Exercise 5: Invariant Checker for BookInventory

Related Topics

Exception,Invariants, Dynamic Memory Allocation, Destructor

Problem Overview

Recall that in exercise 2, we implement the `Book` struct and `Library` struct. In exercise 5, there are some changes on `Book` and `Libiary` .

In `Book` struct, `ID` and `isAvailable` is removed, while two constructor are provided.

```
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) {}
};
```

In Exercise 5, we change the `Library` to the `bookInventory` , which is a class supporting a few functions(Require you to implement).

```
class bookInventory {
protected:
  Book *books;
  unsigned int numBooks;
  bool empty;
  unsigned int size;

public:
  bookInventory();
  explicit bookInventory(int maxBooks);
  int getNumBooks() const { return (int)this->numBooks;}
  bool isEmpty() const {return this->empty;}
  unsigned int getSize() const {return this->size;}
  void setNumBooks(unsigned int numBooks) {this->numBooks = numBooks;}
  void setEmpty(bool empty) {this->empty = empty;}
  void setBook(const Book &book, int ID) {books[ID - 1] = book;}
  bool repOK();
  ~bookInventory();
  void addBook(const Book &book);
  void removeBook(int ID);
  void printInventory() const;
};
```

Different from `Library`, `bookInventory` has following features.

- 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 dynamic array.

You need to maintain some invariants in the `bookInventory` class. Ultimately, you need to implement a function `repOK()` to check the invariants of the `bookInventory` class.

Check the detailed 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.

Exception

You should throw an `Exception` object when an error occurs along with the corresponding error message. Check the detailed comment with prefix `@throw` in `ex5.h` of each function.

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.
*        and fills the books with default values.
*        The size of the books array is maxBooks.
*
* @brief Constructor with parameter. Initializes the number of books to 0
* @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);
bookInventory();
```

- 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;
```

Example printing format:

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

- 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 for local testing, but **only `ex5.cpp` should be submitted**.
- Only methods listed above need to be implemented.
- For `printInventory()`, note that `isAvailable` has been removed. Only print **title** and **author**.

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

- We won't deliberately test the exception handling, but you should still implement them correctly.
- 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.
- Since the memory management is simple in this exercise, any cases that fail due to memory issues will be considered as failed test cases.
- **Pay attention to any typo or formatting errors.** They will be treated as failed test cases.

Submission

Save file as `ex5.cpp` and submit it to JOJ. **The due time is 11.29 23:59**.