

Nathan Willis Assignment

by Nathan Willis

Submission date: 10-Jan-2024 02:56PM (UTC+0000)

Submission ID: 221381055

File name: 41436_Nathan_Willis_Nathan_Willis_Assignment_532657_1971681100.docx (824.8K)

Word count: 2374

Character count: 12119

2024

Adv Software Development

PROGRAMMING ASSIGNMENT
NATHAN WILLIS (2103532)

Contents

Documentation **Error! Bookmark not defined.**

 Roadblocks..... **Error! Bookmark not defined.**

Object Orientation 2

 Classes 2

 Inheritance 2

 Polymorphism 2

Testing 2

 Class 1 tests..... 2

 Class 2 tests..... 8

 Class 3 tests..... 12

Video Link 14

References..... 24

1 Object Orientation

Classes

I implemented 3 different classes, user, staff and sysAdmin. The user class is the standard class accessible by all, it has all the standard functions in it. The staff class is only available to those who have the password, it has access to all the same functions as the user class but it also has access to other functions that allow it to alter the library structure. The sysAdmin class is different to the previous ones in that it does not have access to the standard functions but it does have access to the backup inventory file.

Inheritance

The staff class inherits the public members of the user class, this is very useful as it allows the staff to do everything the users do but it can be detrimental as the staff have access to functions which require the variable userNumber. This is not a major issue here as I have largely worked around it but it could come up if there was more editing done to the staff class.[1]

Polymorphism

The member function inventory has an overload where it takes a string input and returns a specific item from the library structure. This is useful as both these functions do similar things but could be confusing.

Testing

Class 1 tests

```
0.public access
1.staff access
2.ststems admin
3.quit
0
enter user number:
for testing purposes there are 10 user numbers going from 0-9
but in an actual implementation there would be many user numbers ideally non sequential and not 1 digit long
0
invalid input try again
-1
User does not exist
0.public access
1.staff access
2.ststems admin
3.quit
0
enter user number:
for testing purposes there are 10 user numbers going from 0-9
but in an actual implementation there would be many user numbers ideally non sequential and not 1 digit long
7
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
```

User class initialisation, tested for negative numbers and characters, no errors found.



```

0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
1
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight

```

Inventory function works as expected, requires no input so no param testing needed.

```

0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
2
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight

Which book would you like to take out? p
invalid input try again
9

```

Checkout function not working as expected, allowing user to borrow from empty items.

```

if (userNum > 0 && userNum < library.leng) {
    userNum--;
    users.bookName[userNumber] = library.bookName[userNum];
    bookRemove(userNum);
    users.borrowed[userNumber] = 1;
}
else {
    cout << "invalid entry" << endl;
}

```

Fixed with this if statement.

```
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
2
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight

Which book would you like to take out? 9
invalid entry
```

```
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight

Which book would you like to take out? -1
invalid entry
```

```
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight

Which book would you like to take out? 4
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
1
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight: breaking dawn
5. twilight: eclipse
6. twilight: new moon
7. twilight
```

Now works as expected.

```
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
3
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
2
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight: breaking dawn
5. twilight: eclipse
6. twilight: new moon
7. twilight
8. twilight
```

```
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
3
No book borrowed
```

Return function works as expected, no user inputs required so no parameter testing.

```
Which book would you like to take out? 0
invalid entry
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
4
book name: twilight
position 7
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
4
book name: p
position 10
```

Book check function not working as expected, showing very weird behaviour when entering a single character.

```
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
4
book name: p
8
```

Library has a length of 8 but I has a value of 9.



Was caused by:

```
while (i <= library.leng && library.bookName[i] != bName) {
    i++;
}
```

Should be:

```
while (i < library.leng && library.bookName[i] != bName) {
    i++;
}
```

```
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
4
book name: p
8
not in inventory
```

Now works as expected.

```
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
5
no outstanding books
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
2
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight

Which book would you like to take out? 4
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
5
borrowed: twilight
```

Borrow function works as expected.

Because of the way I've structured the code the library list reinitialises when you exit one class and enter another so I created a save function that runs when exiting a class.

```

1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight

Which book would you like to take out? 4
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
0
save?(y/n) y
0.public access
1.staff access
2.ststems admin
3.user guide
4.quit
0
enter user number:
for testing purposes there are 10 user numbers going from 0-9
but in an actual implementation there would be many user numbers ideally non sequential and not 1 digit long
1
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
1
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight: breaking dawn
5. twilight: eclipse
6. twilight: new moon
7. twilight

```

Save function works as expected.

Class 2 tests

```

0.public access
1.staff access
2.ststems admin
3.user guide
4.quit
y
invalid input try again
1
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
11
INCORRECT PASSWORD 2 MORE TRIES
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
222
INCORRECT PASSWORD 1 MORE TRIES
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
abc123

```

```
0.public access
1.staff access
2.ststems admin
3.user guide
4.quit
1
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
ABC123
INCORRECT PASSWORD 2 MORE TRIES
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
abc123
0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
```

Password entry works as expected.

```
0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
2
0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
1
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon

0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
```

Inventory sort works as expected.

```

0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
3
book name: twilight: new moon
0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
invalid input try again
1
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight:
0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file

```

Forgot to make sure book title entry allows spaces, changing the cin to a getline will fix this.

```

0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
3
book name: 0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
h h h
invalid input try again

```

Didn't work as expected.

Forgot the cin.ignore().

```
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
3
book name: twilight:  new moon
0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
1
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight:
9. twilight:  new moon
```

Works as expected.

```
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight:
9. twilight:  new moon

remove(int): 9
0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
1
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight:
```

Remove book function works as expected and was already tested in the book borrowing function so I will skip more thorough retests.

The staff book check function is inherited from the user class so I will skip testing it.

The save function is also inherited from the user class and has also already been tested.

Class 3 tests

```
0.public access
1.staff access
2.ststems admin
3.user guide
4.quit
1
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
a
INCORRECT PASSWORD 2 MORE TRIES
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
a
INCORRECT PASSWORD 1 MORE TRIES
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
a
INCORRECT PASSWORD 0 MORE TRIES

locked out please contact system admin
SYSTEMS ADMIN
enter password(123abc): 123
bricked
```

```
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
l
INCORRECT PASSWORD 2 MORE TRIES
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
l
INCORRECT PASSWORD 1 MORE TRIES
Enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
l
INCORRECT PASSWORD 0 MORE TRIES

Locked out please contact system admin
SYSTEMS ADMIN
enter password(123abc): 123abc
correct
```

Systems admin login works as expected.

```
0.public access
1.staff access
2.ststems admin
3.user guide
4.quit
2
SYSTEMS ADMIN
enter password(123abc): 123abc
correct
0.main menu
1.restore from backup
2.update backup
1
0.main menu
1.restore from backup
2.update backup
0
0.public access
1.staff access
2.ststems admin
3.user guide
4.quit
0
enter user number:
for testing purposes there are 10 user numbers going from 0-9
but in an actual implementation there would be many user numbers ideally non sequential and not 1 digit long
1
0.main menu
1.see inventory
2.checkout book
3.return book
4.book check(case sensitive)
5.borrow status
1
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight
```

Restore from backup works as expected.

Forgot to actually call the backup update function in the systems admin menu.

```
case 2:
    backupUpdate();
    break;
```

```

1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon
8. twilight

remove(int): 8
0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
7
save?(y/n) y
0.public access
1.staff access
2.systems admin
3.user guide
4.quit
7
SYSTEMS ADMIN
enter password(123abc): 123abc
correct
0.main menu
1.restore from backup
2.update backup
7
0.main menu
1.restore from backup
2.update backup
7
0.main menu
1.restore from backup
2.update backup
7
0.public access
1.staff access
2.systems admin
3.user guide
4.quit
7
enter password:
FOR TEST PURPOSES THE PASSWORD IS "abc123"
abc123
0.main menu
1.see inventory
2.sort inventory
3.add book
4.remove book
5.book check(case sensitive)
6.save changes to inventory file
7
1. A Star is Born
2. The Catcher in the Rye
3. The Vampire Diaries
4. twilight
5. twilight: breaking dawn
6. twilight: eclipse
7. twilight: new moon

```

Now works as expected.

Video Link

[Nathan Willis Code Video.mp4](#)

Code

```

#include <iostream>
#include <string>
#include <fstream>
using namespace std;
struct {
    string bookName[100];
    int leng;
}library;

struct {
    bool borrowed[10];
    string bookName[10];
}users;

```

```

class user {
public:
    user(void) {
        invenIni();
    }
    void login() { // login for users
        actUser = uCheck();
    }
    void invenIni() { // copies the library inventory from the inventory.txt file I wrote, can be appended up to 99
books
        string book;
        int i = 0;
        fstream invFile;
        invFile.open("inventory.txt", ios::in);
        if (invFile.is_open()) {
            while (getline(invFile, book)) {
                library.bookName[i] = book;
                i++;
            }
        }
        else {
            cout << "\nUNFORSEEN ERROR OPENING FILE CHECK IT'S PRESENT AND NOT
OPEN ELSEWHERE\n";
        }
        library.leng = i;
        invFile.close();
    }
    void inventory() { // prints the inventory
        for (int i = 0; i < library.leng; i++) {
            cout << i+1 << ". " << library.bookName[i] << "\n";
        }
        cout << endl;
    }
    void menu() { // menu screen for users
        int uIn;
        string uStr;
        do {
            cout << "0.main menu\n1.see inventory\n2.checkout book\n3.return book\n4.book
check(case sensitive)\n5.borrow status" << endl;
            cin >> uIn;
            while (cin.fail()) {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "invalid input try again" << endl;
                cin >> uIn;
            }
            switch (uIn) {
            case 1:
                inventory();
                break;
            case 2:
                checkout();
                break;
            case 3:
                retBook();
                break;
            case 4:
                cout << "book name: ";
                cin.ignore();
                getline(cin, uStr);
                inventory(uStr);

```

```

        break;
    case 5:
        borrowed();
        break;
    default:
        break;
    }

    } while (uIn != 0);
    cout << "save?(y/n) ";
    cin >> uStr;
    if (uStr[0] == 'y') {
        fileUpdate();
    }
}

void inventory(string bName) { // checks a books position in inventory and prints it also checks for a books
presence
    int i = 0;
    cout << library.leng << endl;
    while (i < library.leng && library.bookName[i] != bName) {
        i++;
    }
    if (i == library.leng) {
        cout << "not in inventory" << endl;
    }
    else {
        cout << "position " << i+1 << endl;
    }
}

bool uCheck() { // checks user number

    int i = 0;
    string line;
    fstream useFile;
    int userName;
    cout << "enter user number: ";
    cout << "\nfor testing purposes there are 10 user numbers going from 0-9\nbut in an actual
implementation there would be many user numbers ideally non sequential and not 1 digit long\n";
    cin >> userName;
    while (cin.fail()) {
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "invalid input try again" << endl;
        cin >> userName;
    }
    if (userName >= 0 && userName < 10) { // there would have to be a search of some sort here from
a list of usernames
        userNumber = userName;
        return 1;
    }
    else {
        cout << "User does not exist" << endl;
        return 0;
    }
}

void checkout() { // checks out a book

    int userNum;
    if (!users.borrowed[userNumber]) { // User has not got a book borrowed
        inventory();
    }
}

```

```

        cout << "Which book would you like to take out? ";
        cin >> userNum;
        while (cin.fail()) {
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

            cout << "invalid input try again" << endl;
            cin >> userNum;
        }
        if (userNum > 0 && userNum <= library.leng) {
            userNum--;
            users.bookName[userNum] = library.bookName[userNum];
            bookRemove(userNum);
            users.borrowed[userNum] = 1;
        }
        else {
            cout << "invalid entry" << endl;
        }
    }
    else {
        cout << "please return any books that may be out before borrowing another" << endl;
    }
}

void retBook() { // returns a book
    if (users.borrowed[userNum]) {
        library.bookName[library.leng] = users.bookName[userNum];
        users.bookName[userNum].clear();
        users.borrowed[userNum] = 0;
        library.leng++;
    }
    else {
        cout << "No book borrowed" << endl;
    }
}

void bookRemove(int i) { // removes a book from the library structure
    for (i; i < library.leng; i++) {
        if (!library.bookName[i + 1].empty()) { // if the string in the next element of the array is
not empty
            library.bookName[i] = library.bookName[i + 1];
        }
        else {
            library.bookName[i].clear();
        }
    }
    library.leng--;
}

void borrowed() { // prints the book currently borrowed if no book borrowed says that
    if (users.borrowed[userNum]) {
        cout << "borrowed: " << users.bookName[userNum] << endl;
    }
    else {
        cout << "no outstanding books" << endl;
    }
}

void fileUpdate() { // saves progress to file
    fstream invFile;
    invFile.open("inventory.txt", ios::out);
    if (invFile.is_open()) {
        for (int i = 0; i < library.leng; i++) {
            invFile << library.bookName[i] << "\n";
        }
    }
}

```

```

    }
    else {
        cout << "\nUNFORSEEN ERROR OPENING FILE CHECK IT'S PRESENT AND NOT
OPEN ELSEWHERE\n";
    }
    invFile.close();
}
bool actUser; // actual user
private:
    int userNumber; // current user
};

class staff:public user {
public:
    staff(void) {
        pass();
    }
    bool check() { // returns the private valid value
        return valid;
    }
    void menu() { // staff menu screen
        int uIn;
        string uStr;
        do {
            cout << "0.main menu\n1.see inventory\n2.sort inventory\n3.add book\n4.remove
book\n5.book check(case sensitive)\n6.save changes to inventory file" << endl;
            cin >> uIn;
            while (cin.fail()) {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "invalid input try again" << endl;
                cin >> uIn;
            }
            switch (uIn) {
            case 1:
                inventory();
                break;
            case 2:
                bblsort();
                break;
            case 3:
                cout << "book name: ";
                cin.ignore();
                getline(cin, uStr);
                addItem(uStr);
                break;
            case 4:
                inventory();
                cout << "remove(int): ";
                cin >> uIn;
                if (uIn <= 0 || uIn > library.leng) {
                    break;
                }
                while (cin.fail()) {
                    cin.clear();
                    cin.ignore(numeric_limits<streamsize>::max(), '\n');
                    cout << "invalid input try again" << endl;
                    cin >> uIn;
                }
                bookRemove(uIn - 1);
                uIn = 99;
                break;

```

```

        case 5:
            cout << "book name: ";
            cin >> uStr;
            inventory(uStr);
            break;

        case 6:
            fileUpdate();
            break;

        default:
            break;
    }

} while (uIn != 0);
cout << "save?(y/n) ";
cin >> uStr;
if (uStr[0] == 'y') {
    fileUpdate();
}

}

void bblsort() { // bubble sort

    string temp;
    for (int x = 0; x < library.leng; x++) {
        for (int i = 0; i < library.leng - (x + 1); i++) {
            if (library.bookName[i] == library.bookName[i + 1]) {
                bookRemove(i);
            }
            else
            if (nameCheck(library.bookName[i], library.bookName[i + 1])) {
                if (library.bookName[i].length() > library.bookName[i + 1].length()) {
                    temp = library.bookName[i];
                    library.bookName[i] = library.bookName[i + 1];
                    library.bookName[i + 1] = temp;
                }
            }
            else {
                int z = 0;
                while (library.bookName[i][z] == library.bookName[i + 1][z]) {
                    z++;
                }
                if (library.bookName[i][z] > library.bookName[i + 1][z]) {
                    temp = library.bookName[i];
                    library.bookName[i] = library.bookName[i + 1];
                    library.bookName[i + 1] = temp;
                }
            }
        }
    }
}

}

bool nameCheck(string s1, string s2) { // checks for books with the same names but one is longer like:
twilight and twilight: new moon
    string temp;
    int x;
    if (s1.length() == s2.length()) {
        return 0;
    }
    else if (s1.length() > s2.length()) {
        temp = s1;
        s1 = s2;
        s2 = temp;
    }
}

```

```

    }
    x = s1.length();
    for (int i = 0; i < s1.length() - 1 ; i++) {
        if (s1[i] == s2[i]) {
            x--;
        }
    }
    if (x == 0) {
        return 1;
    }
    else {
        return 0;
    }
}
void addItem(string s1) { // adds a book to the library
    library.bookName[library.leng] = s1;
    library.leng++;
}

private:
void pass(){ // password entry and checker
    string password;
    int n = 3;
    do {
        cout << "Enter password: ";
        cout << "\nFOR TEST PURPOSES THE PASSWORD IS \"abc123\" \n";
        cin >> password;
        if (password == word) {
            valid = 1;
        }
        else {
            valid = 0;
            cout << "INCORRECT PASSWORD " << --n << " MORE TRIES" << endl;
        }
    } while (password != word && n > 0);
}
string word = "abc123";
bool valid; // whether the user is locked out or not
};

class sysAdmin {
public:
    sysAdmin(void) {
        invenIni();
    }
    void menu() { // systems admin menu
        int uIn;
        do {
            cout << "0.main menu\n1.restore from backup\n2.update backup\n";
            cin >> uIn;
            while (cin.fail()) {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "invalid input try again" << endl;
                cin >> uIn;
            }
            switch (uIn) {
                case 1:
                    backUp();
                    break;
                case 2:
                    backupUpdate();

```

```

        break;
    case 3:
        invenIni(); // refreshes inventory
        users.bookName->clear();
        for (int i = 0; i < 10; i++) { // resets user borrows
            users.borrowed[i] = 0;
        }
        break;
    }
} while (uIn != 0);
}
bool login() { // systems admin login
    string password;
    cout << "SYSTEMS ADMIN\nenter password(123abc): ";
    cin >> password;
    if (password == "123abc") {
        cout << "correct" << endl;
        return 1;
    }
    else {
        cout << "bricked";
        return 0;
    }
}
void invenIni() { // copies the library inventory from the inventory.txt file I wrote, can be appended up to 99
books
    string book;
    int i = 0;
    fstream invFile;
    invFile.open("inventory.txt", ios::in);
    if (invFile.is_open()) {
        while (getline(invFile, book)) {
            library.bookName[i] = book;
            i++;
        }
    }
    else {
        cout << "\nUNFORSEEN ERROR OPENING FILE CHECK IT'S PRESENT AND NOT
OPEN ELSEWHERE\n";
    }
    library.leng = i;
    invFile.close();
}

private:
void backUp() { // restores the library file from the backup
    string book;
    int i = 0;
    fstream invFile;
    invFile.open("inventoryB.txt", ios::in);
    if (invFile.is_open()) {
        while (getline(invFile, book)) {
            library.bookName[i] = book;
            i++;
        }
    }
    else {
        cout << "\nUNFORSEEN ERROR OPENING FILE CHECK IT'S PRESENT AND NOT
OPEN ELSEWHERE\n";
    }
    library.leng = i;

```

```

        invFile.close();
        fileUpdate();
    }
    void fileUpdate() { // saves progress to file
        fstream invFile;
        invFile.open("inventory.txt", ios::out);
        if (invFile.is_open()) {
            for (int i = 0; i < library.leng; i++) {
                invFile << library.bookName[i] << "\n";
            }
        }
        else {
            cout << "\nUNFORSEEN ERROR OPENING FILE CHECK IT'S PRESENT AND NOT
OPEN ELSEWHERE\n";
        }
        invFile.close();
    }
    void backupUpdate() { // updates the backup with what's currently in the library
        fstream invFile;
        invFile.open("inventoryB.txt", ios::out);
        if (invFile.is_open()) {
            for (int i = 0; i < library.leng; i++) {
                invFile << library.bookName[i] << "\n";
            }
        }
        else {
            cout << "\nUNFORSEEN ERROR OPENING FILE CHECK IT'S PRESENT AND NOT
OPEN ELSEWHERE\n";
        }
        invFile.close();
    }
};

int main() {
    int uIn; // user input
    bool lockOut = 0;
    for (int i = 0; i < 10; i++) { // setting all users to unborrowed meaning they can borrow books and cant return
any
        users.borrowed[i] = 0;
    }
    do {
        if (lockOut) {
            uIn = 2;
        }
        else {
            cout << "0.public access\n1.staff access\n2.systems admin\n3.user guide\n4.quit" <<
endl;
            cin >> uIn;
            while (cin.fail()) {
                cin.clear();
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                cout << "invalid input try again" << endl;
                cin >> uIn;
            }
        }
        switch (uIn) {
            case 0:
                // had to enclose case 0 otherwise I get "initialization of 'person' is skipped by 'case' label"
error
    
```

```

        // do not get that error for the case 1 or 2 but I enclosed them anyway for sake of symetry
    {
        user person;
        person.login();
        if (!person.actUser) {
            break;
        }
        person.menu();
        break;
    }
    case 1:
    {
        staff personL;
        if (!personL.check()) {
            cout << "\nlocked out please contact system admin\n";
            lockOut = 1;
            break;
        }
        personL.menu();
        break;
    }
    case 2:
    {
        sysAdmin itGuy;
        if (lockOut) {
            if (itGuy.login()) {
                lockOut = 0;

            }
            else {
                uIn = 4;

            }

        }
        else {
            if (itGuy.login()) {
                itGuy.menu();
            }
            else {
                uIn = 4;

            }
        }
        break;
    }
    case 3:
        cout << "public access for returning/borrowing books\nstaff access for adding/removing
books\nadmin access for restoring from/saving to backup file" << endl;
    default:
        break;
    }
} while (uIn != 4);

return 0;
}

```

References

- [1] "Computer science," StudySmarter UK,
<https://www.studysmarter.co.uk/explanations/computer-science/computer-programming/inheritance-in-oops/> (accessed Jan. 10, 2024).

Nathan Willis Assignment

GRADEMARK REPORT

FINAL GRADE

44/100

GENERAL COMMENTS

Hi Nathan,

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:

video demonstration provides a visual walkthrough, making it easier to follow the steps.

Effective use of test cases to identify and resolve issues encountered during testing.

areas of improvement:

Ensure consistent and descriptive naming conventions for variables, functions, and classes throughout the codebase.

Review the code for potential memory management issues, especially in functions that manipulate arrays or dynamic memory.

Completion of the course from CISCO to a possible extent could have added credits

PAGE 1

PAGE 2

PAGE 3



Comment 1
no comment



Comment 2

what is inheritance?



Comment 3

elaboration on specific aspects is anticipated.



Comment 4

all validation on main menu are explored. good job

PAGE 4



Comment 5

textual description about each option could have been specified

PAGE 5



Comment 6

textual displays in output could provide more clarity to the user

PAGE 6

PAGE 7



Comment 7

not clear. could have provided more specifics

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13



Comment 8

error handling has been done well, however success attempt should also been recorded

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21

PAGE 22

PAGE 23

PAGE 24

PAGE 25

GRADING FORM: DC-ASD

NATHAN WILLIS

44

LITERATURE REVIEW

with IEEE referencing



2

KNOWLEDGE&UNDERSTANDING SKILLS

use of OOPs features



8

COGNITIVE&INTELLUCTUAL SKILLS

Testing &Error handling



12

PRACTICAL SKILLS

Working program Demo video



20

TRANSFERRABLE SKILLS

User Manual, document style



2

CISCO CERTIFICATE



0