 5: Namespaces & Nested Namespaces

What the student MUST create

1. **ONE main namespace**:
   - `LibrarySystem`
2. Inside it, create **TWO nested namespaces**:
   - `Items`
   - `Users`
3. Place:
   - `Book` and `Magazine` inside `Items`
   - One user class (`Member`) inside `Users`
4. Create **ONE namespace alias**:

- Alias `LibrarySystem.Items`

What the student MUST show as proof

- Use alias to create Book and Magazine objects
- Write **2 points** explaining:
  - Why nested namespaces are useful

TASK 6: Partial & Static Classes

What the student MUST create

1. **ONE partial class**:
   - `LibraryAnalytics`
2. The class MUST be split into:
   - **Two separate parts**
3. The class MUST contain:
   - **ONE static property** to track total borrowed items
   - **ONE static method** to display analytics

What the student MUST show as proof

- Increase borrowed count
- Display total borrowed items
- Write **1 line** explaining:
  - Why static members are used

TASK 7: Enumerations (Enums)

What the student MUST create

1. **ONE enum** named:
   - `UserRole`
     - Admin
     - Librarian
     - Member
2. **ONE enum** named:
   - `ItemStatus`

        - Available
        - Borrowed
        - Reserved
        - Lost
3. Assign:
    - A role to a user
    - A status to a library item

What the student MUST show as proof

- Display:
    - User role
    - Item status
- Write **1 sentence** explaining:
    - Why enums are better than strings

BONUS TASK (HIGH WEIGHTAGE)

What the student MUST create

1. Modify notification logic so that:
    - Admin → system alert
    - Member → borrowing update
2. Add **ONE new item type**:
    - eBook
3. eBook MUST:
    - Inherit from `LibraryItem`
    - Have **one additional digital-specific behavior**

What the student MUST show as proof

- Demonstrate:
    - Notification difference by role
    - eBook behavior execution

Below is the **EXPECTED OUTCOME / EXPECTED OUTPUT** for **EACH TASK**, written in a **clear, observable, examiner-friendly format**.

EXPECTED OUTCOME

Library Management System Assignment

TASK 1: Abstract Class & Abstract Methods

Expected Outcome (What MUST appear)

Displayed Output:

Item Type: Book
Title: C# Fundamentals
Author: John Doe
Item ID: 101
Late Fee for 3 days: 3.0

Item Type: Magazine
Title: Tech Today
Author: Jane Doe
Item ID: 201
Late Fee for 3 days: 1.5

Concept Confirmation:

- Book late fee = 3 days × 1 = 3.0
- Magazine late fee = 3 days × 0.5 = 1.5

What this proves:

- Abstract methods are correctly overridden
- Different logic is applied based on item type

TASK 2: Interfaces & Multiple Inheritance

Expected Outcome

Displayed Output:

Book reserved successfully.
Notification sent: Your reserved book is ready for pickup.

What this proves:

- One class implements **multiple interfaces**
- Optional behaviors are added without inheritance misuse

TASK 3: Dynamic Polymorphism

Expected Outcome

Displayed Output (from loop):

Item Type: Book
Title: C# Fundamentals
Author: John Doe
Item ID: 101

Item Type: Magazine
Title: Tech Today
Author: Jane Doe
Item ID: 201

Mandatory Student Explanation (1–2 lines):

The method executed depends on the object type at runtime, not the reference type.

What this proves:

- Runtime polymorphism is working
- Correct method binding occurs dynamically

TASK 4: Explicit Interface Implementation

Expected Outcome

Displayed Output:

Book reserved successfully.
Notification: Please return the book on time.

How it must be called (conceptually):

- Through `IReservable` reference
- Through `INotifiable` reference

Mandatory Explanation (1 sentence):

Explicit implementation prevents direct access and exposes functionality only through interfaces.

What this proves:

- Controlled access
- Proper use of explicit interface implementation

TASK 5: Namespaces & Nested Namespaces

Expected Outcome

Successful Object Creation Using Alias

Book and Magazine objects created using namespace alias.

Mandatory Explanation (2 points):

1. Nested namespaces organize large projects logically.
2. Aliases reduce long namespace references and improve readability.

What this proves:

- Correct namespace hierarchy
- Professional project organization

TASK 6: Partial & Static Classes

Expected Outcome

Displayed Output:

Total Items Borrowed: 5

Mandatory Explanation (1 line):

Static members store system-wide data shared across all objects.

What this proves:

- Static data persists globally
- Partial classes are merged into one logical unit

TASK 7: Enumerations (Enums)

Expected Outcome

Displayed Output:

User Role: Member
Item Status: Borrowed

Mandatory Explanation (1 sentence):

Enums prevent invalid values and improve code readability.

What this proves:

- Controlled value sets
- Safer and cleaner logic

BONUS TASK: Advanced Design

Expected Outcome

Notification Based on Role:

Admin Alert: System maintenance scheduled.
Member Notification: Your borrowed item is due tomorrow.

eBook Behavior Output:

eBook downloaded successfully.

What this proves:

- Role-based behavior variation
- Digital item extension without breaking design
- Scalable and extensible architecture

FINAL EVALUATION CHECKLIST (For Faculty)

Concept	Verified By
Abstraction	Different late fee outputs
Interfaces	Reservation + Notification
Polymorphism	Correct runtime method calls
Explicit Interface	Interface-only access
Namespaces	Alias usage
Static	Shared analytics
Enums	Controlled roles/status
Design Quality	Bonus task
