

Michal Pfeiffer OOP Assignment

by Michal Pfeiffer

Submission date: 09-Jan-2024 06:57PM (UTC+0000)

Submission ID: 221238095

File name: 40136_Michal_Pfeiffer_Michal_Pfeiffer_OOP_Assignment_532657_725355241.docx (456.69K)

Word count: 2459

Character count: 10905

Advanced Software Development – OOP Programming (C++) Assignment

By Michal Emil Pfeiffer (2204569) - Game Development (BSc)

Introduction	1
Task 1	1
Importance of OOP	2
Modularity	2
Inheritance	2
Polymorphism	2
Conclusion	2
Video Demo	3
User Guide to the code	3
Code Testing.....	6
Task 2	7
References	8
Code used:	8

Introduction

In this assignment, I am tasked to create an application that uses object-oriented programming (OOP) and demonstrate its uses, like inheritance and polymorphism within the code (That can be seen below, past references), and show how the code functions with a video demonstration. For the second task, I need to show that I have undertook the course of “C++Essentials” and prove that I have completed it with the appropriate certification.

Task 1

In this task, I have created a functional program, and I will go step by step on how the code works, testing certain parts of the code and documenting them and the importance of OOP when coding.

Importance of OOP

Object-Oriented Programming has many uses for enhancing the way the code is programmed and used, mainly being more flexible than the traditional method of top-down [2]. I shall now describe some of the key features of Object-Oriented Programming;



Modularity

With the modularity of OOP, each class is broken up into building blocks of separate functions within the class that make it up. For example, when trying to trouble shoot the program and you know where the issue happens in 'main' and if that problem turns out to be a function from the created class, it is an easy fix of going to that function and checking what went wrong with it [2].

This can also be seen in my code as well, the use of different functions that all have something going on that has a chance of going wrong, if you are able to track down the problem to the root cause of being either a wrong variable called or incorrect calculations, it easily fixable.

Inheritance

Another key aspect of using OOP, when creating another class that has a lot of similarities with another class, for example; car and a racecar, they are both cars, but the racecar has some functions that are not commonly found with other cars. Instead of creating another class called 'racecar' and rewriting all the same functions over to the new class, it can instead inherit the same functions that the 'car' class uses, essentially making the 'racecar' class a subclass. [2]

This is also used in my code, with the 'User' class and the 'Librarian' class, where the Librarian class is inheriting the functions and variables of the User class without the need of rewriting all the functions into another class. Inheritance is very useful to master, as it can save a lot of time in coding up new classes to be used that also use the same functions.

Polymorphism

Polymorphism grants additional flexibility within OOP, as you can reuse the same name of a function, let us use 'read' as an example, instead of having other function be 'read2' or 'read3' for different classes/objects or different functionalities, we can just reuse the same name of the function [2] but instead of passing one argument through to the function, we can instead pass two, three or more variables. For example; we can have 'read(book1)' in 'main' and we can also have 'read(book1, book2)', this is an example of using polymorphism, mainly the use of function overloading.

I have not used much of polymorphism within my code, as there were not many benefits of doing so but the use polymorphism can be very useful when dealing with multiple variables that need to be in a certain function (as with function overloading).

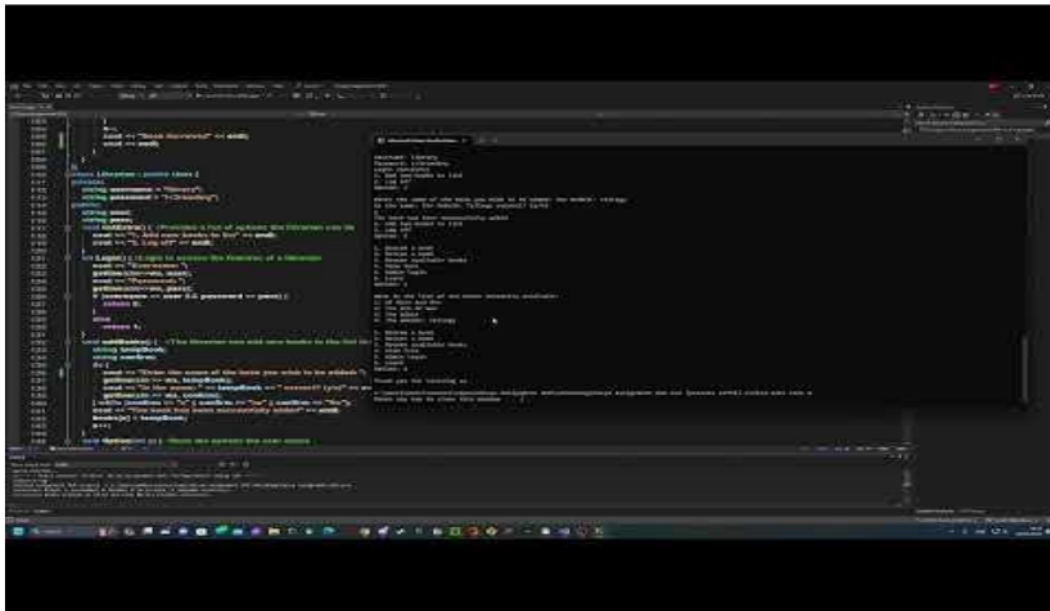
Conclusion

To conclude this section, polymorphism is the go-to method when it comes down to programming new software, as the versatility and flexibility of OOP far surpasses the older method of top-down, as you can easily reuse certain functions without the need of them being rewritten, being far easier to troubleshoot and just simply being easier to work with.

Video Demo

Here is a link to a video demonstrating the code, the code used can be viewed towards the end of the document:

YouTube Link: <https://youtu.be/wOT7wRq7OZw> [1]



User Guide to the code

When using the code, I advise that mostly integer values are used when handling the code, as the user does not want unwanted errors when going through the menus.

A simple menu will greet the user, prompting them to input an **integer** for their option of choice;

```
Welcome to the library service!  
What will you like to do?  
1. Borrow a book  
2. Return a book  
3. Browse available books  
4. View fees  
5. Admin login  
6. Leave  
Option: |
```

As long as the user inputs an **integer** when selecting an option on the menu, everything will work fine, and the rest of the code is self-explanatory. For example, borrowing a book;

```
Option: 1

Here is the list of the books currently available:
1: Da Vinci Code
2: Of Mice And Men
3: The Bible
4: The Art of War
5: Lord of the Rings: Trilogy

Enter the book number you wish to borrow: 3
Book borrowed
```

If the user inputs the appropriate input for the main menu (being '1' to borrow a book), then they will have a menu of available books that can be borrowed and are prompted to choose one from the selection. Selecting '3' will borrow 'The Bible' and remove it from the list of available books.

Moving on, if the user using this is a librarian and they know the login credentials of 'library' for the username and 'i<3reading' for the password (yes, very original for a library management system), then they would pick option '5' on the list in the main menu;

```
1. Borrow a book
2. Return a book
3. Browse available books
4. View fees
5. Admin login
6. Leave
Option: 5

Username:
```

The "user" will be prompted to enter the username and then the password, like so;

```
Username: library
Password: i<3reading
Login succesful
```

If the credentials are correct, then the login is successful and then they are greeted with the following menu;

```
1. Add new books to list
2. Log off
Option:
```

It is indeed small, but the librarian can add new books to the list of available books that can be borrowed by regular users, if the user inputs incorrect credentials, then they are booted back to the main menu;

```
Username: lib
Password: read
Username or password is incorrect
1. Borrow a book
2. Return a book
3. Browse available books
4. View fees
5. Admin login
6. Leave
Option: |
```

Otherwise, when the librarian is adding a new book to be added onto the list, they are prompted with inputting in the name of the book;

```
1. Add new books to list
2. Log off
Option: 1

Enter the name of the book you wish to be added: |
```

The librarian can add any book they desire (in reality, that cannot do so) by just typing out its name, for example, 'The Hobbit: Trilogy';

```
Enter the name of the book you wish to be added: The Hobbit: Trilogy
Is the name: The Hobbit: Trilogy correct? (y/n): y
The book has been successfully added
```

There is an additional prompt to ensure that the name written is correct, if the name is correct, the user needs to input either "y", "yes" or "Yes" to confirm the name is correct, otherwise input "n", "no" or "No" if it is incorrect and then retype the name of the book to be added.

The book added can be viewed by any user using the third option at the main menu;

```
1. Borrow a book
2. Return a book
3. Browse available books
4. View fees
5. Admin login
6. Leave
Option: 3

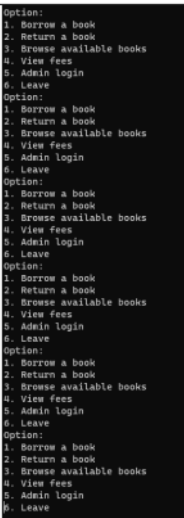
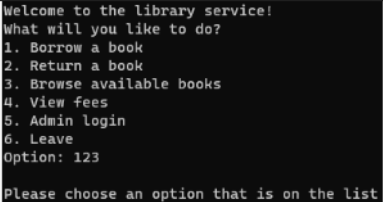
Here is the list of the books currently available:
1: Da Vinci Code
2: Of Mice And Men
3: The Art of War
4: Lord of the Rings: Trilogy
5: book
6: The Hobbit: Trilogy
```



As we can see, the book has been added successfully and can be borrowed by other users. This will also wrap up on how to use the code in its intended way.

Code Testing

Here I will be documenting tests that I have done on my code;

Test	Result (Image)	Conclusion
1) Typing in a letter for selecting an option on the list.		Code loops infinitely, this happens at any part of the code where the user needs to input a number for one of the options, even when selecting a book. Should implement a way to check if the input contains any letters or use a different variable and run checks for specific inputs.
2) Typing in a decimal (float) value for selecting an option on the list.		Once again, it loops infinitely as with the error above. A solution to this will most likely be added automatically with the first solution.
3) Typing in a value that is out of range (integer) for selecting an option on the list.		Handled it correctly, prompts the user to select an option that is on the list rather than some random number.

4) Typing in different characters for confirming a new addition of a book title	<pre>Enter the name of the book you wish to be added: book Is the name: book correct? (y/n): 123 The book has been successfully added Enter the name of the book you wish to be added: book Is the name: book correct? (y/n): good The book has been successfully added</pre>	Any string of characters that are not solely "n", "no" or "No" will confirm the book title. I have tried having the check be 'while(confirm != "y"...)' but ends being an endless loop with no actual exit. Will have to investigate it further.
--	---	--

Task 2

This is the certification I have gotten from completing the course "C++Essentials" on Cisco Netacad:



CPA: Programming Essentials in C++

During the Cisco Networking Academy® self-paced course, the student has studied the following skills:

- the universal concepts of computer programming,
- developer tools,
- the syntax and semantics of the C++ language, as well as basic data types in the C++ language,
- the principles of the object-oriented model and its implementation in the C++ language,
- the means by which to resolve typical implementation problems with the aid of standard C++ language libraries,
- the writing of C++ programs using standard language infrastructure, regardless of the hardware or software platform.

This Statement of Achievement is to acknowledge that during the course **CPA: Programming Essentials in C++**, the student has been able to accomplish coding tasks related to the basics of programming in the C++ language, and understands the fundamental notions and techniques used in object-oriented programming.

By completing the course, the student is now ready to attempt the qualification **CPA – C++ Certified Associate Programmer Certification**, from the C++ Institute.

Maciej Wichary
VP - OpenEDG C++ Institute

Michał Pfeiffer

Student

7 Jan 2024

Date

References

[1] YouTube, *OOP Assignment – Video Demonstration*, SlavikCrusader96, Released: 8 Jan 2024. [Online Video] Available: <https://youtu.be/wOT7wRq7OZw> [Created by me on 8 Jan 2024]

[2] Rober Half, "4 Advantages of Object-Oriented Programming", *Robert Half*, 5 Jan 2023. [Online] Available: <https://www.roberthalf.com/us/en/insights/career-development/4-advantages-of-object-oriented-programming> [Accessed: 9 Jan 2024]

Code used:

```
#include <iostream>

#include <string>

using namespace std;

class User {
public:
    int n = 5;
    int m = 0;
    string books[100] = { "Da Vinci Code", "Of Mice And Men", "The Bible", "The Art of War", "Lord of the Rings: Trilogy" };
    string borrowed[100];
    void list() {    //Provides a list of options
        cout << "1. Borrow a book" << endl;
        cout << "2. Return a book" << endl;
        cout << "3. Browse available books" << endl;
        cout << "4. View fees" << endl;
        cout << "5. Admin login" << endl;
        cout << "6. Leave" << endl;
    }
    void borrowBook() {    //Provides a list of books that the user can borrow
```

```

int borrow;
viewBooks();
if (books[0] == "") { //Checks if there are books available to be borrowed
cout << endl;
}
else {
cout << "Enter the book number you wish to borrow: ";
cin >> borrow;
borrow -= 1;
if (borrow < 0 || borrow > n) {
cout << "Please select the book on the list\nReturning to menu...\n" << endl;
}
else {
sort(borrow, 2);
}
}
}

void returnBook() { //Allows the user to return a borrowed book
if (borrowed[0] == "") { //Checks if the user has any books that can be returned
cout << "It appears that you have no books to return, come back when you have books that
need to be returned" << endl;
}
else {
cout << "Which book do you wish to return?" << endl;
for (int i = 0; i < m; i++) {
cout << i + 1 << ": " << borrowed[i] << endl;
}
}
}

```

```

cout << "Book: ";
int q;
cin >> q;
if (q - 1 > m || q - 1 < 0) {    //Checks if the input is valid
cout << "Please pick the book that is on the list\n\nReturning to menu..." << endl;
}
else {
q -= 1;
sort(q, 1);    //Sorts the list borrowed books, so that the list has no gaps
}
}
cout << endl;
}

void viewBooks() {    //User can look at available books
cout << "Here is the list of the books currently available:" << endl;
if (books[0] == "") {    //Checks if there are available books
cout << "Oops, looks like we have no books available, come back soon!" << endl;
}
else {
for (int i = 0; i < n; i++) {
cout << i + 1 << ": " << books[i] << endl;
}
}
cout << endl;
}

void viewFees() {    //Shows the fees of borrowing books
cout << "Our current fees are \x9c" << "1.50 per book borrowed" << endl;
}

```

```

if (borrowed[0] == "") {      //Checks if the user has any borrowed books
    cout << "You have no borrowed books, so you have no fees" << endl;
}
else { //Shows the fees if you have any borrowed books
    cout << "Your current fees will be: ";
    float fee = 1.5, totalFee;
    totalFee = m * fee;
    cout << "\n%c" << totalFee << endl;
    cout << "You will have to pay these fees if your books are not returned on time" << endl;
}
cout << endl;
}

void sort(int d, int id) {      //Sorter, so no gaps are present in the 'books' and 'borrowed'
    arrays
    if (id == 1) { //Borrowed book sorter
        books[n] = borrowed[d];
        n++;
        borrowed[d] = "";
        for (int i = d; i < m; i++) {
            borrowed[i] = borrowed[i + 1];
        }
        m--;
        cout << "Book returned" << endl;
        cout << endl;
    }
    else if (id == 2) { //Available books sorter
        borrowed[m] = books[d];

```

```

m++;
books[d] = "";
for (int i = d; i < n; i++) {
    books[i] = books[i + 1];
}
n--;
cout << "Book borrowed" << endl;
cout << endl;
}
}
};

class Librarian : public User {
private:
    string username = "library";
    string password = "i<3reading";
public:
    string user;
    string pass;
    void listExtra() {    //Provides a list of options the librarian can do
        cout << "1. Add new books to list" << endl;
        cout << "2. Log off" << endl;
    }
    int Login() {    //Login to access the features of a librarian
        cout << "Username: ";
        getline(cin>>ws, user);
        cout << "Password: ";
        getline(cin>>ws, pass);
    }
};

```

```

if (username == user && password == pass) {
    return 0;
}
else
    return 1;
}

void addBooks() {    //The librarian can add new books to the list that can be borrowed
    string tempBook;
    string confirm;
    do {
        cout << "Enter the name of the book you wish to be added: ";
        getline(cin >> ws, tempBook);
        cout << "Is the name: " << tempBook << " correct? (y/n): ";
        getline(cin >> ws, confirm);
    } while (confirm == "n" || confirm == "no" || confirm == "No");
    cout << "The book has been successfully added" << endl;
    books[n] = tempBook;
    n++;
}

void Option(int p) {    //Runs the options the user chose
    if (p < 0 || p > 6) {    //Check if the value is appropriate
        cout << "Please choose an option that is on the list" << endl;
    }
    else {
        switch (p) {
        case 1:
            borrowBook();

```

```
break;
case 2:
returnBook();
break;
case 3:
viewBooks();
break;
case 4:
viewFees();
break;
case 5:
int c;
c = Login();
if (c == 0) {    //Checks if the login was correct
cout << "Login succesful" << endl;
int r;
do {
listExtra();
cout << "Option: ";
cin >> r;
cout << endl;
Option(r, 1);
} while (r != 2);
}
else {
cout << "Username or password is incorrect" << endl;
}
```

```

break;
default:
break;
}
}
}

void Option(int r, int j) {    //Runs the options the librarian chose
if (r < 0 || r > 2) {    //Checks if the value is appropriate
cout << "Please choose an option that is on the list" << endl;
}
else {
switch (r) {
case 1:
addBooks();
break;
default:
break;
}
}
}
};

int main() {
int pick;
Librarian user; //Create class
cout << "Welcome to the library service!" << endl;
cout << "What will you like to do?" << endl;

```

```
do {    //Does the code until the user decides to leave
user.list();
cout << "Option: ";
cin >> pick;
cout << endl;
user.Option(pick);
} while (pick != 6);    //Input required to leave program
cout << "Thank you for visiting us" << endl;
return 0;
}
```

Michal Pfeiffer OOP Assignment

GRADEMARK REPORT

FINAL GRADE

71 /100

GENERAL COMMENTS

Hi Michal,

In-depth feedback is provided within the document, and a summary is presented here. Scores for individual sections are accessible through the rubric.

Areas of appreciation:

The inclusion of a video demonstration is a great addition.

The user guide is informative and walks the user through different scenarios, explaining how to interact with the code

Areas of improvement:

In the "Code Testing" section, it's mentioned that typing in a letter for selecting an option causes an infinite loop. Consider adding more robust error handling to prevent such issues and provide clear feedback to the user.

More details about the classes, object used to access the methods etc. should have been explained.

PAGE 1

PAGE 2



Comment 1

Good start by emphasizing on the importance of oop. however, explanation of key elements like classes , objects, access specifiers etc., should have been explained

PAGE 3

PAGE 4

PAGE 5



Comment 2

what is the difference between option 1 and 3?

screen shots for options 2 and 4 along with explanations could have been included

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

GRADING FORM: DC-ASD

MICHAL PFEIFFER

71

LITERATURE REVIEW

with IEEE referencing



2

KNOWLEDGE&UNDERSTANDING SKILLS

use of OOPs features



12

COGNITIVE&INTELLUCTUAL SKILLS

Testing &Error handling



12

PRACTICAL SKILLS

Working program Demo video



16

TRANSFERRABLE SKILLS

User Manual, document style



4

