

COS 212 Assignment

by Naing Min Khant

Submission date: 29-Jun-2023 01:23PM (UTC+0800)

Submission ID: 2123288929

File name: COS_212_Assignment_Documentation_-_Naing_Min_Khant.docx (1.79M)

Word count: 1017

Character count: 4726

British University College

2023-2024 Academic Year



Assignment

COS212 – Object-Oriented Programming with C++

by

Naing Min Khant (HDIT Class - D)

Presented On – June 30 2023

Table of Content

❖ Task 1 - I/O File Manipulation	2
❖ Task 2 - Inheritance	20
❖ Task 3 - Encapsulation	23

Task 1 - I/O File Manipulation

A simple library management system with file manipulation including read, write, edit, and delete features. Once we run a program, it will ask for the guest's name and print the greeting message as shown in Fig (1.1). Then, Main Menu will display the options. We can see the code behind this output in Fig (1.2), Fig (1.2.1), and Fig (1.3).

```
Please Enter your name : Naing Min Khant
-----
Hello, Naing Min Khant. Welcome to our library.
-----
→ Main Menu (Choose one number).
    1. View book List
    2. Add new book
    3. Edit book information
    4. Delete book
    5. Exit the program
Choose a number : █
```

Fig(1.1)



```
1 int main()
2 {
3     LibraryMenu library_menu;
4     library_menu.displayMainMenu();
5     return 0;
6 }
```

Fig(1.2)



```
1 Menu::Menu() // constructor definition outside class
2 {
3     cout << "Please Enter your name : ";
4     getline(cin, guest_name);
5     cout << endl
6     << "-----" << endl
7     << "Hello, " << guest_name << ". Welcome to our library." << endl
8     << "-----" << endl;
9 }
```

Fig(1.2.1)

```
● ● ●
1 void LibraryMenu::displayMainMenu() // for main menu
2 {
3     cout << endl
4     << " - Main Menu (Choose one number)." << endl
5     << endl;
6
7     displayMenuList(MAIN_MENU, sizeof(MAIN_MENU) / sizeof(string)); // displaying main menus
8     cout << endl
9     << "Choose a number : ";
10    cin >> selected_menu_no;
11
12    switch (selected_menu_no)
13    {
14        case 1:
15            displaySubMenu();
16            break;
17        case 2:
18            library.addNewBook();
19            cout << " _____" * << endl;
20            displayMainMenu();
21            break;
22        case 3:
23            library.editBook();
24            cout << " _____" * << endl;
25            displayMainMenu();
26            break;
27        case 4:
28            library.deleteBook();
29            cout << " _____" * << endl;
30            displayMainMenu();
31            break;
32        case 5:
33            cout << endl
34            << "Good Bye, " << guest_name << ". See you again." << endl;
35            break;
36        default:
37            cout << "Invalid number!" << endl;
38            cout << "Choose again!" << endl;
39            cin.clear();
40            cin.ignore(numeric_limits<streamsize>::max(), '\n');
41            displayMainMenu();
42            break;
43    }
44 }
```

Fig(1.3)

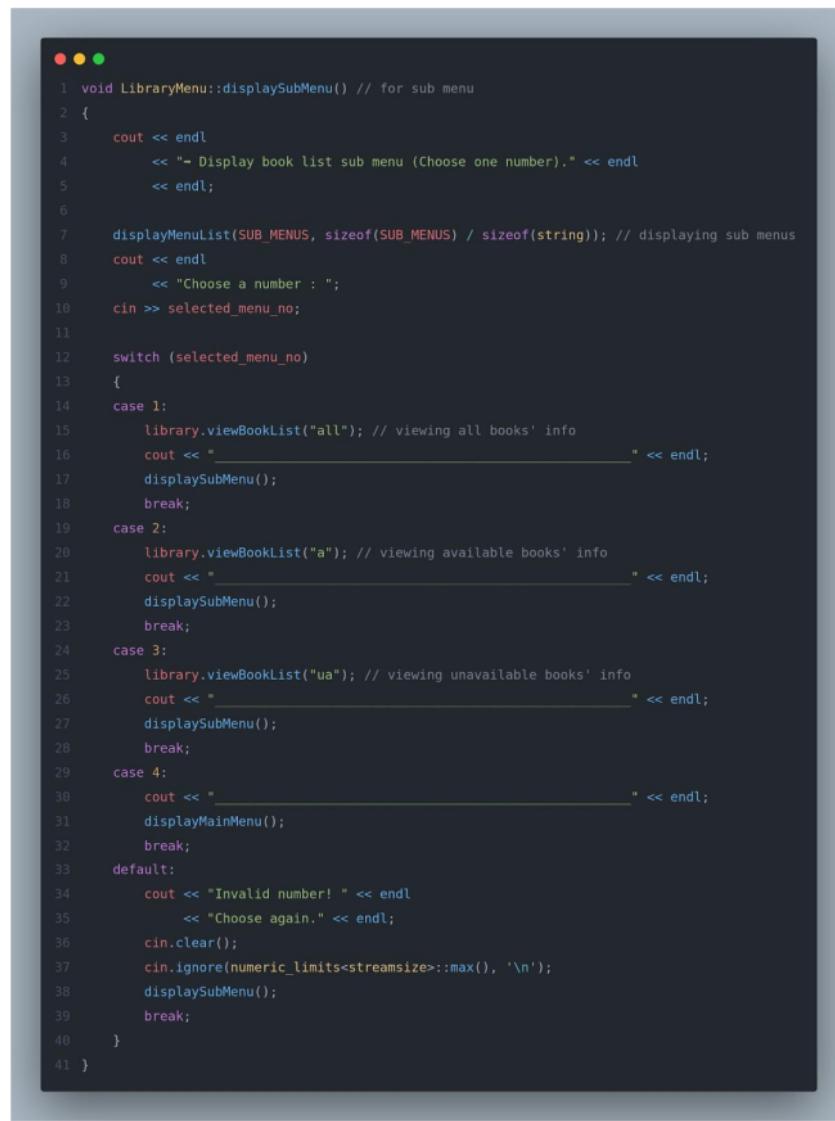
If we choose the number 1, a sub-menu will display. There are more options for viewing books. The sub-menu will be shown so that the user can choose the option to view the book list as shown in Fig (1.4).

```
→Display book list sub menu (Choose one number).
    1. View all books in library
    2. View all available books in library
    3. View all unavailable books in library
    4. Return to main menu

Choose a number :
```

Fig(1.4)

The code working behind this output can be seen in Fig (1.5).



```
1 void LibraryMenu::displaySubMenu() // for sub menu
2 {
3     cout << endl
4     << "-- Display book list sub menu (Choose one number)." << endl
5     << endl;
6
7     displayMenuList(SUB_MENU, sizeof(SUB_MENU) / sizeof(string)); // displaying sub menus
8     cout << endl
9     << "Choose a number : ";
10    cin >> selected_menu_no;
11
12    switch (selected_menu_no)
13    {
14        case 1:
15            library.viewBookList("all"); // viewing all books' info
16            cout << "________________________________________________" << endl;
17            displaySubMenu();
18            break;
19        case 2:
20            library.viewBookList("a"); // viewing available books' info
21            cout << "________________________________________________" << endl;
22            displaySubMenu();
23            break;
24        case 3:
25            library.viewBookList("ua"); // viewing unavailable books' info
26            cout << "________________________________________________" << endl;
27            displaySubMenu();
28            break;
29        case 4:
30            cout << "________________________________________________" << endl;
31            displayMainMenu();
32            break;
33        default:
34            cout << "Invalid number! " << endl
35            << "Choose again." << endl;
36            cin.clear();
37            cin.ignore(numeric_limits<streamsize>::max(), '\n');
38            displaySubMenu();
39            break;
40    }
41 }
```

Fig(1.5)

If we choose the number 1, we will see all of the books in the library like in Fig(1.6). Numbers 2 and 3 will work the same as number 1. Number 2 will only show the available books in the library while number 3 does ² the opposite. It will show the unavailable books. The code behind this feature can be seen in Fig(1.7) and Fig(1.8).

View all books	
ID-----	-----1
Title-----	To_Kill_a_Mockingbird
Author-----	Harper_Lee
Category-----	Fiction
Status:-----	unavailable
+-----+	
ID-----	-----2
Title-----	The_Great_Gatsby
Author-----	Scott_Fitzgerald
Category-----	_Fiction
Status:-----	unavailable
+-----+	
ID-----	-----3
Title-----	A_Brief_History_of_Humankind
Author-----	Noah_Harari
Category-----	History
Status:-----	unavailable
+-----+	
ID-----	-----4
Title-----	The_Alchemist
Author-----	Paulo_Coelho
Category-----	Inspirational
Status:-----	unavailable
+-----+	
ID-----	-----5
Title-----	The_Hunger_Games
Author-----	Suzanne_Collins
Category-----	Fiction
Status:-----	unavailable
+-----+	

Fig(1.6)

```
1 // methods definition outside Library class
2 void Library::viewBookList(string type) // display books' info according to type
3 {
4     string header, temp[5], tmpStr, line;
5
6     if (type == "all")
7         header = "all";
8     else if (type == "a")
9         header = "available";
10    else if (type == "ua")
11        header = "unavailable";
12
13    cout << endl
14        << "-----" << endl
15        << "View " << header << " books" << endl
16        << "-----" << endl;
17
18    ifstream bookFile("book.txt"); // open book file
19
20    if (bookFile.is_open())
21    {
22        while (getline(bookFile, line))
23        {
24            stringstream ss(line);
25            int i = 0;
26
27            while (getline(ss, tmpStr, ' '))
28            {
29                temp[i] = tmpStr;
30                i++;
31            }
32
33            if (type == temp[4] || type == "all") // display book's info base on type
34            {
35                Book book;
36                book.setId(temp[0]);
37                book.setTitle(temp[1]);
38                book.setAuthor(temp[2]);
39                book.setCategory(temp[3]);
40                book.setStatus(temp[4]);
41                book.displayInfo();
42            }
43        }
44        bookFile.close(); // close book file
45    }
46    else
47        cout << "Cannot open file!" << endl;
48 }
```

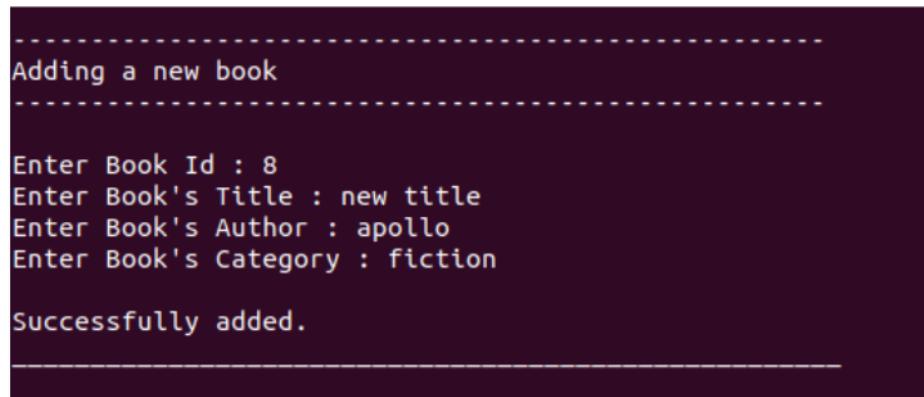
Fig(1.7)



```
● ● ●
1 // methods definition outside Book class
2 void Book::displayInfo()
3 {
4     // output book info
5     cout << endl
6     << setfill('-')
7     << "+" << right << setw(51) << "+" << endl
8     << left << setw(20) << "| ID" << right << setw(30) << id << " |" << endl
9     << left << setw(20) << "| Title" << right << setw(30) << title << " |" << endl
10    << left << setw(20) << "| Author" << right << setw(30) << author << " |" << endl
11    << left << setw(20) << "| Category" << right << setw(30) << category << " |" << endl
12    << left << setw(20) << "| Status:" << right << setw(30) << getStatus() << " |" << endl
13    << "+" << right << setw(51) << "+" << endl;
14 }
```

Fig(1.8)

If we click number 2 of the main menu, the program will ask for the data for a new book. As we can see in Fig(1.9).



```
-----  
Adding a new book  
-----  
  
Enter Book Id : 8  
Enter Book's Title : new title  
Enter Book's Author : apollo  
Enter Book's Category : fiction  
  
Successfully added.  
-----
```

Fig(1.9)

If we add the id that already exists, the program will show the warning message `duplicate` and ask for the id again as shown in Fig (1.10). We can see the code for this feature in [Fig \(1.11\)](#) and [Fig\(1.12\)](#).

```
-----  
Adding a new book  
-----  
  
Enter Book Id : 1  
Duplicated ID found! Enter the Id again.  
  
Enter Book Id : █
```

Fig(1.10)

```
1 void Library::addNewBook() // for storing new book
2 {
3     string id, title, author, category;
4
5     cout << endl
6         << "-----" << endl
7     << "Adding a new book" << endl
8     << "-----" << endl;
9
10    // check duplicated id
11    do
12    {
13        bool control = true;
14        string line;
15        cout << endl;
16        cout << "Enter Book Id : ";
17        cin >> id;
18        ifstream file("book.txt"); // open book file to check duplicated id
19        while (getline(file, line))
20        {
21            if (line.find(id) != string::npos) // id found
22            {
23                cout << "Duplicated ID found!"
24                    << " Enter the Id again." << endl;
25                control = false;
26                break;
27            }
28        }
29        if (control) // control will still be true if duplicated id is not found
30            break; // break the loop
31    } while (true);
32
33    cin.ignore(numeric_limits<streamsize>::max(), '\n');
34    cout << "Enter Book's Title : ";
35    getline(cin, title);
36    for (int i = 0; i < title.length(); i++) // rename title to snake case
37    {
38        if (title[i] == ' ')
39            title[i] = '_';
40    }
41
42    cout << "Enter Book's Author : ";
43    getline(cin, author);
44    for (int i = 0; i < author.length(); i++) // rename author to snake case
45    {
46        if (author[i] == ' ')
47            author[i] = '_';
48    }
49    cout << "Enter Book's Category : ";
50    cin >> category;
51
52    // set book infos using setter methods
53    Book book;
54    book.setId(id);
55    book.setTitle(title);
56    book.setAuthor(author);
57    book.setCategory(category);
58
59    if (book.store()) // book.store() will return true on success, otherwise false
60        cout << endl
61            << "Successfully added." << endl;
62    else
63        cout << endl
64            << "Failed to store new book!" << endl;
65 }
```

Fig(1.11)

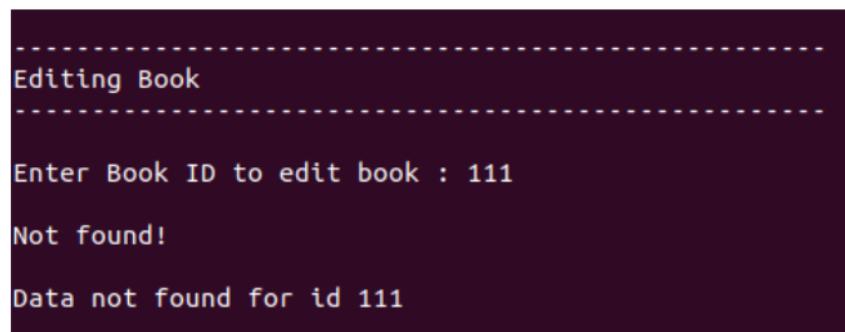
(12)



```
1 bool Book::store() // return true on success, false on fail
2 {
3     try
4     {
5         ofstream bookFile("book.txt", ios::app); // open book.txt file using ofstream
6         if (!bookFile.is_open())
7         {
8             string err_message = "Could not open file!";
9             throw(err_message); // throw exception if file can't open
10        }
11        // write new book's info to file
12        bookFile << id << " " << title
13            << " " << author << " " << category << " " << status << endl;
14
15        bookFile.close(); // file close
16        return true;
17    }
18    catch (string error_message)
19    {
20        cerr << endl
21            << error_message << endl; // display error message
22        return false;
23    }
24 }
```

Fig(1.12)

Number 3 is for editing the data of the book using book id. Firstly, we need to input the id of the book that we want to edit. So, the program will find the book using the id that we provided. If we enter the id which does not exist in the library, the program will say a 'Not Found' message as we can see in Fig (1.13).



```
-----
Editing Book
-----
Enter Book ID to edit book : 111
Not found!
Data not found for id 111
```

Fig(1.13)

If we enter the invalid input for the book's status in editing, the program will print a message and ask for the input again. That feature is shown in Fig (1.14). The code that works behind this feature is shown in Fig (1.15) and Fig (1.16).

```
-----  
Editing Book  
-----  
  
Enter Book ID to edit book : 1  
  
We found the following data.  
+-----+  
| ID-----1 |  
| Title-----To_Kill_a_Mockingbird |  
| Author-----Harper_Lee |  
| Category-----Fiction |  
| Status:-----unavailable |  
+-----+  
  
Choose the number below to edit.  
  
Choose 1 to edit title.  
Choose 2 to edit author.  
Choose 3 to edit category.  
Choose 4 to edit status.  
Choose 5 to return to main menu.  
  
Enter number : 4  
  
Enter new status ('a' for available / 'ua' for unavailable) : fd  
Invalid Input! Choose again.  
Enter new status ('a' for available / 'ua' for unavailable) :
```

Fig(1.14)

```
● ● ●  
1 void Library::editBook()  
2 {  
3     string id;  
4     Book book;  
5     cout << endl  
6         << "-----" << endl  
7         << "Editing Book" << endl  
8         << "-----" << endl;  
9     cout << endl  
10        << "Enter Book ID to edit book : ";  
11     cin >> id;  
12     if (!book.edit(id)) // book.edit(id) will return true on success, otherwise false  
13         cout << endl  
14             << "Data not found for id " << id << endl;  
15 }
```

Fig(1.15)

```

1  book Book(string ID) // file and edit book info // return true or success otherwise false
2  {
3      string line, temp;
4      ifstream fin("book.txt");
5      ofstream fout("book.txt");
6      book current = book();
7
8      if (file.is_open())
9      {
10         while (getline(fin, line)) // get books by each line
11         {
12             stringstream ss(line);
13             ss >> ID;
14             ss >> title;
15             ss >> author;
16             ss >> category;
17             ss >> price;
18             ss >> count;
19             ss >> date;
20
21             if (tempID == ID) // id is found
22             {
23                 this->ID = tempID;
24                 this->title = tempTitle;
25                 this->author = tempAuthor;
26                 this->category = tempCategory;
27                 this->price = tempPrice;
28                 this->count = tempCount;
29
30                 cout << endl
31                 << "The book id following data is modified";
32                 cout << endl;
33             }
34
35             if (tempID != ID) // id is found
36             {
37                 this->ID = tempID;
38                 this->title = tempTitle;
39                 this->author = tempAuthor;
40                 this->category = tempCategory;
41                 this->price = tempPrice;
42
43                 cout << endl
44                 << "The book id following data is added";
45                 cout << endl;
46
47                 cout << "Choose 1 to add title." << endl;
48                 cout << "Choose 2 to edit author." << endl;
49                 cout << "Choose 3 to edit category." << endl;
50                 cout << "Choose 4 to edit price." << endl;
51                 cout << "Choose 5 to return to main menu." << endl;
52                 cout << endl;
53                 cout << "Enter your choice : ";
54                 cin >> selected;
55                 cout << endl;
56                 switch (selected)
57                 {
58                     case 1:
59                         cout << "Enter new title : ";
60                         cin >> tempTitle;
61                         persistency.setBookTitle(tempTitle, "txt");
62                         persistency.write();
63                         for (int i = 0; i < file.length(); i++) // change the title in each row
64                         {
65                             if (file[i] == ' ')
66                             {
67                                 if (selected == 1)
68                                     file[i] = tempTitle[i];
69                             else
70                                 file[i] = ' ';
71                         }
72                     }
73                     break;
74                 }
75                 cout << endl;
76                 cout << "Entered new author : ";
77                 cin >> tempAuthor;
78                 persistency.setBookAuthor(tempAuthor, "txt");
79                 persistency.write();
80                 for (int i = 0; i < file.length(); i++) // change the author in each row
81                 {
82                     if (file[i] == ' ')
83                     {
84                         if (selected == 2)
85                             file[i] = tempAuthor[i];
86                         else
87                             file[i] = ' ';
88                     }
89                 }
90                 cout << endl;
91                 cout << "Entered new category : ";
92                 cin >> tempCategory;
93                 persistency.setBookCategory(tempCategory, "txt");
94                 persistency.write();
95                 for (int i = 0; i < file.length(); i++) // change the category in each row
96                 {
97                     if (file[i] == ' ')
98                     {
99                         if (selected == 3)
100                             file[i] = tempCategory[i];
101                         else
102                             file[i] = ' ';
103                     }
104                 }
105                 cout << endl;
106                 cout << "Entered new price : ";
107                 cin >> tempPrice;
108                 persistency.setBookPrice(tempPrice, "txt");
109                 persistency.write();
110                 for (int i = 0; i < file.length(); i++) // change the price in each row
111                 {
112                     if (file[i] == ' ')
113                     {
114                         if (selected == 4)
115                             file[i] = tempPrice[i];
116                         else
117                             file[i] = ' ';
118                     }
119                 }
120                 cout << endl;
121                 cout << "The value is successfully updated" << endl;
122                 cout << endl;
123             }
124         }
125     }
126
127     cout << endl;
128     cout << "Book list has been updated" << endl;
129     cout << endl;
130 }

```

Fig(1.16)

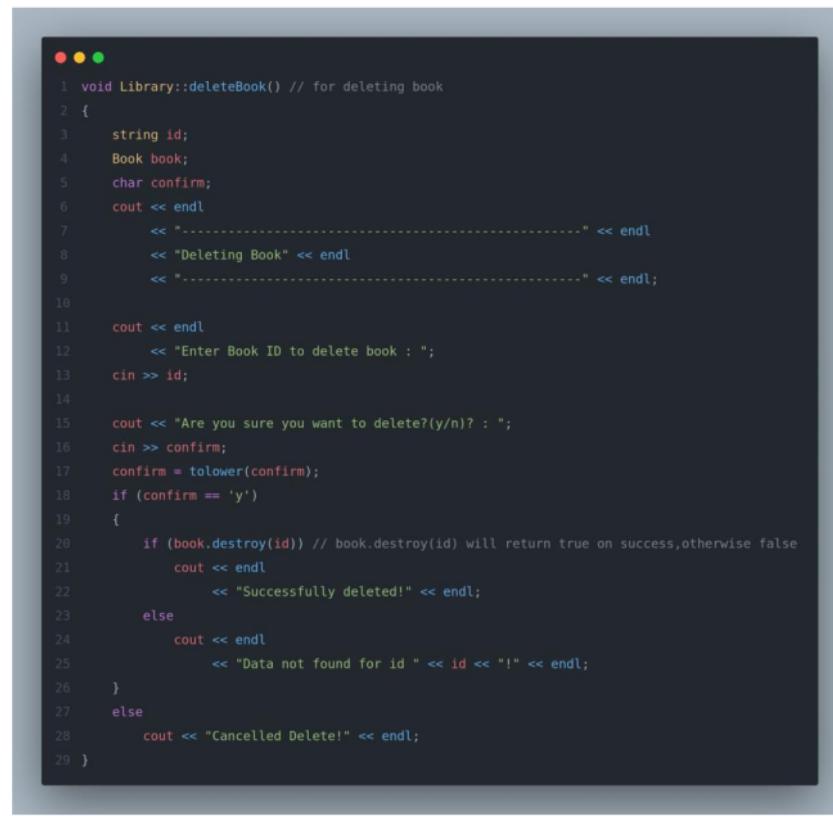
Number 4 is for deleting a book using its id. Firstly, we need to provide the id of the book that we want to delete. So, the program will find the book using the id that we provided. The program will also print the confirmation message which asks the user whether he really wants to delete. If we enter the id that does not exist in the library, the program⁶ will say a 'Not Found' message. Otherwise, it will be successfully deleted. For this feature, Fig (1.17), Fig (1.18), Fig(1.19), and Fig (1.20) are shown below.

```
-----  
Deleting Book  
-----  
  
Enter Book ID to delete book : 111  
Are you sure you want to delete?(y/n)? : y  
  
Not found  
  
Data not found for id 111!  
-----
```

Fig(1.17)

```
-----  
Deleting Book  
-----  
  
Enter Book ID to delete book : 8  
Are you sure you want to delete?(y/n)? : y  
  
Successfully deleted!  
-----
```

Fig(1.18)



```
1 void Library::deleteBook() // for deleting book
2 {
3     string id;
4     Book book;
5     char confirm;
6     cout << endl
7         << "-----" << endl
8         << "Deleting Book" << endl
9         << "-----" << endl;
10
11    cout << endl
12        << "Enter Book ID to delete book : ";
13    cin >> id;
14
15    cout << "Are you sure you want to delete?(y/n)? : ";
16    cin >> confirm;
17    confirm = tolower(confirm);
18    if (confirm == 'y')
19    {
20        if (book.destroy(id)) // book.destroy(id) will return true on success, otherwise false
21            cout << endl
22                << "Successfully deleted!" << endl;
23        else
24            cout << endl
25                << "Data not found for id " << id << "!" << endl;
26    }
27    else
28        cout << "Cancelled Delete!" << endl;
29 }
```

Fig(1.19)

```
1 bool Book::destroy(string id) // find and delete book with id // return true on success, otherwise false
2 {
3     string line, book_id;
4     fstream file("book.txt", ios::in | ios::out);
5
6     if (!file.is_open())
7     {
8         cerr << endl
9         << "Could not open file!" << endl;
10    return false;
11 }
12
13 while (getline(file, line))
14 {
15     stringstream ss(line);
16
17     getline(ss, book_id, ' ');
18
19     if (book_id == id) // id is found
20    {
21        file.close();
22
23        // create and open temporary file
24        ofstream outfile;
25        file.open("book.txt");
26        outfile.open("temp.txt", ios::app);
27
28        while (getline(file, line))
29            if (line.find(id) == string::npos) // rewriting existing data expect the given id
30                outfile << line << endl;
31
32        outfile.close();
33        file.close();
34        remove("book.txt");           // delete book file
35        rename("temp.txt", "book.txt"); // rename temp.txt to book.txt
36
37        return true;
38    }
39 }
40 // if the search id is not found, the code will reach here, and print 'Not found' and return false
41 cout << endl
42 << "Not found" << endl;
43 return false;
44 }
```

Fig (1.20)

To exit the program, choose number 5. The program will print the goodbye message to the user with his name which was provided at the start of the program as shown in Fig (1.21).

```
→Display book list sub menu (Choose one number).

1. View all books in library
2. View all available books in library
3. View all unavailable books in library
4. Return to main menu

Choose a number : 4

→Main Menu (Choose one number).

1. View book List
2. Add new book
3. Edit book information
4. Delete book
5. Exit the program

Choose a number : 5

Good Bye, Naing Min Khant. See you again.
```

Fig (1.21)

The code behind this feature is shown in Fig (1.22).

```
● ● ●

1 Menu::~Menu() // destructor definition outside class
2 {
3     cout << endl
4     << "Good Bye, " << guest_name << ". See you again." << endl;
5 }
```

Fig (1.22)

Task 2 - Inheritance

One class cannot access the properties and methods of another class without a relationship, although it can be done by instantiating an object of another. In this case, Inheritance can be used as a relationship between classes. Inheritance is one of the most important features of Object-Oriented Programming. It is just like a Parent-Child relationship which means Child has the ability to derive the parent's possession. By implementing Inheritance, the derived class(child class) will have the ability to access the properties of the base class (parent class). However, the members of the base class can be controlled by the access specifiers and will not be affected by the derived class. The derived class now is said to be inherited from the base class.

8

There are basically three types of Inheritance in C++, which are Single Inheritance, Multi-Level Inheritance, and Multiple Inheritance. If a derived class has only one base class(parent class), it can be called Single Inheritance. One derived class can inherit from more than one class, and it is known as Multiple Inheritance. If a derived class is created from another derived class, it is said to be Multi-Level Inheritance.

I implement the Single Inheritance in the program. Implementation of Single Inheritance in the program can be seen in Fig (2.1) and Fig (2.2).

```
● ● ●
1 class Menu
2 {
3 public:
4     Menu(); // constructor declaration
5     ~Menu(); // destructor declaration
6 protected:
7     string guest_name;
8     int selected_menu_no;
9     void displayMenuList(const string menus[], int size);
10 };
11
12 Menu::Menu() // constructor definition outside class
13 {
14     cout << "Please Enter your name : ";
15     getline(cin, guest_name);
16     cout << endl
17         << "-----" << endl
18         << "Hello, " << guest_name << ". Welcome to our library." << endl
19         << "-----" << endl;
20 }
21 Menu::~Menu() // destructor definition outside class
22 {
23     cout << endl
24     << "Good Bye, " << guest_name << ". See you again." << endl;
25 }
```

Fig (2.1)

In this figure, the Menu class, which has a constructor, destructor, and three protected members, is the base class (parent class).



```
● ● ●
1 class LibraryMenu : public Menu
2 {
3     const string MAIN_MENU[5] = {"1. View book List",
4                                 "2. Add new book",
5                                 "3. Edit book information",
6                                 "4. Delete book",
7                                 "5. Exit the program"};
8     const string SUB_MENU[4] = {"1. View all books in library",
9                                "2. View all available books in library",
10                               "3. View all unavailable books in library",
11                               "4. Return to main menu"};
12     Library library;
13
14 public:
15     void displayMainMenu();
16
17     void displaySubMenu();
18 }
```

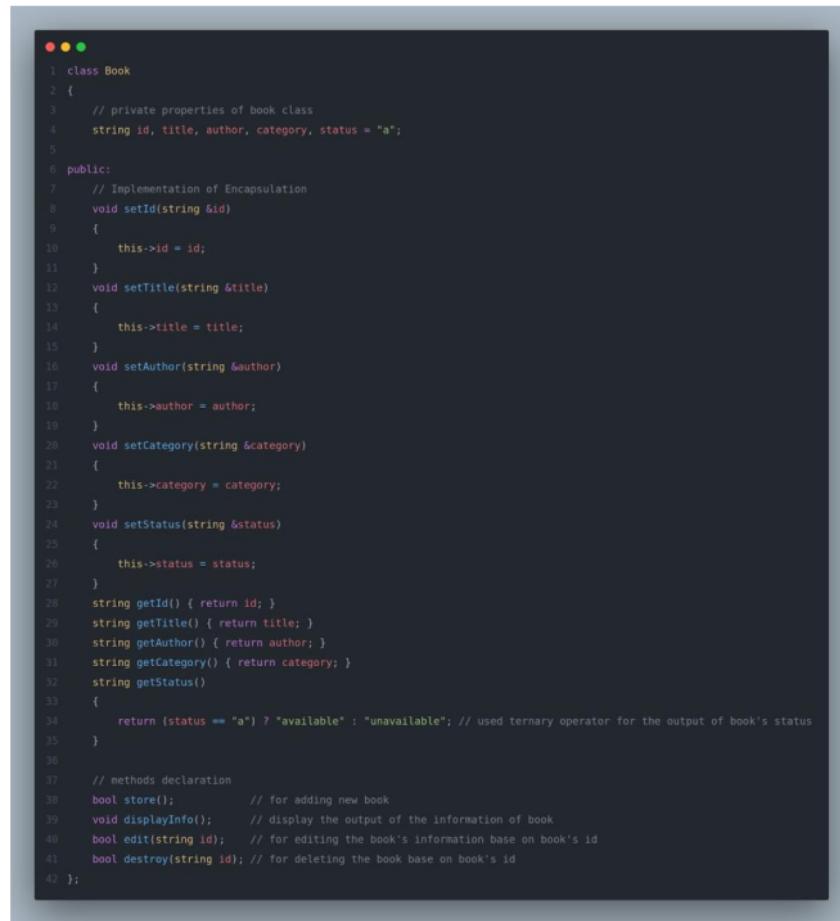
Fig (2.2)

In Fig (2.2), the Library class is said to be inherited publicly from the Menu class (the base class). The Library class can, now, have access to manipulate the protected members of the Menu class.

Task 3 - Encapsulation

7
Encapsulation is one of the features of Object-Oriented Programming. Encapsulation can be defined as setting the data of a class as private properties and manipulating them by using class functions. Those class functions can only be accessed by an object of its class and must use only member variables. If there is no function inside the class that is using the member variable of the class, it cannot be called encapsulation. By using encapsulation, the security of class data can be increased and it also helps to control the modification of class data members.

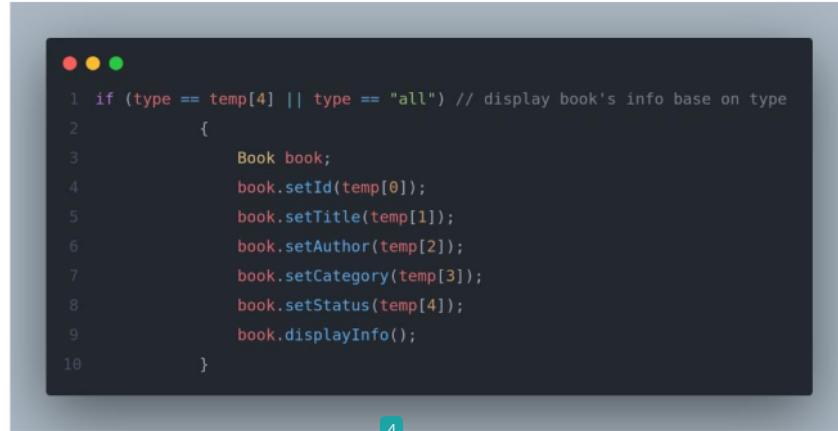
4
Implementation of Encapsulation, which is used in the program, is shown in Fig (3.1) and Fig (3.2).



```
1 class Book
2 {
3     // private properties of book class
4     string id, title, author, category, status = "a";
5
6 public:
7     // Implementation of Encapsulation
8     void setId(string &id)
9     {
10         this->id = id;
11     }
12     void setTitle(string &title)
13     {
14         this->title = title;
15     }
16     void setAuthor(string &author)
17     {
18         this->author = author;
19     }
20     void setCategory(string &category)
21     {
22         this->category = category;
23     }
24     void setStatus(string &status)
25     {
26         this->status = status;
27     }
28     string getId() { return id; }
29     string getTitle() { return title; }
30     string getAuthor() { return author; }
31     string getCategory() { return category; }
32     string getStatus()
33     {
34         return (status == "a") ? "available" : "unavailable"; // used ternary operator for the output of book's status
35     }
36
37 // methods declaration
38 bool store();           // for adding new book
39 void displayInfo();     // display the output of the information of book
40 bool edit(string id);   // for editing the book's information base on book's id
41 bool destroy(string id); // for deleting the book base on book's id
42 };
```

Fig (3.1)

The Book class has the private properties and those can only be manipulated by the getters and setters functions of the class.



A screenshot of a terminal window displaying Java code. The code is a snippet from a program that handles book information. It includes an if statement to check if the type is equal to temp[4] or "all". If true, it creates a new Book object, sets its ID, title, author, category, and status using setter methods, and then calls the displayInfo() method. The code is numbered from 1 to 10. A small number '4' is visible at the bottom center of the terminal window.

```
1 if (type == temp[4] || type == "all") // display book's info base on type
2 {
3     Book book;
4     book.setId(temp[0]);
5     book.setTitle(temp[1]);
6     book.setAuthor(temp[2]);
7     book.setCategory(temp[3]);
8     book.setStatus(temp[4]);
9     book.displayInfo();
10 }
```

Fig (3.2)

Fig (3.2) shows the private members of the Book class are being set by the setters methods of its class.

COS 212 Assignment

ORIGINALITY REPORT



PRIMARY SOURCES

- | | | |
|---|---|----|
| 1 | Li Zheng, Yuan Dong, Fang Yang. "7. Inheritance And Derivation", Walter de Gruyter GmbH, 2019
Publication | 4% |
| 2 | etheses.whiterose.ac.uk
Internet Source | 3% |
| 3 | Ivor Horton's Beginning ANSI C++ The Complete Language, 2004.
Publication | 2% |
| 4 | digital.lib.washington.edu
Internet Source | 2% |
| 5 | data.epo.org
Internet Source | 2% |
| 6 | www.joiebaby.com
Internet Source | 2% |
| 7 | Kishori Sharan, Adam L. Davis. "Chapter 20 Inheritance", Springer Science and Business Media LLC, 2022
Publication | 2% |
| | csharprescue.wordpress.com | |

8

Internet Source

1 %

9

uhra.herts.ac.uk

Internet Source

1 %

Exclude quotes Off

Exclude bibliography Off

Exclude matches Off