Library Management System (Console-Based)

1. Getters and setters.

Ngl most of the stuff are same like java in C#, but some of them were different like, the way how the getters and setters are handled.

public int Id { get; set; }

public string Title { get; set; }

public string Author { get; set; }

public bool IsAvailable { get; set; } = true;

1. No CAMEL CASE ???

C# follows Microsoft's official naming conventions, which recommend:

**Properties, methods, and public fields** ➜ PascalCase → Title, Author, IsAvailable

**Private fields and variables** ➜ camelCase → title, author, \_isAvailable

PascalCase = Each word starts with a capital (e.g., FirstName)

camelCase = First word lowercase, next words capitalized (e.g., firstName)

1. See how Override looks

public override string ToString()

        {

            return $"{Id}.{Title} by {Author} - {(IsAvailable ? "Available" : "Borrowed")}";

        }

**✅ 1. using System.Collections.Generic;**

This is a **namespace import**, similar to Java’s import statement.  
It gives access to **generic collection types** like:

* List<T>
* Dictionary<TKey, TValue>
* Queue<T>, etc.

So here, it's needed to use List<User>.

**2. public List<User> Users { get; }**

This defines a **read-only auto-property** named Users:

* public → Anyone can access this property.
* List<User> → It’s a list (dynamic array) that stores User objects.
* { get; } → It has a public getter (readable), but **no setter** — meaning it **can't be replaced**, only **modified** (i.e., you can add/remove users, but can't assign a new list).

**🟩 public class Admin : User**

* This defines a class called Admin that **inherits** from a base class called User.
* : is used to specify **inheritance** in C# (like extends in Java).

**🟩 public Admin(int id, string name) : base(id, name)**

This is the **constructor** for the Admin class, and it uses : base(...) to **call the parent (User) constructor**.

* Admin(int id, string name) → Constructor that takes two arguments.
* : base(id, name) → Calls the constructor of the **User** class and passes id and name to it.

This is called **constructor chaining**.