**Practical coding improvement**

**Problem Statement**

The Library Management System works, but it could be optimized. Use Microsoft Copilot to enhance the code's readability, reduce repetition, and make it more efficient. You will:

* Remove code duplication.
* Improve input validation.
* Address case sensitivity issues for entering and removing books.

**Starting Code to Input into Copilot:**

**class** **LibraryManager**

{

**static** **void** Main()

    {

**string** book1 = "";

**string** book2 = "";

**string** book3 = "";

**string** book4 = "";

**string** book5 = "";

**while** (**true**)

        {

            Console.WriteLine("Would you like to add or remove a book? (add/remove/exit)");

**string** action = Console.ReadLine().ToLower();

**if** (action == "add")

            {

**if** (!**string**.IsNullOrEmpty(book1) && !**string**.IsNullOrEmpty(book2) && !**string**.IsNullOrEmpty(book3) && !**string**.IsNullOrEmpty(book4) && !**string**.IsNullOrEmpty(book5))

                {

                    Console.WriteLine("The library is full. No more books can be added.");

                }

**else**

                {

                    Console.WriteLine("Enter the title of the book to add:");

**string** newBook = Console.ReadLine();

**if** (**string**.IsNullOrEmpty(book1))

                    {

                        book1 = newBook;

                    }

**else** **if** (**string**.IsNullOrEmpty(book2))

                    {

                        book2 = newBook;

                    }

**else** **if** (**string**.IsNullOrEmpty(book3))

                    {

                        book3 = newBook;

                    }

**else** **if** (**string**.IsNullOrEmpty(book4))

                    {

                        book4 = newBook;

                    }

**else** **if** (**string**.IsNullOrEmpty(book5))

                    {

                        book5 = newBook;

                    }

                }

            }

**else** **if** (action == "remove")

            {

**if** (**string**.IsNullOrEmpty(book1) && **string**.IsNullOrEmpty(book2) && **string**.IsNullOrEmpty(book3) && **string**.IsNullOrEmpty(book4) && **string**.IsNullOrEmpty(book5))

                {

                    Console.WriteLine("The library is empty. No books to remove.");

                }

**else**

                {

                    Console.WriteLine("Enter the title of the book to remove:");

**string** removeBook = Console.ReadLine();

**if** (removeBook == book1)

                    {

                        book1 = "";

                    }

**else** **if** (removeBook == book2)

                    {

                        book2 = "";

                    }

**else** **if** (removeBook == book3)

                    {

                        book3 = "";

                    }

**else** **if** (removeBook == book4)

                    {

                        book4 = "";

                    }

**else** **if** (removeBook == book5)

                    {

                        book5 = "";

                    }

**else**

                    {

                        Console.WriteLine("Book not found.");

                    }

                }

            }

**else** **if** (action == "exit")

            {

**break**;

            }

**else**

            {

                Console.WriteLine("Invalid action. Please type 'add', 'remove', or 'exit'.");

            }

            // Display the list of books

            Console.WriteLine("Available books:");

**if** (!**string**.IsNullOrEmpty(book1)) Console.WriteLine(book1);

**if** (!**string**.IsNullOrEmpty(book2)) Console.WriteLine(book2);

**if** (!**string**.IsNullOrEmpty(book3)) Console.WriteLine(book3);

**if** (!**string**.IsNullOrEmpty(book4)) Console.WriteLine(book4);

**if** (!**string**.IsNullOrEmpty(book5)) Console.WriteLine(book5);

        }

    }

}

**Steps to Improve Code Quality:**

1. **Run the Program to Test Functionality**
   * Use the Visual Studio Code console application you created at the start of the course. Remove any existing code in the Program.cs file of your console application and run the code Starting Code to Input into Copilot in that file.
   * Verify the functionality of the application.
2. **Use Copilot to Suggest Improvements**
   * Use Microsoft Copilot to suggest ways to reduce code repetition and improve readability.
   * For example, Copilot might suggest using a method to handle repetitive tasks like adding or removing books.
3. **Simplify the Code**
   * Simplify the program using the suggestions provided by Copilot. This could include creating helper methods, improving variable names, and adding comments to clarify the code’s functionality.
4. **Test the Simplified Program**
   * Use the Visual Studio Code console application you created at the start of the course. Remove any existing code in the Program.cs file of your console application and add all the updated code in that file.
   * Run the program after refactoring to ensure that it functions the same way, but with cleaner and more efficient code.

When completed, save your code. You will use this code to complete the final project in this course.

**Code:**

**using** **System.Globalization**;

**class** **Book**(**string** title)

{

**public** **string** Title { **get**; } = title;

**public** **override** **string** **ToString**() => Title;

}

**enum** Command

{

Add,

Remove,

Exit,

Invalid

}

**class** **LibraryManager**

{

**private** **static** **int** MaxBooks;

**private** **static** List<Book?>? books;

**static** **void** **Main**()

{

MaxBooks = GetLibrarySize();

books = [.. **new** Book?[MaxBooks]];

**while** (**true**)

{

PrintMenu();

**string?** actionInput = Console.ReadLine()?.Trim().ToLowerInvariant();

**switch** (ParseCommand(actionInput))

{

**case** Command.Add:

AddBook();

**break**;

**case** Command.Remove:

RemoveBook();

**break**;

**case** Command.Exit:

Console.WriteLine("Exiting Library Manager...");

**return**;

**default**:

Console.WriteLine("Invalid input. Please enter 'add', 'remove', or 'exit'.");

**break**;

}

DisplayBooks();

}

}

**private** **static** **int** **GetLibrarySize**()

{

Console.Write("Enter the maximum number of books in the library: ");

**if** (**int**.TryParse(Console.ReadLine(), **out** **int** size) && size > **0**)

**return** size;

Console.WriteLine("Invalid number. Defaulting to 5.");

**return** **5**;

}

**private** **static** Command **ParseCommand**(**string?** input) => input **switch**

{

"add" => Command.Add,

"remove" => Command.Remove,

"exit" => Command.Exit,

\_ => Command.Invalid

};

**private** **static** **void** **PrintMenu**()

{

Console.WriteLine("\nLibrary Manager");

Console.WriteLine($"Books stored: {(books ?? []).Count(b => b != null)} / {MaxBooks}");

Console.Write("Choose an action (add / remove / exit): ");

}

**private** **static** **void** **AddBook**()

{

**if** (books == **null** || !books.Any(b => b == **null**))

{

Console.WriteLine("The library is full. Cannot add more books.");

**return**;

}

**string?** title = GetValidatedInput("Enter the book title to add: ");

**if** (title == **null**) **return**;

**if** (BookExists(title))

{

Console.WriteLine("This book already exists in the library.");

**return**;

}

**int** index = books.FindIndex(b => b == **null**);

books[index] = **new** Book(title);

Console.WriteLine($"Book \"{title}\" added to the library.");

}

**private** **static** **void** **RemoveBook**()

{

**if** (books == **null** || books.All(b => b == **null**))

{

Console.WriteLine("The library is empty. No books to remove.");

**return**;

}

**string?** title = GetValidatedInput("Enter the book title to remove: ");

**if** (title == **null**) **return**;

**int** index = books.FindIndex(b => b != **null** && b!.Title.Equals(title, StringComparison.OrdinalIgnoreCase));

**if** (index >= **0**)

{

books[index] = **null**;

Console.WriteLine($"Book \"{title}\" removed from the library.");

}

**else**

{

Console.WriteLine("Book not found.");

}

}

**private** **static** **void** **DisplayBooks**()

{

Console.WriteLine("\nCurrent books in the library:");

**var** nonEmptyBooks = (books ?? []).Where(b => b != **null**).Select(b => b!.Title).ToList();

**if** (nonEmptyBooks.Count == **0**)

{

Console.WriteLine("No books available.");

}

**else**

{

Console.WriteLine(**string**.Join("\n", nonEmptyBooks.Select(b => "- " + b)));

}

}

**private** **static** **string?** GetValidatedInput(**string** prompt)

{

Console.Write(prompt);

**string?** input = Console.ReadLine()?.Trim();

**if** (**string**.IsNullOrWhiteSpace(input))

{

Console.WriteLine("Book title cannot be empty.");

**return** **null**;

}

**return** **NormalizeTitle**(input);

}

**private** **static** **string** **NormalizeTitle**(**string** title)

{

**return** CultureInfo.CurrentCulture.TextInfo.ToTitleCase(title.ToLowerInvariant());

}

**private** **static** **bool** **BookExists**(**string** title)

{

**return** books != **null** && books.Any(b => b != **null** && b.Title.Equals(title, StringComparison.OrdinalIgnoreCase));

}

}