CMPE 261: Large Scale Programming C++ Project: Digital Library Management System

Name: Emre Şentürk

Student id: 123200133

Date: 30.11.2024

1. Class: **Library Book**

   This class represents individual books in the library system. Each instance of the LibraryBook class shows the details and status of a book.

   Attributes:

   - **Title:** The name of the book.
   - **Author:** The author of the book.
   - **ISBN:** International standard book number, a unique identifier for each book.
   - **Available:** A Boolean value for availability of the book.
   - **TotalBooks:** A static member that keeps track of the total number of books in the library.

   Functions:

   - **Constructor:** Initializes a book's details and increments the static.
   - **displayInfo():** Prints detailed information about the book, including its title, author, ISBN, and availability.
   - **setAvailability(bool):** Updates the availability status of the book. This function is used when a book is borrowed or returned.
   - **getTitle():** Getter funciton for title of the book.
   - **getAuthor():** Getter funciton for author of the book.
   - **getISBN():** Getter funciton for ISBN of the book.
   - **isAvailable():**Returns the current availability status of the book.
   - **getTotalBooks():**A static method that provides the total count of books in the system.

2. Class: **LibraryUser**

This class serves as a blueprint for all types of library users, such as members and librarians.

Attributes:

- **Name:** The name of the user.
- **userID:** A unique identifier for the user.

Funcitons:

- **Constructor:** Initializes the user's name and ID.
- **DisplayUserInfo():** A virtual function intended to be overridden by derived classes. It displays basic user details.

3. Cass: **Member**

Derived from LibraryUser, this class represents a library member.

Attributes:

- **borrowedBooks:** A vector that stores pointers to the books borrowed by the member.

Functions:

- **Constructor:** Inherits form the LibraryUser class and initializes member attributes.

- **borrowBook(LibraryBook*):** Allows the member to borrow a book if it is available. The book's availability status is updated, and the book is added to the borrowedBooks list.
- **returnBook(LibraryBook*):** Allows the member to return a book. If the book is found in the borrowedBooks list, it is removed, and its availability status is updated.
- **displayUserInfo():** Overrides the base class method to display member details.

4. Class: **Librarian**

Derived from LibraryUser , this class represents a librarian responsible for managing the library's book collection.

Attributes:

- **Library**: A reference to the library's collection of books.

Funcitons:

- **Constructor:** Initializes the librarian's details and sets up a reference to the library's book collection.
- **addBook(LibraryBook*):** Allows the librarian to add a new book to the library.
- **removeBook(string):** Removes a book from the library by its ISBN. If the book is found, it is deleted from memory and removed from the collection.

- **displayUserInfo():**Overrides the base class method to include librarian-specific details.

5. Class: **LibrarySystem**

This class represents the overall library system, managing books and users.

Attributes:

- **books:** A vector of pointers to all books in the library.
- **users:** A vector of pointers to all users in the system.

Functions:

- **Destructor**: Ensures proper cleanup of dynamically allocated memory for books and users.
- **addUser(LibraryUser*)**: Adds a new user (member or librarian) to the system.
- **addBook(LibraryBook*)**: Adds a new book to the system.
- **searchBook(string)**: Searches for a book by title using a recursive helper function (searchBookHelper).
- **displayAllBooks()**: Displays all books in the library, along with their details and total count.
- **displayAllUsers()**: Displays all users in the system, categorized by type (member or librarian).

Design Decisions:

Encapsulation:

- Encapsulating book and user data ensures that all interactions happen through well-defined interfaces.

Inheritance:

- The LibraryUser class is a base class, promoting code reuse and extensibility for Member and Librarian.

Dynamic Memory Management:

- All books and users are stored as pointers to allow dynamic allocation and proper cleanup when the program ends.

Recursion:

- A recursive approach is used for book searching, which is elegant and concise for this purpose.

Vectors:

- Used for managing collections (books and users) due to their dynamic size and efficient indexing.

Challenges and Solutions

Experience and Reflections on Using Vectors for the First Time:

In this project, i used vector to manage and store books and users in the library system. Since i had limited prior experience with dynamic data structures like vector, i initially found it challenging to adapt to this approach. However, as the project progressed, i became more comfortable with using vectors and appreciated the advantages they offer.

Understanding Dynamic Behavior:

At first, i was not fully familiar with how vectors dynamically resize themselves as elements are added or removed. Managing the vector's size and ensuring that operations like adding and removing elements did not cause unintended issues required extra attention.

Pointer Management in Vectors:

Since the project involved storing pointers in vectors, it was necessary to carefully handle memory management. Forgetting to delete dynamically allocated objects stored in the vector, or inadvertently deleting the same object twice, could lead to memory leaks or crashes.

Iterating Safely:

When modifying vectors, such as during book removal, ı had to ensure that iterators remained valid and did not cause runtime errors. This required learning about safe iterator handling, particularly when erasing elements.

To address these challenges, I explored various online resources, including forums, websites, and YouTube tutorials, where others had encountered similar issues. These platforms provided valuable insights and practical examples, which helped me better understand how to work with vectors and manage common problems such as memory leaks, invalid iterators, and dynamic resizing. By learning from others experiences and solutions, I was able to effectively apply best practices and improve my implementation.

Here is a sample output for library project:

```
Library System Initialized.
User Emre added to the library system.
User john Doe added to the library system.
User Sarah Smith added to the library system.
Total books: 1
Librarian Emre added the book 1984
Book 1984 added to the library system.
Librarian Emre added the book To Kill a Mockingbird
Book To Kill a Mockingbird added to the library system.
Total books: 2
john Doe borrowed 1984
john Doe returned 1984
Book 1984 found.
Displaying all books:
All the books in library:
Title of the book is 1984.
Author of the book is George Orwell.
International Standard Book Number is 9780451524935.
The book is available.

Title of the book is To Kill a Mockingbird.
Author of the book is Harper Lee.
International Standard Book Number is 9780060935467.
The book is available.

Total books in the library is 2
Displaying all users:
All the users in the system:
Librarian: Emre, ID: 1

Member: john Doe, ID: 101

Member: Sarah Smith, ID: 102
```