Exercise 4: Redesigned Library Management System

Related Topics

Exception, ADT, Inheritance

Problem Overview

Recall the library management system from **Exercise 2**, which is implemented through structs and functions. In this exercise you will redesign the library management system using classes and objects, transitioning to an object-oriented approach. This exercise will deepen your understanding of the concept of classes and object-oriented programming.

Background

Instead of merely implementing the library class, you will first declare a base class `bookInventory` and derive the class `library` from it. The base class will contain the common attributes and methods of the derived class. The derived class will inherit the base class and add its own attributes and methods. You have the following objects predefined in `ex4.h`:

- `MAX_BOOKS` - Maximum number of books that can be stored in the library, set to 10

```
const int MAX_BOOKS = 10;
```

- `Book` struct along with its constructors

```
struct Book {
  std::string title;
  std::string author;
  bool isAvailable;
  Book() : title(""), author(""), isAvailable(true) {}
  Book(const std::string &title, const std::string &author) : title(title),
  author(author), isAvailable(true) {}
};
```

- `Exception` class for error messages

```
class Exception {

  std::string message = "";

public:
  Exception(const std::string &message) : message(message) {}
  std::string what() const { return message; }
};
```

Note that in this exercise, you should throw an `Exception` object when an error occurs along with the corresponding error message.

Your Task

Notes on the Book Struct

- Unlike in Exercise 2, the book's `ID` is not defined explicitly in each `Book` object. Instead, the `ID` is the array index of the book in the inventory plus 1, which may be dynamic.
- We declared two constructors for the `Book` struct. The default constructor returns an empty `Book` object, while the second constructor initializes the `Book` object with the given title and author.

Part 1: bookInventory Class

- **Implement the bookInventory Class**: This class manages a collection of `Book` objects. The following methods are required:
  - Default constructor: Initializes the inventory by filling it with empty `Book` objects and set `numBooks` to 0.
  - `addBook(const Book &book)`: Adds a new `Book` to the collection. If the inventory reaches its limit (`MAX_BOOKS`), this function throws an `Exception` with the message `"The inventory is full."`.
  - `searchBook(const std::string &title) const`: Searches for the book by title and

returns its ID. If the book is not found, it throws an `Exception` with the message `"Book [title] not found."`

- Sample Output: `Book 1984 not found.`

- `setBook(const Book &book, int ID)`: Updates the details of a book at a specific ID. If the ID is less than `MAX_BOOKS` but not occupied, it adds the book to the inventory's next empty spot. If the ID is invalid, it throws an `Exception` with the message `"Invalid book ID."`
- `viewBook(int ID) const`: Returns the Book at a specific ID. If the ID is invalid, it throws an `Exception` with the message `"Invalid book ID."`
- `removeBook(int ID)`: Removes a book from the inventory based on its ID. It rearranges the remaining books to fill the gap by shifting them to the left until the last book. If the ID is invalid, it throws an `Exception` with the message `"Invalid book ID."`
- `printInventory() const`: Prints the details of all books in the inventory. If the inventory is empty, it throws an `Exception` with the message `"The inventory is empty."`

  - Sample Output: The format is the same as in Exercise 2.

```
Book ID: 1
Title: The Catcher in the Rye
Author: J.D. Salinger
Status: available
```

Part 2: Library Class

- **Extend bookInventory with Library Class**: The `library` class inherits from `bookInventory` and adds specific functionalities:
  - Default constructor: The same as the default constructor of `bookInventory`.
  - `borrowBook(int ID)`: Marks a book as borrowed by setting `isAvailable` to `false`. Throws an `Exception` with the message `"Invalid book ID."` for invalid IDs, and `"Book [title] is not available."` if the book is already

borrowed.

- `returnBook(int ID)`: Marks a book as returned by setting `isAvailable` to `true`. Throws an `Exception` with the message `"Invalid book ID."` for invalid IDs, and `"Book [title] is already available."` if the book is not currently borrowed.
- `listBorrowed() const`: Lists all books that are currently borrowed. If no books are borrowed, it throws an `Exception` with the message `"All books are available."`
  - Sample Output: Here `Status` is omitted.

```
Book ID: 1
Title: The Catcher in the Rye
Author: J.D. Salinger
Book ID: 2
Title: 1984
Author: George Orwell
```

Implementation Details

- The declarations of the classes and methods are provided for you in `ex4.h`. You should implement the methods in `ex4.cpp`. You are welcome to write your own `main` function for testing, but only `ex4.cpp` should be submitted.
- You should not modify the provided `ex4.h` file.
- You should throw an `Exception` object when an error occurs along with the corresponding error message.
- The `ID` of a book is dynamic due to the `removeBook` method. For some methods, **if the book is not found, the `ID` is treated as invalid.**
- Specifically, for the `setBook` method, an `ID` is invalid outside the range of the array index instead of empty slots.
- For books added by `setBook`, if the `ID` is not occupied, the book is added to the inventory and the `ID` is set to the next available index, which is not necessarily the same as the `ID` passed to the method.
- **Pay attention to any typo or formatting errors.** They will be treated as failed test cases.

Submission

Compress your `ex4.cpp` into a zip file and submit it to JOJ. **The due date is March 31st**.