Exercise 5: Invariant Checker for BookInventory

Related Topics

Invariants, Dynamic Memory Allocation, Destructor

Problem Overview

This exercise is a continuation of Exercise 4. In this exercise, some invariants are added to the `bookInventory` class and some implementation details are changed. You need to update the `bookInventory` class to maintain the invariants. Ultimately, you need to implement a function `repOK()` to check the invariants of the `bookInventory` class.

Modifications from Exercise 4

- The `library` class is removed. The `bookInventory` class is directly responsible for managing the books.
- The `isAvailable` member is removed from the `book` struct, since the `library` class is removed.

```
struct Book {
  std::string title;
  std::string author;

  // Default constructor
  Book() : title(""), author("") {}
  // Constructor with parameters
  Book(const std::string &title, const std::string &author) :
    title(title), author(author) {}
};
```

- Member modifications in the `bookInventory` class:
  - The fixed `books` array is changed to a dynamic `Book` array.
  - A boolean `empty` to indicate whether the inventory is empty.
  - An unsigned integer `size` to store the size of the `books` array.

- Some member functions are deleted since they are not needed to maintain the invariants. You may check the new declarations in `ex5.h`.

Invariant Rules

The invariants should obey the following **but not limited to**:

- The inventory is empty if and only if all the books are default empty books.
- The books should be tightly packed in the array. That is, there should be no empty slots between books.
- All books after the last book should be default empty books.

Your Task

The following methods are required to be implemented in `ex5.cpp`:

- **Default Constructor**

```
/**
 * TODO: Implement the default constructor.
 * @brief Default constructor. Initializes the number of books to 0
 *        and fills the books with default values.
 *        The size of the books array is MAX_BOOKS.
 */
bookInventory();
```

- **Overloaded Constructor**

```
/**
 * TODO: Implement the constructor with parameter.
 * @brief Constructor with parameter. Initializes the number of books to 0
 *        and fills the books with default values.
 *        The size of the books array is maxBooks.
 *
 * @param maxBooks The size of the books array.
 * @throw Exception if maxBooks is less than 1 or greater than MAX_BOOKS.
 *        The exception message should be "Invalid size.".
 */
```

```
bookInventory(int maxBooks);
```

- **Destructor**

```
/**
 * TODO: Implement the destructor.
 * @brief Destructor for the bookInventory class.
 */
~bookInventory();
```

- **`addBook()`**

```
/**
 * @brief Adds a book to the inventory.
 *
 * @param book The book to be added.
 * @throw Exception if the inventory is full.
 *        The exception message should be "The inventory is full.".
 */
void addBook(const Book &book);
```

- **`removeBook()`**

```
/**
 * @brief Removes a book from the inventory.
 *
 * @param ID The ID of the book to be removed.
 *           The ID is the index of the book in the books array + 1.
 * @throw Exception if the ID is invalid.
 *        The exception message should be "Invalid book ID.".
 */
void removeBook(int ID);
```

- **`printInventory()`**

```
/**
 * @brief Prints the inventory of books.
 *
```

```
 * @throw Exception if the inventory is empty.
 *        The exception message should be "The inventory is empty.".
 */
void printInventory() const;
```

- **`repOK()`**

```
/**
 * TODO: Implement the invariant checker.
 * @brief Checks if the invariants are true.
 *
 * @return True if the invariants are true, false otherwise.
 */
bool repOK();
```

Implementation Details

- The declarations of the classes and methods are provided in `ex5.h`. You should not modify them.
- You may write your own `main` function to test locally, but only `ex5.cpp` should be submitted.
- Only methods with `TODO` comments or listed above need to be implemented.
- For the `printInventory()` method, note that since the `isAvailable` member is removed from the `book` struct, you only need to print the title and author of the book.

```
Book ID: 1
Title: Title1
Author: Author1
Book ID: 2
Title: Title2
Author: Author2
```

- You can write your implementation based on what you have done in Exercise 4, but you need to make some modifications to deal with the new invariants.

- We won't deliberately test the exception handling for methods from Exercise 4, but you should still implement them correctly.
- You may notice some member functions in `bookInventory` that modify some private members by force. Your `repOK()` function should be able to detect these violations.
- **Consider more rules for the invariants beyond the explicit rules given above.** They might be tested in the hidden test cases. But don't worry, they won't be too tricky.
- Any typo or format errors will be considered as failed test cases.
- Since the memory management is simple in this exercise, any cases that fail due to memory issues will be considered as failed test cases.

Submission

Compress your `ex5.cpp` and submit it to JOJ. **The due date is December 5, 23:59.**