*Last Updated: 3/8/2021 2:17 PM*

# Assignment: Managing Books, Part 2

*CSE 1384 Intermediate Computer Programming W/C++*

**Assignment Objectives**

- To practice writing object-oriented programs in C++.
- To practice writing functionalized programs in C++.
- To practice writing arrays in C++.
- To apply problem solving and critical thinking skills to produce a code solution that meets assignment requirements.
- To practice testing a complete program to ensure its correctness.

**Tools Used**

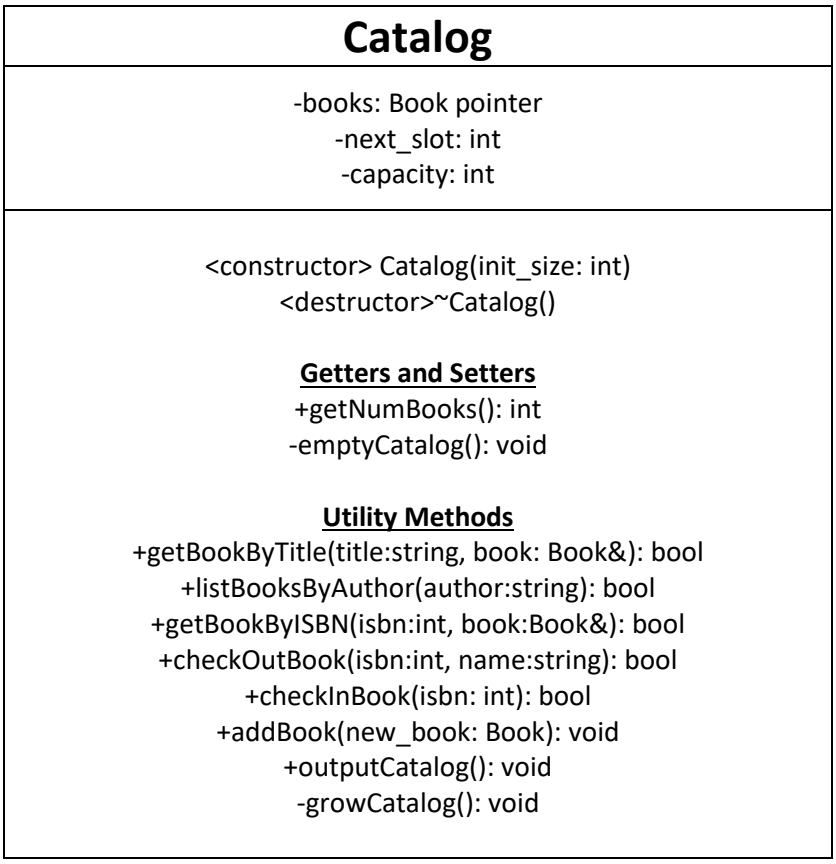
1. Installation of Visual Studio Link: https://visualstudio.microsoft.com/downloads/
2. Access to Screenshotting Software (if screenshots are required to complete assignment; Windows Snipping Tool Recommended)
3. Word Processing Document to Create Analysis Report (if analysis questions are asked; Microsoft Office Strongly Recommended)

## Part 1 Instructions

In the previous assignment, we created a class called "Book", which stored basic information about a book at a library. The program used this class to create sample book objects to store information about popular novels.

In this program, we are going to build a new class called "Catalog", which will be used to store and manage books in the library. The program should support the following menu commands:

1. Search Catalog
    a. By Title
    b. By ISBN
2. List Books by Author
3. Add New Book
4. Get Book Status
5. Check in a Book
6. Check Out a Book
7. Exit




### Descriptions of Notable Methods

1. **listBookByAuthor** – prints out a list of books written by the author. Returns true if the author has at least 1 book in the catalog and false if not.
2. **getBookBy<title, ISBN>** - These methods search the Catalog for a book matching the specified attribute. The "book" parameter is a reference parameter used to pass the book back by. If the book is found, assign the book parameter to the book found and return true. If not found, do not assign the book object to anything, and return false.
3. **checkOutBook –** Searches the catalog of the book for the ISBN number and checks the book out, if the book is found and is not checked out. Returns true if it successfully checks the book out, false if it does not.
4. **checkInBook –** Checks in a book in the catalog. Returns true if the book is found and checked in, and false otherwise.
5. **addBook** – Adds a book to the catalog. Takes in a book object and then copies that book object into the next available slot. If the "books" array is at capacity, then growCatalog() should be called.
6. **outputCatalog** – Prints out the entire catalog of books. Calls the outputBook method from each book.

7. **growCatalog** – This function will be provided to you in the Appendix. This function should be copied into your Catalog class definition. Used to increase the number of elements in the "books" array to support more books. ***Check Appendix A for the solution to this function.***
8. **emptyCatalog** – This function deallocates all book objects in the "books" array. Use delete[] on the array to achieve this.

## Hints

1. Start by creating the class definition first. Implement the methods in the class definition before working on the .cpp file.
2. Test each method individually in the main function before writing the code for the menu. Create a Catalog object to work with and add random books to the catalog. Call the methods and observe how they behave to test the Catalog class.
    - Tip: You may add additional helper methods to your Catalog. One method you might add is the ISBN search method from Program 3 to determine the index location of a book in the array. This might be useful for check in and check out methods.
3. Once you feel confident the Catalog class works, begin working on the menu code. You may use either an if/else if statement or switch statement to decide what menu item is chosen.
4. Test each menu command to ensure they work.

| Rubric | |
|---|---|
| **Total: 100 pts** | |
| Overall Implementation of CatalogClass.h | 60 pts |
| → Correctly Defined Catalog Attributes (2 total) | 6 pts |
| → Correctly Defined Book Getters and Setters (2 total) | 6 pts |
| →Correctly Defined Utility Methods (8 total)<br>Note: remember that growCatalog is already defined and not graded. | 48 pts |
| Overall Implementation of TestCatalogClass.cpp | 40 pts |
| →Correct Display of Menu Commands | 3 pts |
| →Correct Selection of Menu Item | 3 pts |
| →Correct implementation of Book Search Commands | 8 pts |
| →Correct implementation of Add New Book Command | 9 pts |
| →Correct implementation of Check Out Book Command | 9 pts |
| →Correct implementation of Quit Program | 3 pts |
| →Support for appropriate error checking and error output in program | 5 pts |

| Penalties | |
|---|---|
| Late | Late penalties apply as per syllabus. |
| Assignment Formatting<br>This includes:<br>Header Comment Inclusion<br>Comments in Code<br>Following Analysis Report Template when Required<br>Overall cleanliness of solution | Up to -20 pts (depending on the issue) |
| Syntactical Errors | -100 pts (automatic zero) |

## Header Comment

**/***

**Name:** ***<your name>***

**NetID:** ***<your NetID>***

**Program Description:** ***<enter your program description here>***

***/**

## Submission Instructions

This assignment is due by the due date specified in Canvas. You are expected to submit the following files:

1. *netid*_prog4_CatalogClass.h - File containing your solution in h format.
2. *netid*_prog4_TestCatalogClass.cpp - File containing your solution in CPP format.

## Appendix A

***Grow Method for the Catalog Class***

```
void grow_catalog()
{
        Book* temp = books;

        capacity *= 2;
        books = new Book[capacity];

        for (int i = 0; i < next_slot; i++)
                books[i] = temp[i];

        delete[] temp;
        temp = nullptr;
}
```