Project Specification - Major Practical

Introduction

Our project is about library systems that we use in real life, such as logging system, borrow-return material (book, e-book and DVD) system and managing library material system for staff. The library system allows the current users (public, staff) log in, log out, borrow materials, return materials and change password. It also allows a new user to make a new account. There are more specific behaviours that the staff is able to access, such as check history and add materials.

Design Description

Assessment Concepts

Memory allocation from stack and the heap

- **Arrays :** we are using a dynamic array for storing the books(DVDs) that the user have borrowed.
- **Vectors :** we are using a vector for storing all the materials(ebooks, books, dvds) and user(public, staff) lists
- **Strings :** material names, author names and user names
- **Objects :** Ebook, book, DVD, public, staff

User Input and Output

- **I/O of different data types :** Command-line.

    - A person enters a number to choose log in types(user, staff or new user) - console displays the possible options that user or staff can select.
    - The user is asked to enter a behaviour number to borrow materials(ebook, book or dvd), return materials or change password - the program displays corresponding outputs.
    - The staff is asked to input a number to choose the opitions of borrowing materials, returning materials, changing password, checking history or adding material – the program prints out the corresponding outputs.

Object-oriented programming and design

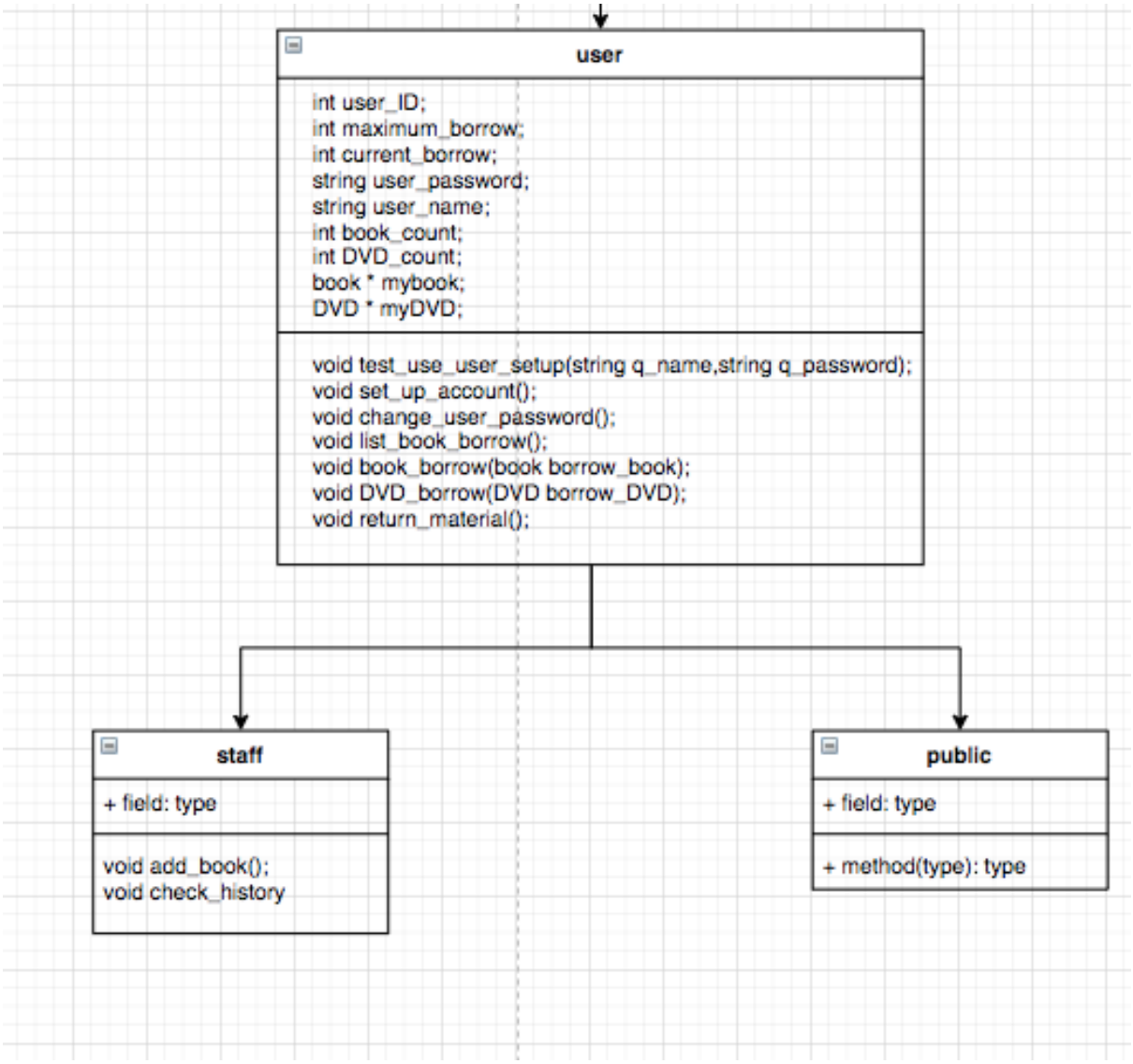
- **Inheritance:**
    - material -parent class / Ebook, book and DVD – child classes
        - There are different types of materials such as Ebook, book and DVD.
        - Each child class (Ebook, book, DVD) inherits from the parent class(material)

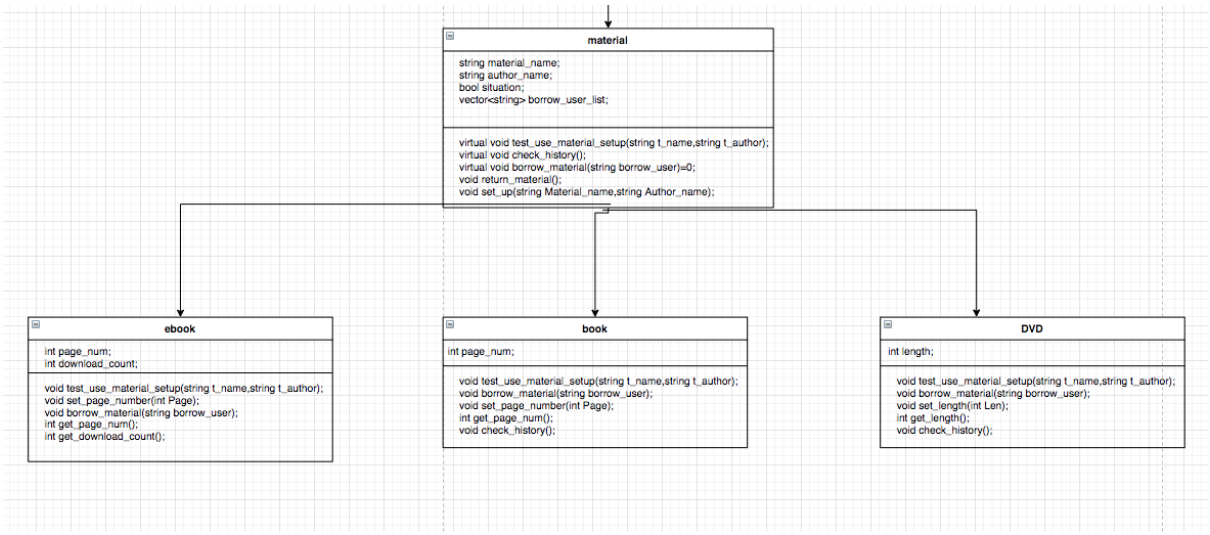
    - Each child class has few different behaviours to other child class
- user(staff and public)
    - There are two types of users such as staff and public
    - Both child classes inherit from parent class(user)
    - Each class has few different behaviours (the staff user can access more library system)

- **Polymorphism:**
    - test_use_material_setup(string t_name, string t_author) function
        - Child classes(Ebook, book and DVD) have the same function for quick setting up the objects but each child class has different implementations
    - check_history() function
        - Child classes(Ebook, book and DVD) have the same function check_history() but the outcomes are different. The check history for book(DVD) prints out the list of people who borrowed the book(DVD) and the check history for ebook prints out the number of ebooks that people have downloaded

- **Abstract classes:**
    - borrow_material(string borrow_user) = 0;
        - Every child class has the function called borrow_material to borrow the material, which has different outcomes.
        - According to ebook function(borrow_material), we are going to count out how many ebooks the user has downloaded. However, the borrow_material function for book and DVD shows the situation of availability of borrowing the book and DVD.
    - add_new_material() = 0;
        - Parent class does not declare the add_new_material function details but every child class (ebook, book, dvd) defines different function details.
        - The add_new_material function for Ebook and book asks user for entering material's name, author's name and page numbers but the add_new_material function for DVD save the information about material's name, author's name and length.

Class Diagram





Class Descriptions

material:

the superclass of all material in the library.(include book, ebook and DVD), contain attribute :

material name
author_name
situation (of available to borrow or not)
vector<string> borrow_user_list (record the name of user who borrow this material)

ebook:

child class of material, one of the material type. contain extra attribute:

download_count (count the time user download this book instead of borrow)
page_num (pages record)

book:

child class of material, one of the material type. contain extra attribute:

page_num

DVD:

child class of material, one of the material type. contain extra attribute:

length (time length of CD);

user:

the superclass of staff. contain attribute

user_ID;
maximum borrow (limit number of book can borrow)
current borrow (count of recent book borrow)
user_password
user_name
book_count & DVD count (use to calculate the user borrow and check it hit the limit.)
mybook, myDVD (array for storing material that user have borrowed)

User Interface

The users (public, staff) is going to input the numbers to choose one of the options using the command-line.

The options that the users can select is like:

What would you like to do today?
1: Borrow material
2: Return material
3. Change current password
4. Check history (only for staff)
5. Add material (only for staff)
6. Delete material (only for staff)

Testing Plan

We are going to compile all the codes through our makefile.

Unit Testing

We have separate class function files and corresponding test files to check if the individual files can be implemented. When each individual class is tested successfully, we test the main file to see if all files are well compiled and run completely.

- *test_add_material:*
  g++ add_material_test.cpp material.cpp add_material_function_list.cpp -o test
- *test_user_setup :*
  g++ material.cpp user.cpp user_setup_test.cpp choose_material.cpp -o test_user_setup
- *test_user_borrow_return_test :*
  g++ user.cpp material.cpp user_borrow_return_test.cpp choose_material.cpp -o test_user_borrow_return_test
- *test_material :*
  g++ material.cpp material_test.cpp -o test_material
- *test_material_borrow_return_checkhistory :*
  g++ material.cpp material_borrow_return_checkhistory_test.cpp -o test_material_borrow_return_checkhistory
- *test_user_staff :*
  g++ material.cpp user.cpp choose_material.cpp user_staff_test.cpp -o test_user_staff
- *test_choose_material :*
  g++ material.cpp choose_material.cpp choose_material_test.cpp -o test_choose_material

Input/output testing

Through the input/output testing, we check if the output is equal to the expected output.

Integration testing

We have a main.cpp file and integrate all functions together including the functions in our classes that we have tested in separate test files so far. Through the main file, we do integration testing to see if all files are well compiled and run completely.

- *All:*

  g++ main.cpp material.cpp user.cpp log_in.cpp choose_material.cpp list_of_materials.cpp user_behavior.cpp -o test

Schedule Plan

Stretch Goals

Our goal is to make a diagram to check if we satisfy all the rubrics and write down the pseudo codes for every class and functions. When we satisfy the requirements, we are going to expand our idea.

week 8	- Write down the diagram - *Waitong* - Make a group and create SVN – *Hyun Ji* - make our plan – Waitong, Hyun Ji
break	- organise the group works – *Waitong, Hyun Ji* - separate the classes and write some codes - *Waitong* - write down separate codes(class, function) - *Waitong* ✓ material.h / material.cpp ✓ user.h / user.cpp ✓ add_material_function_list.cpp - write down separate corresponding test files– *Waitong*, *Hyun Ji* ✓ add_material_test.cpp ✓ borrow_return_DVD_test.cpp ✓ borrow_return_book_test.cpp
week 9	- write down project plan – *Waitong, Hyun Ji* - write down separate codes(class, function) - *Waitong* ✓ log-in.cpp ✓ add_material_function_list.cpp ✓ user_behaviour.cpp - write down separate corresponding test files and check if the individual files are run properly – *Waitong*, *Hyun Ji* ✓ material_test.cpp

	✓ user_borrow_return_test.cpp ✓ user_setup_test.cpp ✓ log_in_test.cpp - write makefile - Waitong - testing
week 10	- edit project plan – *Hyun Ji* - write down separate codes(class, function) - *Waitong* ✓ choose_material.cpp - write down separate corresponding test files– *Waitong*, *Hyun Ji* ✓ material_borrow_return_checkhistory_test.cpp ✓ user_staff_test.cpp ✓ choose_material_test.cpp - write down the main file combining all the codes – *Waitong* ✓ main.cpp - correcting the codes and debug – *Waitong, Hyun Ji* - write makefile – *Waitong, Hyun Ji*
week 11	- correcting the codes and debug – *Waitong, Hyun Ji* - fixing some problems that we made in week 10 – *Waitong, Hyun Ji* - testing final main file – *Waitong, Hyun Ji* - write makefile – *Waitong, Hyun Ji* - write down specific testing files - *Waitong, Hyun Ji*