**Practical coding exercises**

**Objective:**

You will use Microsoft Copilot to extend the Library Management System by solving new challenges, such as searching for a book or limiting the number of books that can be borrowed. You should use the code you’ve already optimized in the previous “You Try It!” activity to complete this activity. At the end of this course, you’ll submit this code as your final project.

**Problem Statement:**

The Library Management System works well, but it can be extended to solve new problems. You need to add the following features:

* Search for a book by its title.
* Limit the number of books a user can borrow at once.

With Microsoft Copilot, you will solve these new challenges and extend the functionality of the system.

**Steps**

1. Add a Search Feature
   1. Prompt the user to input a book title to search for.
   2. If the book is found, display a message indicating it is available.
   3. If the book is not found, display a message that it’s not in the collection.
2. Limit Borrowing
   1. Add a feature that tracks how many books a user has borrowed.
   2. Limit the number of books to 3 at a time.
3. Check-in a borrowed book
   1. Add a feature that enables the user to check in a book that’s been checked out.
   2. Check If the book is checked out. If it is remove the checked-out flag to check the book in. If it isn’t, inform the user.
4. Test the New Features
   1. Run the program and test the search and borrowing limit functionalities with different inputs.

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** **bool** IsBorrowed { **get**; **set**; } = **false**;

**public** **override** **string** **ToString**() => $"{Title}" + (IsBorrowed ? " [Borrowed]" : "");

}

**enum** Command

{

Add,

Remove,

Search,

Borrow,

CheckIn,

Exit,

Invalid

}

**class** **LibraryManager**

{

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

**private** **const** **int** BorrowLimit = **3**;

**private** **static** **int** BorrowedCount = **0**;

**private** **static** List<Book?> books = **default**!;

**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.Search:

SearchBook();

**break**;

**case** Command.Borrow:

BorrowBook();

**break**;

**case** Command.CheckIn:

CheckInBook();

**break**;

**case** Command.Exit:

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

**return**;

**default**:

Console.WriteLine("Invalid input. Please enter a valid command.");

**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,

"search" => Command.Search,

"borrow" => Command.Borrow,

"checkin" => Command.CheckIn,

"exit" => Command.Exit,

\_ => Command.Invalid

};

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

{

Console.WriteLine("\nLibrary Manager");

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

Console.WriteLine($"Books borrowed: {BorrowedCount} / {BorrowLimit}");

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

}

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

{

**if** (!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.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**)

{

**if** (books[index]!.IsBorrowed)

{

Console.WriteLine("Cannot remove a book that is currently borrowed.");

**return**;

}

books[index] = **null**;

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

}

**else**

{

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

}

}

**private** **static** **void** **SearchBook**()

{

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

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

**var** book = books.FirstOrDefault(b => b != **null** && b.Title.Equals(title, StringComparison.OrdinalIgnoreCase));

**if** (book != **null**)

{

Console.WriteLine($"Book found: {book}");

}

**else**

{

Console.WriteLine("Book not found in the library.");

}

}

**private** **static** **void** **BorrowBook**()

{

**if** (BorrowedCount >= BorrowLimit)

{

Console.WriteLine($"You cannot borrow more than {BorrowLimit} books at a time.");

**return**;

}

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

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

**var** book = books.FirstOrDefault(b => b != **null** && b.Title.Equals(title, StringComparison.OrdinalIgnoreCase));

**if** (book == **null**)

{

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

}

**else** **if** (book.IsBorrowed)

{

Console.WriteLine("That book is already borrowed.");

}

**else**

{

book.IsBorrowed = **true**;

BorrowedCount++;

Console.WriteLine($"You have borrowed \"{book.Title}\".");

}

}

**private** **static** **void** **CheckInBook**()

{

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

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

**var** book = books.FirstOrDefault(b => b != **null** && b.Title.Equals(title, StringComparison.OrdinalIgnoreCase));

**if** (book == **null**)

{

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

}

**else** **if** (!book.IsBorrowed)

{

Console.WriteLine("That book is not currently borrowed.");

}

**else**

{

book.IsBorrowed = **false**;

BorrowedCount--;

Console.WriteLine($"You have checked in \"{book.Title}\".");

}

}

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

{

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

**var** nonEmptyBooks = books.Where(b => b != **null**).Select(b => b!.ToString()).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.Any(b => b != **null** && b.Title.Equals(title, StringComparison.OrdinalIgnoreCase));

}

}