

Object Oriented Programing Project Report



Photo by [Alfons Morales](#) on [Unsplash](#)

Library Management System

**Prepared by
Joseph Oladunmoye
at
Seneca college**

June 2024

Table of Contents

I	Conceptual Questions	3
1	Virtual functions and polymorphism.....	3
2	Virtual Destructor	3
3	UML Class Diagram.....	4
II	Output Screenshots	5
1	Main Navigation Menu.....	5
2	Main Menu – option 1 (View catalog sorted by title)	6
3	Main Menu – option 2 (View catalog sorted by genre).....	7
4	Main Menu – option 3 (View catalog sorted by publication type).....	8
5	Main Menu – option 4 (Other view options).....	8
6	Main Menu – option 5 (Search for an item)	10
7	Main Menu – option 6 (Add items to the catalog)	11
8	Main Menu – option 7 (Remove an item)	13
9	Main Menu – option 8 (Exit).....	15
III	Complete Code	16
1	LibraryItem.h.....	16
2	LibraryItem.cpp	17
3	Book.h	17
4	Book.cpp.....	18
5	Magazine.h	18
6	Magazine.cpp.....	19
7	Catalog.h.....	19
8	Main.cpp.....	25

I Conceptual Questions

1 Virtual functions and polymorphism

Explain the role of virtual functions and polymorphism in the Library Management System. Provide a detailed description of how virtual functions enable polymorphism and give an example from the provided code.

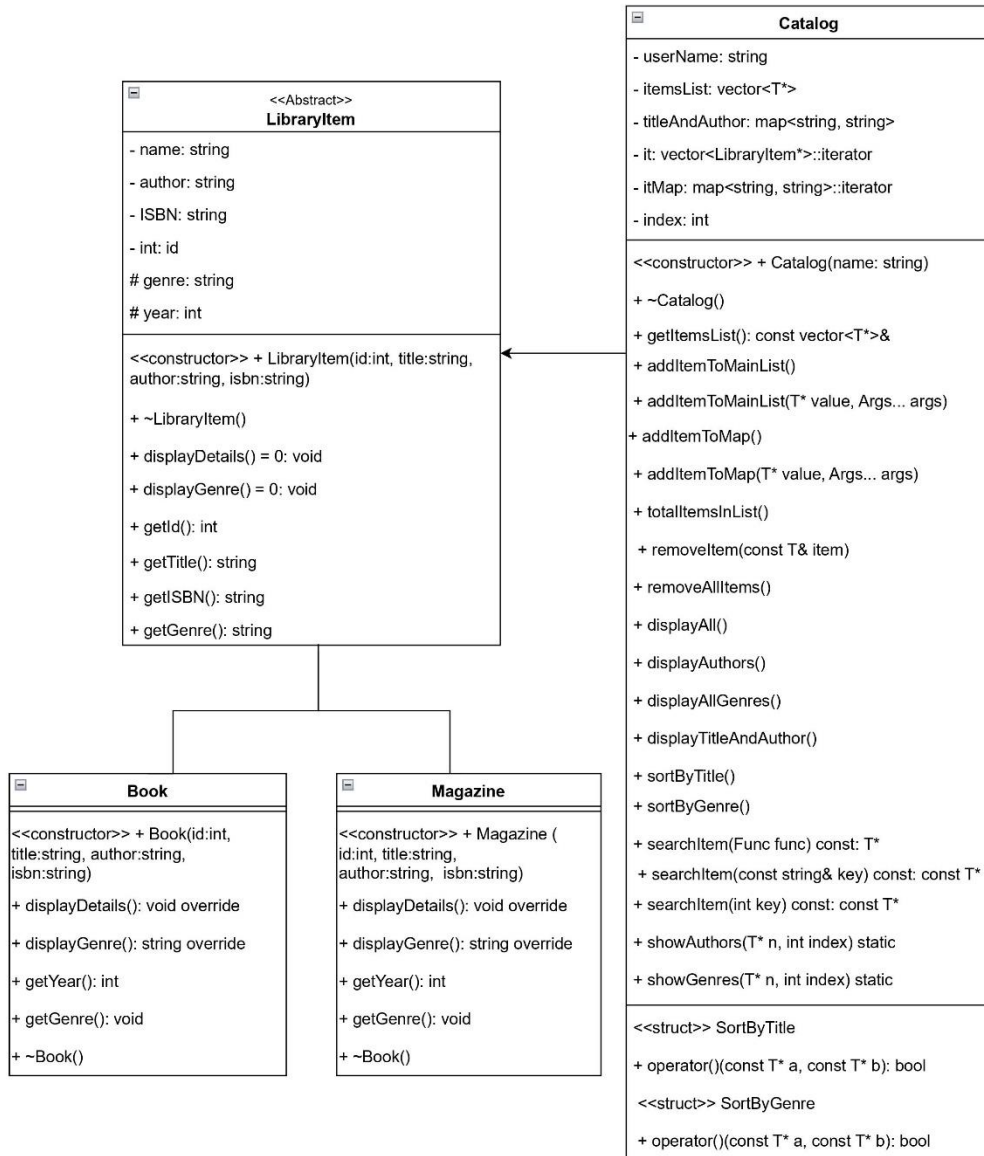
- Virtual functions enforce implementation by all derived classes. In this project the display function is set as a pure virtual function in the Base class (LibraryItem). In the implementation in the child classes (Book and Magazine), each child class can override the display function as required by its business class.
- In the Catalog class, a Base class pointer to vector created. This vector can point to any child class. In so doing, it becomes possible for the program to choose the correct derived class for each object at execution (dynamic binding).
- This behavior where a single function can be used for different object types is a type of polymorphic behavior.

2 Virtual Destructor

Additionally, discuss the importance of having a virtual destructor in the base class. What would happen if the destructor was not declared as virtual?

- If Base class destructor is not declared virtual, only the base class destructor will be called. Although when a derived class is destroyed, the base class destructor is also called automatically. However, this can lead to undefined behavior and potential memory leakage. It is therefore important to synchronize the order in which destructors are called.
- When a virtual destructor is declared in the base class, destructors in the derived classes also become virtual destructors and overrides the base class destructor. This will ensure that the destructors for each specific object is called before the base class destructor.

3 UML Class Diagram

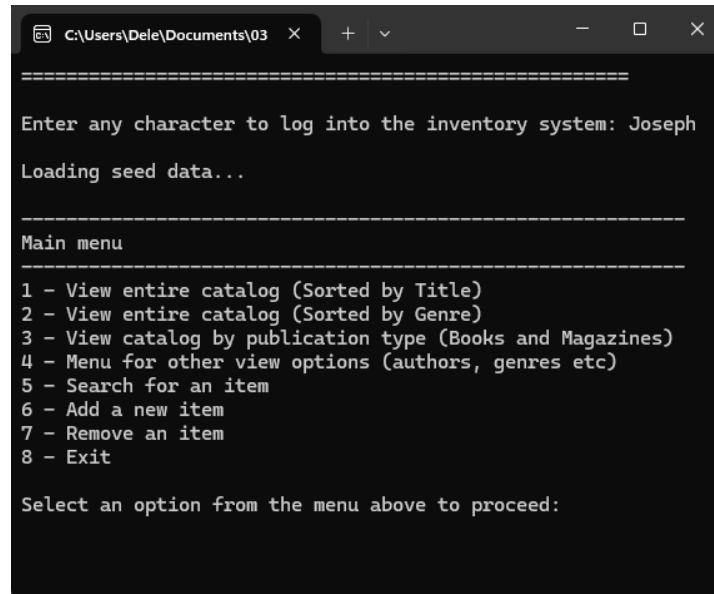


UML CLASS DIAGRAM
LIBRARY MANAGEMENT SYSTEM

II Output Screenshots

1 Main Navigation Menu

Main Navigation Menu



```
C:\Users\Dele\Documents\03 X + - □ X
=====
Enter any character to log into the inventory system: Joseph
Loading seed data...

-----
Main menu
-----
1 - View entire catalog (Sorted by Title)
2 - View entire catalog (Sorted by Genre)
3 - View catalog by publication type (Books and Magazines)
4 - Menu for other view options (authors, genres etc)
5 - Search for an item
6 - Add a new item
7 - Remove an item
8 - Exit

Select an option from the menu above to proceed:
```

Main navigation menu

- Welcome message is displayed
- Log in message is displayed. The username is used to create a Catalog object in main
- Seed data is loaded which is used to populate the current Catalog for easy testing. 'Loading seed data ...' is displayed.
- 8 main navigation options are available in the main menu. Programmatically, these options are structured in a while loop with if/else clauses as control to navigate the options.
- On selection of any of the menu options, results are displayed or a sub menu is presented as required.
- The main nav is further accessible from any of the sub menus once the sub menu function is complete.
- Option #8 of the main menu exits the user from the program.

2 Main Menu – option 1 (View catalog sorted by title)

Main Menu – Option 1 (View Entire Catalog sorted by title)

```
C:\Users\Dele\Documents\03 x + - □ x

Select an option from the menu above to proceed: 1

-----
Current menu selection:
Main menu - Option 1 (View entire collection sorted by title)
-----
Displaying entire collection sorted by title...

1. 1984
   by George Orwell - 1949
   ISBN: 9780140817744
   Genre: Dystopian

2. Sports Illustrated
   by ABG - 2024
   ISBN: 985123456789
   Genre: Sport

3. The Great Gatsby
   by F. Scott Fitzgerald - 1925
   ISBN: 123456789
   Genre: Fiction

4. To Kill a Mockingbird
   by Harper Lee - 1960
   ISBN: 9781846572562
   Genre: Classic

Total number of items in the catalog: 4

-----
Main menu
-----
1 - View entire catalog (Sorted by Title)
2 - View entire catalog (Sorted by Genre)
3 - View catalog by publication type (Books and Magazines)
4 - Menu for other view options (authors, genres etc)
5 - Search for an item
6 - Add a new item
7 - Remove an item
8 - Exit

Select an option from the menu above to proceed:
```

Main menu – option 1 (View Entire catalog sorted by title)

- Entire catalog is displayed sorted by title
- Main nav is displayed

3 Main Menu – option 2 (View catalog sorted by genre)

Main Menu – Option 2 (View Entire Catalog sorted by title)

```
C:\Users\Dele\Documents\03 x + - □ x

Select an option from the menu above to proceed: 2

-----
Current menu selection:
Main menu - Option 2 (View entire collection sorted by genre)
-----
Displaying entire collection sorted by genre...

1. To Kill a Mockingbird
   by Harper Lee - 1960
   ISBN: 9781846572562
   Genre: Classic

2. 1984
   by George Orwell - 1949
   ISBN: 9780140817744
   Genre: Dystopian

3. The Great Gatsby
   by F. Scott Fitzgerald - 1925
   ISBN: 123456789
   Genre: Fiction

4. Sports Illustrated
   by ABG - 2024
   ISBN: 985123456789
   Genre: Sport

Total number of items in the catalog: 4

-----
Main menu
-----
1 - View entire catalog (Sorted by Title)
2 - View entire catalog (Sorted by Genre)
3 - View catalog by publication type (Books and Magazines)
4 - Menu for other view options (authors, genres etc)
5 - Search for an item
6 - Add a new item
7 - Remove an item
8 - Exit

Select an option from the menu above to proceed:
```

Main menu – option 1 (View Entire catalog sorted by genre)

- Entire catalog is displayed sorted by genre
- Main nav is displayed

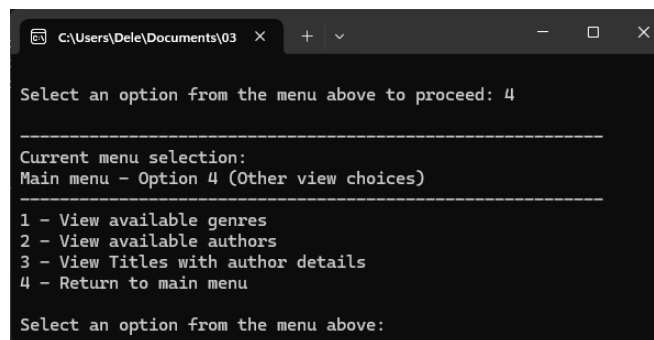
4 Main Menu – option 3 (View catalog sorted by publication type)

Main Menu – Option 3 (View Entire Catalog sorted by title)

In progress

5 Main Menu – option 4 (Other view options)

Main Menu – Option 4 (View Entire Catalog sorted by title)



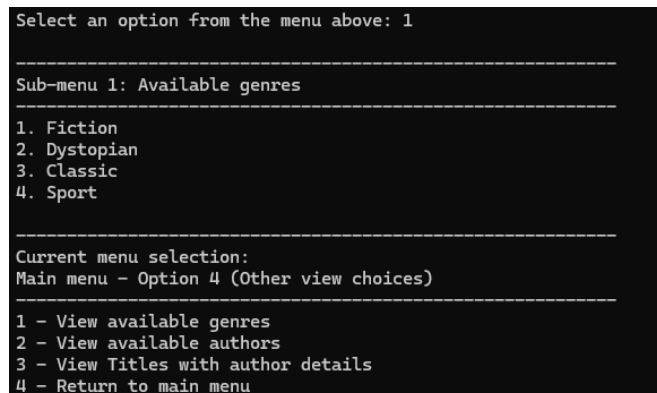
```
C:\Users\Dele\Documents\03 x + - □ x

Select an option from the menu above to proceed: 4

-----
Current menu selection:
Main menu - Option 4 (Other view choices)
-----
1 - View available genres
2 - View available authors
3 - View Titles with author details
4 - Return to main menu
Select an option from the menu above:
```

Main menu – option 4 (Other view options)

- This presents a sub menu of 4 items
-



```
Select an option from the menu above: 1

-----
Sub-menu 1: Available genres
-----
1. Fiction
2. Dystopian
3. Classic
4. Sport
-----
Current menu selection:
Main menu - Option 4 (Other view choices)
-----
1 - View available genres
2 - View available authors
3 - View Titles with author details
4 - Return to main menu
```

Sub menu 1 – View available genres

```
Select an option from the menu above: 2
```

```
-----  
Sub-menu 2: Available authors
```

- ```

1. F. Scott Fitzgerald
2. George Orwell
3. Harper Lee
4. ABG
```

```

Current menu selection:
Main menu - Option 4 (Other view choices)
```

- ```
-----  
1 - View available genres  
2 - View available authors  
3 - View Titles with author details  
4 - Return to main menu
```

Sub menu 2 – View available authors

```
Select an option from the menu above: 3
```

```
-----  
Sub-menu 3: Available titles with author details
```

```
-----  
Title: ABG  
Author: Sports Illustrated  
  
Title: F. Scott Fitzgerald  
Author: The Great Gatsby  
  
Title: George Orwell  
Author: 1984  
  
Title: Harper Lee  
Author: To Kill a Mockingbird
```

```
-----  
Current menu selection:  
Main menu - Option 4 (Other view choices)
```

- ```

1 - View available genres
2 - View available authors
3 - View Titles with author details
4 - Return to main menu
```

---

### Sub menu 3 – View combination of available authors and titles

---

```
Select an option from the menu above: 4
```

```
Return to main menu
```

```

Main menu
```

- ```
-----  
1 - View entire catalog (Sorted by Title)  
2 - View entire catalog (Sorted by Genre)  
3 - View catalog by publication type (Books and Magazines)  
4 - Menu for other view options (authors, genres etc)  
5 - Search for an item  
6 - Add a new item  
7 - Remove an item  
8 - Exit
```

```
Select an option from the menu above to proceed:
```

Sub menu 4 – Return to main menu

6 Main Menu – option 5 (Search for an item)

Main Menu – Option 5 (Search for an item)

```
Select an option from the menu above to proceed: 5
```

```
-----  
Current menu selection:  
Main menu - Option 5 (Search for an item)  
-----
```

```
1 - Search item by title  
2 - Search item by Id  
3 - Return to main menu
```

```
Select an option from the menu above: 3
```

Main menu – option 5 (Other view options)

- This presents a sub menu of 3 items
-

```
Select an option from the menu above: 1
```

```
-----  
Sub-menu 1: Search item by title  
-----
```

```
Enter the title of the item ('Enter x' to exit): The Great Gatsby
```

```
>>>>> Found item with title: The Great Gatsby
```

```
The Great Gatsby  
  by F. Scott Fitzgerald - 1925  
ISBN: 123456789  
Genre: Fiction
```

Sub menu 1 – Search by title

```
Select an option from the menu above: 2
```

```
-----  
Sub-menu 1: Search item by ID  
-----
```

```
Enter the ID of the item ('Enter -1' to exit): 200
```

```
>>>>> Found item with id: 200
```

```
1984  
  by George Orwell - 1949  
ISBN: 9780140817744  
Genre: Dystopian
```

Sub menu 2 – Search by Id

```
Select an option from the menu above: 4

Return to main menu

-----
Main menu
-----
1 - View entire catalog (Sorted by Title)
2 - View entire catalog (Sorted by Genre)
3 - View catalog by publication type (Books and Magazines)
4 - Menu for other view options (authors, genres etc)
5 - Search for an item
6 - Add a new item
7 - Remove an item
8 - Exit

Select an option from the menu above to proceed:
```

Sub menu 3 – Return to main menu

7 Main Menu – option 6 (Add items to the catalog)

Main Menu – Option 6 (Add item to the catalog)

```
-----
Main menu
-----
1 - View entire catalog (Sorted by Title)
2 - View entire catalog (Sorted by Genre)
3 - View catalog by publication type (Books and Magazines)
4 - Menu for other view options (authors, genres etc)
5 - Search for an item
6 - Add a new item
7 - Remove an item
8 - Exit

Select an option from the menu above to proceed: 6

-----
Current menu selection:
Main menu - Option 4 (Add Items to catalog)
-----
1 - Load additional seed data into the current catalog
2 - Add a new item to the catalog
3 - Return to main menu

Select an option from the menu above:
```

Main menu – option 6 (Add items to the catalog)

- This presents a sub menu of 3 items

```
Select an option from the menu above: 1

-----

Sub-menu 1: Load additional seed data
-----

Loading 3 new items...

Pride and Prejudice
  by Jane Austen - 1813
  ISBN: 9780140434262
  Genre: Romance

Moby-Dick
  by Herman Melville - 1851
  ISBN: 9780140434262
  Genre: Adventure

The Lord of the Rings
  by J.R.R. Tolkien - 1954
  ISBN: 9788445071793
  Genre: Fantasy / Adventure

1 - Load additional seed data into the current catalog
2 - Add a new item to the catalog
3 - Return to main menu
```

Sub menu 1 – Load additional seed data to the catalog

```
Select an option from the menu above: 2

-----

Sub-menu 2: Add a new item to the catalog
-----

Enter the following details for the item ('Enter -1' to exit):

ID: 800
Title: War and peace
Author: Leo Tolstoy
ISBN: 98743220
Genre: Fiction
Year: 1869

Adding "War and peace" to the catalog...

War and peace
  by Leo Tolstoy - 1869
  ISBN: 98743220
  Genre: Fiction
```

Sub menu 2 – Add a new item to the catalog

```
Select an option from the menu above: 4

Return to main menu

-----

Main menu
-----

1 - View entire catalog (Sorted by Title)
2 - View entire catalog (Sorted by Genre)
3 - View catalog by publication type (Books and Magazines)
4 - Menu for other view options (authors, genres etc)
5 - Search for an item
6 - Add a new item
7 - Remove an item
8 - Exit

Select an option from the menu above to proceed:
```

Sub menu 3 – Return to main menu

8 Main Menu – option 7 (Remove an item)

Main Menu – Option 7 (Add item to the catalog)

```
Main menu
-----
1 - View entire catalog (Sorted by Title)
2 - View entire catalog (Sorted by Genre)
3 - View catalog by publication type (Books and Magazines)
4 - Menu for other view options (authors, genres etc)
5 - Search for an item
6 - Add a new item
7 - Remove an item
8 - Exit

Select an option from the menu above to proceed: 7

-----
Current menu selection:
Main menu - Option 4 (Remove an item)
-----

1 - Remove last item added
2 - Remove all items
3 - Return to main menu

Select an option from the menu above:
```

Main menu – option 6 (Add items to the catalog)

- This presents a sub menu of 3 items
-

```
Select an option from the menu above: 1

-----
Sub-menu 1: Remove last item added
-----

Removing last item added "Sports Illustrated" from the catalog...

Here are the items remaining in the catalog:
1. The Great Gatsby
   by F. Scott Fitzgerald - 1925
   ISBN: 123456789
   Genre: Fiction
2. 1984
   by George Orwell - 1949
   ISBN: 9780140817744
   Genre: Dystopian
3. To Kill a Mockingbird
   by Harper Lee - 1960
   ISBN: 9781846572562
   Genre: Classic

1 - Remove last item added
2 - Remove all items
3 - Return to main menu

Select an option from the menu above:
```

Sub menu 1 – Remove last item

```
1 - Remove last item added
2 - Remove all items
3 - Return to main menu

Select an option from the menu above: 2

-----
Sub-menu 2: Remove all items
-----

All items have been removed...
There are no items in the catalog.

1 - Remove last item added
2 - Remove all items
3 - Return to main menu

Select an option from the menu above:
```

Sub menu 2 – Remove all items

```
Select an option from the menu above: 4

Return to main menu
-----
Main menu
-----
1 - View entire catalog (Sorted by Title)
2 - View entire catalog (Sorted by Genre)
3 - View catalog by publication type (Books and Magazines)
4 - Menu for other view options (authors, genres etc)
5 - Search for an item
6 - Add a new item
7 - Remove an item
8 - Exit

Select an option from the menu above to proceed:
```

Sub menu 3 – Return to main menu

9 Main Menu – option 8 (Exit)

Main Menu – Option 5 (Search for an item)

```
Return to main menu
-----
Main menu
-----
1 - View entire catalog (Sorted by Title)
2 - View entire catalog (Sorted by Genre)
3 - View catalog by publication type (Books and Magazines)
4 - Menu for other view options (authors, genres etc)
5 - Search for an item
6 - Add a new item
7 - Remove an item
8 - Exit

Select an option from the menu above to proceed: 8

Good bye!

Destructor for The Great Gatsby called.
Destructor for LibraryItem called...

Destructor for 1984 called.
Destructor for LibraryItem called...

Destructor for To Kill a Mockingbird called.
Destructor for LibraryItem called...

Destructor for Book item with Sports Illustrated called
Destructor for LibraryItem called...

Destructor for Catalog class is called
```

Main menu – option 8 (Exit)

- A good bye message is displayed
- Destructors are called with derived classes first destroyed, followed by LibraryItem. The Catalog class is destroyed last.

III Complete Code

1 LibraryItem.h

```
#pragma once

#include<iostream>

using std::string;
using std::cout;
using std::endl;

class LibraryItem
{
private:
    string title;
    string author;
    string ISBN;
    int id;

public:
    LibraryItem(int d, string t, string a, string i);

    // pure virtual function
    virtual void displayDetails() const = 0;
    virtual string displayGenre() const = 0;

    virtual ~LibraryItem()
    {
        cout << "Destructor for LibraryItem called..." << endl << endl;
    };

    // getters
    int getId() const;

    std::string getTitle() const;

    std::string getAuthor() const;

    std::string getISBN() const;

    virtual string getGenre() const { return genre; };

protected:
    std::string genre;
    int year = 0;
};
```


2 LibraryItem.cpp

```
#include "LibraryItem.h"
#include <iostream>
#include<vector>

using namespace std;

LibraryItem::LibraryItem(int d, string t, string a, string i) {
    title = t;
    author = a;
    ISBN = i;
    id = d;
}

int LibraryItem::getId() const { return id; }
string LibraryItem::getTitle() const { return title; }
string LibraryItem::getAuthor() const { return author; }

string LibraryItem::getISBN() const { return ISBN; }
```

3 Book.h

```
#pragma once

#include<iostream>
#include "LibraryItem.h"
#include<string>

using std::string;
using std::cout;
using std::endl;

class Book : public LibraryItem {
protected:

    /*std::string genre;
    int year = 0;*/

public:
    Book(int d, const string& t, const string& a,
        const string& i, const string& g, int y);

    void displayDetails() const override;
    string displayGenre() const override;

    // Getters for attributes to be used in functors and templates

    int getYear() const;
    string getGenre() const override;

    // Implement constructors and other necessary member functions
    ~Book()
    {
```

```

        cout << "Destructor for " << getTitle() << " called." << endl ;
    }

};

```

4 Book.cpp

```

#include "Book.h"
#include <iomanip>
#include <iostream>

using std::cout;
using std::endl;

using std::setw;
using std::left;
using std::right;

Book::Book(int d, const string& t, const string& a,
           const string& i, const string& g, int y) : LibraryItem(d, t, a, i) {
    genre = g; year = y;
};

void Book::displayDetails() const
{
    cout << getTitle() << endl;
    cout << "        by " << getAuthor() << " - " << getYear() << endl;
    cout << "        ISBN: " << getISBN() << endl;
    cout << "        Genre: " << getGenre() << endl << endl;
}

string Book::displayGenre() const { return genre; }
int Book::getYear() const { return year; }

string Book::getGenre() const { return genre; }

```

5 Magazine.h

```

#pragma once
#include <iostream>
#include "LibraryItem.h"
#include <string>

using std::string;
class Magazine : public LibraryItem {

protected:
    std::string genre;
    int year = 0;

public:
    Magazine(int d, const string& t, const string& a,
            const string& i, const string& g, int y);

    void displayDetails() const override;

```

```

        string displayGenre() const override;

        // Getters for attributes to be used in functors and templates
        int getYear() const;
        string getGenre() const override;

        ~Magazine()
        {
            cout << "Destructor for Book item with " << getTitle() << " called" <<
endl;
        }

};

```

6 Magazine.cpp

```

#include "Magazine.h"
#include <iostream>

using std::cout;
using std::endl;

Magazine::Magazine(int d, const string& t, const string& a,
                    const string& i, const string& g, int y) : LibraryItem(d, t, a, i)
{
    genre = g; year = y;
};

void Magazine::displayDetails() const
{
    cout << getTitle() << endl;
    cout << "        by " << getAuthor() << " - " << getYear() << endl;
    cout << "        ISBN: " << getISBN() << endl;
    cout << "        Genre: " << getGenre() << endl << endl;
}

string Magazine::displayGenre() const { return genre; }
int Magazine::getYear() const { return year; }

string Magazine::getGenre() const { return genre; }

```

7 Catalog.h

```

#pragma once

#include<iostream>
#include<vector>
#include<list>
#include<functional>
#include<algorithm>
#include<string>
#include<cstdlib>
#include<map>

#include "Book.h"

```

```

#include "Catalog.h"

using std::cout;
using std::endl;
using namespace std;

// Template class for Catalog
template <typename T>
class Catalog {
private:
    string userName;
    int index = 0;
    vector<T*> itemsList; // vector to store all categories of library items
    map<string, string> titleAndAuthor; // map to store title and author
    vector<LibraryItem*>::iterator it; // iterator for itemsList vector
    map<string, string>::iterator itMap; // iterator for titleAndAuthor map

public:

    Catalog(string name); // default constructor

    // destructor
    ~Catalog()
    {
        for (T* item : itemsList)
        {
            delete item;
        }
        itemsList.clear();
        cout << "Destructor for Catalog class is called" << endl;
    }

    // Accessor method for items
    const vector<T*>& getItemsList() const;

    // base condition for adding to vector
    void addItemToMainList();

    // variadic function
    template<typename... Args>
    void addItemToMainList(T* value, Args... args);

    void addItemToMap();

    template<typename... Args>
    void addItemToMap(T* value, Args... args);

    void totalItemsInList();
    void removeLastItem();
    void removeItem(const T& item);
    void removeAllItems();

    void displayAll();

    void displayAuthors();

    void displayAllGenres();

```

```

static void showAuthors(T* n, int index);

static void showGenres(T* n, int index);

void displayTitleAndAuthor();

// Functor for sorting by title
struct SortByTitle {
    bool operator()(const T* a, const T* b) {
        return a->getTitle() < b->getTitle();
    }
};

// function for sort by title
void sortByTitle();

// Functor for sorting by genre
struct SortByGenre {
    bool operator()(const T* a, const T* b) {
        return a->getGenre() < b->getGenre();
    }
};

void sortByGenre();

template <typename Func>
T* searchItem(Func func) const;

const T* searchItem(const string& key) const;

const T* searchItem(int key) const;

}; // End Catalog class

//=====
////////// ** CONSTRUCTOR FUNCTION DEFINITION ** //////////
//=====
template <typename T>
Catalog<T>::Catalog(string name)
{
    userName = name;
}

//=====
////////// ** GETTER FOR ITEMSLIST ** //////////
//=====
template <typename T>
const vector<T*>& Catalog<T>::getItemsList() const {
    return itemsList;
}

//=====
////////// ** ADD ITEMS FUNCTION DEFINITIONS ** //////////
//=====
template<typename T>
void Catalog<T>::addItemToMainList()

```

```

{};

// variadic function to add multiple items to the vector
template<typename T>
template<typename... Args>
void Catalog<T>::addItemToMainList(T* value, Args... args)
{
    itemsList.push_back(value); // Push the first element
    addItemToMainList(args...); // Recursively push the rest
    titleAndAuthor.insert(pair<string, string>(value->getAuthor(), value->getTitle()));
}

template<typename T>
void Catalog<T>::addItemToMap() {};

template<typename T>
template<typename... Args>
void Catalog<T>::addItemToMap(T* value, Args... args)
{
    titleAndAuthor[value->getTitle] = value->getAuthor();
    addItemToMap(args...);
    titleAndAuthor.insert({ value->getTitle(), value->getAuthor() });

    //titleAndAuthor[value->getTitle()] = value->getISBN();
    //titleAndAuthor.insert({ value->getTitle(), value->getISBN() });
};

//=====
//***** ** DISPLAY FUNCTION DEFINITIONS ** *****/
//
//=====
template<typename T>
void Catalog<T>::displayAll()
{
    if (itemsList.size() < 1)
        cout << "There are no items in the list.";
    else
    {
        for (it = itemsList.begin(), index = 1; it != itemsList.end(); it++,
++index)
        {
            cout << index << ". ";
            (*it)->displayDetails();
        }
    }
};

template<typename T>
void Catalog<T>::displayTitleAndAuthor()
{
    if (itemsList.size() < 1)
        cout << "There are no items in the list.";
    else
    {
        for (itMap = titleAndAuthor.begin(); itMap != titleAndAuthor.end();
itMap++)
        {

```

```

        cout << "Title: " << (*itMap).first << endl;
        cout << "Author: " << (*itMap).second << endl << endl;
    }
}

};

template<typename T>
void Catalog<T>::showGenres(T* n, int index)
{
    cout << index << ". " << n->getGenre() << endl;
}

template<typename T>
void Catalog<T>::displayAllGenres()
{
    int index = 1;
    for_each(itemsList.begin(), itemsList.end(), [&index](T* n)
        {showGenres(n, index); ++index; });
}

template<typename T>
void Catalog<T>::showAuthors(T* n, int index)
{
    cout << index << ". " << n->getAuthor() << endl;
}

template<typename T>
void Catalog<T>::displayAuthors()
{
    int index = 1;
    for_each(itemsList.begin(), itemsList.end(), [&index](LibraryItem* n)
        {showAuthors(n, index); ++index; });
}

template <typename T>
void Catalog<T>::totalItemsInList()
{
    cout << "Total number of items in the catalog: ";
    cout << itemsList.size() << endl;
}

//=====
////////// ** REMOVE ITEMS FUNCTION DEFINITIONS ** //////////
//=====
template<typename T>
void Catalog<T>::removeLastItem()
{
    itemsList.pop_back();
}

template<typename T>
void Catalog<T>::removeItem(const T& item)
{
    itemsList.pop_back();
};

```

```

template<typename T>
void Catalog<T>::removeAllItems()
{
    itemsList.clear();
}

//=====
////////// ** SORTING FUNCTION DEFINITIONS ** //////////
//=====

// sorting items by title using functor
template<typename T>
void Catalog<T>::sortByTitle() {
    sort(itemsList.begin(), itemsList.end(), SortByTitle());
}

// sorting items by genre using functor
template<typename T>
void Catalog<T>::sortByGenre() {
    sort(itemsList.begin(), itemsList.end(), SortByGenre());
}

//=====
////////// ** SEARCHING FUNCTION DEFINITIONS ** //////////
//=====

// custom search function
template <typename T>
template <typename Func>
T* Catalog<T>::searchItem(Func func) const {
    return *itemsList.begin();
    //auto it = find_if(itemsList.begin(), itemsList.end(), [&func](T* ptr) {
    //    return func(*ptr); });
    //if (it != itemsList.end()) {
    //    return *it;
    //}
    //throw std::runtime_error("Item not found");
}

template <typename T>
const T* Catalog<T>::searchItem(const string& key) const {
    auto it = std::find_if(itemsList.begin(), itemsList.end(), [&key](const T*
item) {
        return item->getTitle() == key;
    });

    if (it != itemsList.end()) {
        return *it;
    }
    else {
        throw std::runtime_error("Item not found");
    }
}

// Method to get a int key
template <typename T>
const T* Catalog<T>::searchItem(int key) const {

```



```

        auto it = find_if(itemsList.begin(), itemsList.end(), [key](const T* item) {
            return item->getId() == key;
        });

        if (it != itemsList.end()) {
            return *it;
        }
        else {
            throw std::runtime_error("Item not found");
        }
    }
}

```

8 Main.cpp

```

#include<iostream>
#include<vector>
#include<list>
#include<functional>
#include<algorithm>
#include<iomanip>
#include<limits>

#include "Book.h"
#include "Catalog.h"
#include "LibraryItem.h"
#include "Magazine.h"

using std::string;
using std::vector;
using std::list;
using std::sort;
using std::cout;
using std::cin;
using std::endl;

// get and validate integer input
int getIntInput(const string& promptUser, bool& exitCondition);

// get and validate string input
string getStringInput(const string& prompt, bool& exitCondition);

// Main function starts here

int main() {

    cout << "===== " << endl;
    cout << " *** Welcome to the Library Management System *** " << endl;
    cout << "===== " << endl <<
endl;

    string userName; // user name input for Inventory Constructor

```

```

// Take user input to initialize the Catalog class
cout << "Enter any character to log into the inventory system: ";
getline(cin, userName);
cout << endl;

// Create the library catalog and populate with fake data (for testing)
Catalog<LibraryItem> allbooks(userName);

// create seed data with 4 items
Book* b1 = new Book(100, "The Great Gatsby", "F. Scott Fitzgerald",
"123456789", "Fiction", 1925);
Book* b2 = new Book(200, "1984", "George Orwell", "9780140817744",
"Dystopian", 1949);
Book* b3 = new Book(300, "To Kill a Mockingbird", "Harper Lee",
"9781846572562", "Classic", 1960);
Magazine* m1 = new Magazine(400, "Sports Illustrated", "ABG", "985123456789",
"Sport", 2024);

// add seed data to vector in the catalog class
cout << "Loading seed data... " << endl << endl;
allbooks.addItemToMainList(b1, b2, b3, m1);

// take user input for choice of action
int choice{ 0 };

// Main while loop for program navigation
while (choice != 8)
{
    cout << "-----" <<
endl;
    cout << "Main menu" << endl;
    cout << "-----" <<
endl;
    cout << "1 - View entire catalog (Sorted by Title)" << endl;
    cout << "2 - View entire catalog (Sorted by Genre)" << endl;
    cout << "3 - View catalog by publication type (Books and Magazines)" << endl;
    cout << "4 - Menu for other view options (authors, genres etc)" << endl;
    cout << "5 - Search for an item" << endl;
    cout << "6 - Add a new item" << endl;
    cout << "7 - Remove an item" << endl;
    cout << "8 - Exit" << endl << endl;

    cout << "Select an option from the menu above to proceed: ";
    cin >> choice;
    cout << endl;

    // Display all items in the catalog - sorted by title
    if (choice == 1)
    {
        cout << "-----"
<< endl;
        cout << "Current menu selection: " << endl << "Main menu - Option 1
(View entire collection sorted by title)" << endl;
        cout << "-----"
<< endl;
        cout << "Displaying entire collection sorted by title..." << endl<<
endl;
        allbooks.sortByTitle();
        allbooks.displayAll();
    }
}

```

```

        allbooks.totalItemsInList();
        cout << endl;
    }
    // Endif - catalog sorted by title

    // Display all items in the catalog - sorted by genre
    if (choice == 2)
    {
        cout << "-----"
<< endl;
        cout << "Current menu selection: " << endl << "Main menu - Option 2
(View entire collection sorted by genre)" << endl;
        cout << "-----"
<< endl;
        cout << "Displaying entire collection sorted by genre..." << endl <<
endl;
        allbooks.sortByGenre();
        allbooks.displayAll();
        allbooks.totalItemsInList();
        cout << endl;
    }
    // Endif - sorted by genre

    // Display items by publication type
    if (choice == 3)
    {
        cout << "-----"
<< endl;
        cout << "Current menu selection: " << endl << "Main menu - Option 3
(View by publication type)" << endl;
        cout << "-----"
<< endl;

        int viewOptionsChoice{ 0 };

        int collectionChoice{ 0 };
        while (collectionChoice != 3)
        {
            cout << "1 - View books in the catalog" << endl;
            cout << "2 - View magazines in the catalog" << endl;
            cout << "3 - Return to main menu" << endl;

            cout << endl << "Select an option from the menu above: ";

            cin >> collectionChoice;
            cout << endl;

            switch (collectionChoice)
            {
            case 1: // Display books only
                cout << "-----"
-----" << endl;
                cout << "Sub-menu 1: View books collection" << endl;
                cout << "-----"
-----" << endl;
                cout << "In progress... " << endl << endl;
                break;

            case 2: // Display magazines only

```

```

        cout << "-----" << endl;
        cout << "Sub-menu 2: View magazine collection" << endl;
        cout << "-----" << endl;

        cout << "In progress... " << endl << endl;
        break;

    case 3:
        cout << "Return to main menu" << endl;
        break;

    default:
        cout << "Wrong choice. Please select the correct option"
<< endl;
        break;
    }
} // Endif - Display items by publication type

if (choice == 4) // Display available genres, authors and other
combinations
{

    int collectionChoice{ 0 };
    while (collectionChoice != 4)
    {
        cout << "-----" << endl;
        cout << "Current menu selection: " << endl << "Main menu -
Option 4 (Other view choices)" << endl;
        cout << "-----" << endl;

        cout << "1 - View available genres" << endl;
        cout << "2 - View available authors" << endl;
        cout << "3 - View Titles with author details" << endl;
        cout << "4 - Return to main menu" << endl;

        cout << endl << "Select an option from the menu above: ";

        cin >> collectionChoice;
        cout << endl;

        switch (collectionChoice)
        {
            case 1: // Display genres only
                cout << "-----" << endl;
                cout << "Sub-menu 1: Available genres" << endl;
                cout << "-----" << endl;

                allbooks.displayAllGenres();
                cout << endl;
                break;

            case 2: // Display authors only

```

```

-----" << endl;
cout << "Sub-menu 2: Available authors" << endl;
cout << "-----" << endl;

allbooks.displayAuthors();
cout << endl;
break;

collection by genre
case 3: // Display by title and author // Display
cout << "-----" << endl;
cout << "Sub-menu 3: Available titles with author
details" << endl;
cout << "-----" << endl;

allbooks.displayTitleAndAuthor();
break;

case 4:
cout << "Return to main menu" << endl;
break;

default:
cout << "Wrong choice. Please select the correct
option" << endl;
break;
}
} // EndIf - Display available genres // Endif - Display available
authors

if (choice == 5) //Search for an item
{
    int searchChoice{ 0 };
    while (searchChoice != 3)
    {
        cout << "-----" << endl;
        cout << "Current menu selection: " << endl << "Main menu -
Option 5 (Search for an item)" << endl;
        cout << "-----" << endl;

        cout << "1 - Search item by title" << endl;
        cout << "2 - Search item by Id" << endl;
        cout << "3 - Return to main menu" << endl;

        cout << endl << "Select an option from the menu above: ";

        while (!(cin >> searchChoice) || searchChoice < 1 ||
searchChoice > 3) {
            cout << "Invalid input. Please enter a number
between 1 and 3: ";

```

```

        cin.clear(); // clear the error flag
        cin.ignore(numeric_limits<streamsize>::max(),
'\n'); // discard invalid input
    }
    cout << endl;

    switch (searchChoice)
    {
    case 1:
    {
        cout << "-----" << endl;
        cout << "Sub-menu 1: Search item by title" << endl;
        cout << "-----" << endl;

        cout << "Enter the title of the item ('Enter x' to
exit): ";

        string title;
        cin.ignore(); // Clear the buffer before taking a
new line input
        getline(cin, title);
        if (title == "x") {
            break;
        }
        try {
            const LibraryItem* found =
allbooks.searchItem(title);
            if (found) {
                cout << endl << ">>>>>> Found item
with title: " << found->getTitle() << endl << endl;

                found->displayDetails();
            }
            else {
                cout << "Item not found." << endl <<
endl;
            }
        }
        catch (const runtime_error& e) {
            cerr << e.what() << endl;
        }
        break;

    case 2:
    {
        cout << "-----" << endl;
        cout << "Sub-menu 1: Search item by ID" << endl;
        cout << "-----" << endl << endl;

        cout << "Enter the ID of the item ('Enter -1' to
exit): ";

        int id;
        while (!(cin >> id) || id < -1) {

```

```

        cout << "Invalid input. Please enter a valid
ID (or -1 to exit): ";
        cin.clear(); // clear the error flag
        cin.ignore(numeric_limits<streamsize>::max(),
'\n'); // discard invalid input
    }

    if (id == -1) {
        break;
    }

    try {
        const LibraryItem* found =
allbooks.searchItem(id);
        if (found) {
            cout << endl << ">>>>>> Found item
with id: " << found->getId() << endl << endl;
            found->displayDetails();
        }
        else {
            cout << "Item not found." << endl <<
endl;
        }
    }
    catch (const runtime_error& e) {
        cerr << e.what() << endl;
    }
    break;

    case 3:
        cout << "Return to main menu" << endl;
        break;

    default:
        cout << "Wrong choice. Please select the correct
option" << endl;
        break;
    }
} //Endif - Search for an item

if (choice == 6) //Add a new item
{
    cout << "-----"
----" << endl;
    cout << "Current menu selection: " << endl << "Main menu -
Option 4 (Add Items to catalog)" << endl;
    cout << "-----"
----" << endl << endl;

    int addItemsChoice{ 0 };

    while (addItemsChoice != 3)
    {
        cout << "1 - Load additional seed data into the current
catalog" << endl;

```

```

        cout << "2 - Add a new item to the catalog" << endl;
        cout << "3 - Return to main menu" << endl;

    cout << endl << "Select an option from the menu above: ";

    cin >> addItemChoice;
    cout << endl;

    switch (addItemChoice)
    {
    case 1: // sub menu 1 - add additional seed data for testing
        {
            cout << "-----" << endl;
            cout << "Sub-menu 1: Load additional seed data" <<
endl;
            cout << "-----" << endl << endl;

            cout << "Loading 3 new items..." << endl << endl;

            Book* b4 = new Book(500, "Pride and Prejudice",
"Jane Austen", "9780140434262", "Romance", 1813);
            Book* b5 = new Book(600, "Moby-Dick", "Herman
Melville", "9780140434262", "Adventure", 1851);
            Book* b6 = new Book(700, "The Lord of the Rings",
"J.R.R. Tolkien", "9788445071793", "Fantasy / Adventure", 1954);

            allbooks.addItemToMainList(b4, b5, b6);

            b4->displayDetails();
            b5->displayDetails();
            b6->displayDetails();
        }
        break; // End of case 1

    case 2: // sub menu 2 - Add a new item to the catalog
        {
            cout << "-----" << endl;
            cout << "Sub-menu 2: Add a new item to the catalog" <<
endl;
            cout << "-----" << endl << endl;

            cout << "Enter the following details for the item
('Enter -1' to exit): " << endl << endl;

            bool exitCondition = false;

            while (exitCondition != -1)
            {
                int id = getIntInput("ID: ", exitCondition);
                if (exitCondition) {
                    cout << "Exiting..." << endl << endl;
                    break;
                }
                string title = getStringInput("Title: ",
exitCondition);

```



```

        if (exitCondition) {
            cout << "Exiting..." << endl << endl;
            break;
        }
        string author = getStringInput("Author: ",
exitCondition);

        if (exitCondition) {
            cout << "Exiting..." << endl << endl;
            break;
        }
        string ISBN = getStringInput("ISBN: ",
exitCondition);

        if (exitCondition) {
            cout << "Exiting..." << endl << endl;
            break;
        }
        string genre = getStringInput("Genre: ",
exitCondition);

        if (exitCondition) {
            cout << "Exiting..." << endl << endl;
            break;
        }
        int year = getIntInput("Year: ",
exitCondition);

        if (exitCondition) {
            cout << "Exiting..." << endl << endl;
            break;
        }
        Book* B = new Book(id, title, author, ISBN,
genre, year);
        cout << endl << "Adding \"<\" << title << "\"
to the catalog... \" << endl << endl;

        B->displayDetails();

        allbooks.addItemToMainList(B);
    } // end while loop
}
break; // End of case 2

case 3:
    cout << "Return to main menu" << endl;
    break;

default:
    cout << "Wrong choice. Please select the correct option"
<< endl;
    break;
} // End Switch

} // End while loop

} //Endif - Add a new item

if (choice == 7) // Remove an item
{
    cout << "-----"
-----" << endl;

```

```

        cout << "Current menu selection: " << endl << "Main menu -
Option 4 (Remove an item)" << endl;
        cout << "-----"
-----" << endl << endl;

        int removeItemsChoice{ 0 };

        while (removeItemsChoice != 3)
        {
            cout << "1 - Remove last item added" << endl;
            cout << "2 - Remove all items" << endl;
            cout << "3 - Return to main menu" << endl;

            cout << endl << "Select an option from the menu above: ";

            cin >> removeItemsChoice;
            cout << endl;

            switch (removeItemsChoice)
            {
            case 1:
                cout << "-----"
-----" << endl;
                cout << "Sub-menu 1: Remove last item added" <<
endl;
                cout << "-----"
-----" << endl << endl;

                cout << "Removing last item added \";
                allbooks.getLastAdded();
                cout << "\" from the catalog..." << endl<<endl;

                allbooks.removeLastItem();

                cout << "Here are the items remaining in the
catalog: " << endl;
                allbooks.displayAll();

                break;

            case 2:
                cout << "-----"
-----" << endl;
                cout << "Sub-menu 2: Remove all items" << endl;
                cout << "-----"
-----" << endl << endl;

                cout << "All items have been removed... " << endl;
                allbooks.removeAllItems();
                allbooks.displayAll();
                cout << endl << endl;
                break;

            case 3:
                cout << "Return to main menu" << endl;
                break;

            default:
                break;

```

```

        }
    }

    } // Endif - Remove an item

    if (choice == 8) // Exit the program
    {
        cout << "Good bye!" << endl;
    } // Endif - Exit the program

}
// End of main while loop for program navigation

cout << endl;

return 0;
}
// End of function main

//=====
//***** ** FUNCTION DEFINITIONS ** *****/
//=====

int getIntInput(const string& promptUser, bool &exitCondition) {
    int input;
    while (true) {

        cout << promptUser;
        cin >> input;
        if (input == -1)
        {
            exitCondition = true;
            return -1;
        }
        if (cin.fail()) {
            cin.clear(); // clear the error flags
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // discard
invalid input
            cout << "Invalid input... Enter an integer value." << endl;
        }
        else {
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // discard
remaining input
            return input;
        }
    }
}

// get and validate string input
string getStringInput(const string& promptUser, bool& exitCondition) {
    string input;
    cout << promptUser;

```

```
    getline(cin, input);  
    if (input == "x")  
    {  
        exitCondition = true;  
        return "";  
    }  
    return input;  
}
```