Scenario 1: One-to-Many — Student and Assignments

Assignment Class

- **Fields/Properties:**
  - title *(string)* → Property: Title
  - dueDate *(DateTime)* → Property: DueDate
  - score *(double)* → Property: Score
- **Default Constructor:**
- **Parameterized Constructor:**
  Accepts title, dueDate, and score as arguments and sets the respective fields.
- **Method:**
  ShowDetails()
  Purpose: Displays assignment information including title, due date, and score.

◆ **Student Class**

- **Fields/Properties:**
  - studentId *(int)* → Property: StudentId
  - name *(string)* → Property: Name
  - assignments *(List<Assignment>)* → Property: Assignments
- **Default Constructor:**
- **Parameterized Constructor:**
  Accepts studentId, name, and optionally a list of assignments; if not provided, initializes an empty list.
- **Method:**
  GetPendingAssignments()
  Purpose: Returns a list of assignments from the Assignments list where DueDate is in the future.
- **Method:**
  ShowDetails()
  Purpose: Displays the student's ID, name, and the total number of assignments.

Scenario 2: One-to-One — Student and LibraryCard

Association Type: One-to-One

Classes Involved: Student, LibraryCard

🔹 LibraryCard Class

- **Fields/Properties:**
    - cardNumber *(string)* → Property: CardNumber
    - issueDate *(DateTime)* → Property: IssueDate
    - isActive *(bool)* → Property: IsActive
- **Default Constructor:**
- **Parameterized Constructor:**

  Accepts all 3 fields and sets the respective properties.
- **Method:**

  ShowDetails()

  Purpose: Displays card number, issue date, and activation status.

🔹 Student Class

- **Fields/Properties:**
    - studentId *(int)* → Property: StudentId
    - name *(string)* → Property: Name
    - libraryCard *(LibraryCard)* → Property: LibraryCard
- **Default Constructor:**
- **Parameterized Constructor:**

  Accepts studentId, name, and libraryCard as arguments.
- **Method:**

  ActivateLibraryCard()

  Purpose: Sets the IsActive property of LibraryCard to true.
- **Method:**

  ShowDetails()

  Purpose: Displays student info and calls LibraryCard.ShowDetails().

Scenario 3: Online Payment System

Concept: Interface + Abstract Class + Concrete Class

- **Interface:** IPayable
  - **Method:** bool ProcessPayment(double amount)
- **Abstract Class:** Payment implements IPayable
  - **Fields/Properties:** Amount, TransactionId
  - **Abstract Method:** bool Validate()
- **Class:** CreditCardPayment inherits Payment
  - Implements Validate() — checks if card details are valid.
  - Implements ProcessPayment() — processes the payment if valid.
- **Purpose:** To model a payment system with validation and interface-based payment execution.
- **Write default and parameterized constructor for classes.**

Scenario 4: Student Activities

Concept: Interface + Regular Classes

- **Interface: IActivity**
- **Method:** void Participate()

- **Regular Class: Sports implements IActivity**
- **Fields/Properties:**
    - SportName *(string)*
    - TeamSize *(int)*
- **Implements:** Participate() — Print "Playing [SportName] in a team of [TeamSize]."

- **Regular Class: DramaClub implements IActivity**
- **Fields/Properties:**
    - PlayTitle *(string)*
    - Role *(string)*
- **Implements:** Participate() — Print "Acting as [Role] in [PlayTitle]."
- **Write default and parameterized constructor and ShowDetails() method for classes.**

Scenario 5: Student Report Generation

Concept: Interface + Abstract Class + Regular Class

- **Interface: IPrintable**
- **Method:** void PrintReport()

- **Abstract Class: Report implements IPrintable**
- **Fields/Properties:**
  - ReportTitle *(string)*
  - CreatedDate *(DateTime)*
- **Abstract Method:** string GenerateContent()

- **Regular Class: StudentReport inherits Report**
- **Additional Fields/Properties:**
  - StudentName *(string)*
  - AverageScore *(double)*
- **Implements:**
  - GenerateContent() — Returns summary string.
  - PrintReport() — Prints formatted report.

- **Write default and parameterized constructor and ShowDetails() method for classes.**