CP2406 Coding assignment

# Submission details:

Your name: Kantapong Wongsanguan

Your student id: 14405427

My Private GitHub Classroom Link: <https://github.com/CP2406/cp2406-2023-tr3-assignment-KantapongWongJC14405427>

Tasks with marks

The started code is the employee database example from chapter 1.

[5 marks] Compile and run UserInterface.cpp.

Insert screenshots here:

* At first running UserInterface.cpp got terminated.
* To fix this exit error for this to run, I attempted to debug it with the choice,” Debug Anyway”, selected which showed launch.json and tasks.json.

A computer screen with text and images

Description automatically generated

* In tasks.json I adjusted additional arguments to pass to the complier or compilation script from “${file}" to ${fileDirname}/\*.cpp"

A computer screen shot of a program

Description automatically generated

* Now UserInterface.cpp can be compile and run as the result shows below.

A screenshot of a computer screen

Description automatically generated

[10 marks] Add a debug option, see ch-31-debugging. When the debug option is enabled, print tracing to the screen and to a log file.

Insert screenshots of relevant code and output results here:

* Created Debugger.h for debug option. With optional on line 8 for disabling when the debugging is not needed.

A computer screen shot of a program

Description automatically generated

* Line 84: log(“entered”) & line 94: log(“exited”) showing when debugging is called upon.

A computer screen with white text

Description automatically generated

* Evidence for debugging can be found on terminal screen, when option 1) Hire a new employee is entered and when Last name? is entered.

A screenshot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

* Debugging evidence are recorded in debugLogfile.txt when the function is called.

[10 marks] Add option “7) Generate new database”. This option must create 8,000 employees in memory by selecting every combination of 20 first names, and 20 middle names, 20 last names.

Invent an algorithm to generate some fake addresses. Each address must be unique foreach employee. You will need to add new fields “address” and “middle name”.

Then use the interface option (list all) to view your result.

Insert screenshots of relevant code and output results here:

A computer screen shot of a program

Description automatically generated

A computer screen shot of a program

Description automatically generated

* For this task, new fields “address” and “middle name” are added to Employee.h and Employee.cpp as getters and setters.

A computer code on a black background

Description automatically generated

* Hence, the display needed to be edited since middle name and address are added. To test this, necessary codes are edited into EmployeeTest.cpp .

A screen shot of a computer code

Description automatically generated

A screen shot of a computer code

Description automatically generated

* In UserInterface.cpp, new option (7) is added to support the task.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer code

Description automatically generated

* As for how the database is generated, the code in screenshots above shows the application method for this task. This was coded into UserInterface.cpp as referencing to the Database as a parameter avoiding copying. The random database is generated from names and addresses in the file called TwentyNames.txt

A black rectangular object with white text

Description automatically generated

* The above figure shows the working evidence of the generated database, the employee number shows that 8000 new random employees were created since the last employee has the number of 1000.

[10 marks] Add option “8) Save database to file”. This will save your database to a text file.

Ask user for the file name. Ask user to confirm if overwriting an existing file.

Insert screenshots of relevant code and output results here:

A computer screen shot of a code

Description automatically generated

A screen shot of a computer

Description automatically generated

* Option 8) Save database is edited in UserInterface.cpp to support the task.

A computer screen shot of text

Description automatically generated

* The method to save database is added into Database.cpp.

A computer screen shot of text

Description automatically generated

A computer screen shot of text

Description automatically generated

* The screenshots above are the code to support this task function, checkFileName(), to Check and return filename to be used. This is called before the saving process since in case of overwriting permission, case 1: & case 2:, shows in the screenshots.

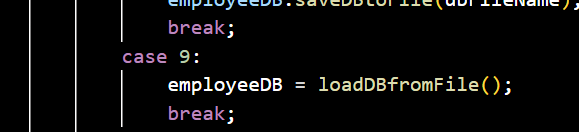
A screenshot of a computer

Description automatically generated

* This is a text file called “9” where the data generated is saved in a quoted format with one line per employee data.

[10 marks] Add option “9) Load database from file”. This will load your database from a text file. Ask user for the file name.

Insert screenshots of relevant code and output results here:



A black screen with white text

Description automatically generated

* The Interface output is edited for this task option, 9).

A screen shot of a computer

Description automatically generated

A black screen with white text

Description automatically generated

* The code for loading function is coded into the UserInterface.cpp file. The method works by reading the text file line by line, feeding the reading into stream of strings which is the loaded data which is passed onto the method called loadEmployee() in Database.h & Database.cpp.

A screen shot of a computer

Description automatically generated

A computer screen shot of code

Description automatically generated

* loadEmployee() method in Database.h and Database.cpp accordingly.

A screenshot of a computer

Description automatically generated

* This output shows that the file data is loaded correctly when the correct txtfile is called upon entering the 9th option. In this case, the file was called “9” as mentioned in the previous task where it contains the 8999 employees generated.

[10 marks] Add option “10) Edit employee”. You should be able to edit address, salary, hired.

Insert screenshots of relevant code and output results here:

A black screen with white text

Description automatically generated

A black background with white text

Description automatically generated

* The new option is added into UserInterface.cpp to support this task.

A computer screen shot of a program code

Description automatically generated A screen shot of a computer program

Description automatically generated

* Here is the code for this task. The function editEmployee() tasks the user must to edit

employee with their number, providing the output with a menu and uses a switch to

perform the edits. The result then displays before and after editing.

A screenshot of a computer screen

Description automatically generated

* This is the evidence to shows the working function where the address of the employee 8999 is edited.

[10 marks] Add option “11) search employee”.

Ask user: how to search. By first, middle, last name or address. You should be able to search by a fraction of text in all cases. For example, “Sm” in last name should return all “Smith” and “Smarth”.

Insert screenshots of relevant code and output results here:

A computer screen with text on it

Description automatically generated A screenshot of a computer code

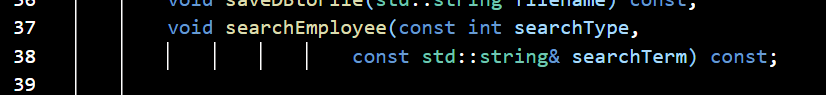
Description automatically generated

* Option 11 is added into UserInterface.cpp to support this task.

A screenshot of a computer program

Description automatically generated

* The method function, searchEmployee(db) is coded into UserInterfaec.cpp.



* Where the function is called into the method in Database.h

A computer code on a black background

Description automatically generated

A computer screen with text on it

Description automatically generated

* And the method in Database.cpp in the following screenshots showing the function to searchEmployee.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* These screenshots above shows the working function and methods necessary for this task to search employee and what output the option displays.

[10 marks] Your code has good style (see chapter-3-style): variable and method names, comments.

* The following screenshots of my code are to show the style I coded along with various variable, spacing and method names, concise comments where appropriate.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

HD-level task [15 marks] Implement two levels of login: Manager and Employee.

Manager: all user-interface options are available. Plus: new options to Create/Edit/Delete/Save-to-file/Load-from-file users’ login ids and passwords.

Employee: Can only view (not edit) own record.

Insert screenshots of relevant code and output results here:

To tackle this task, I created these files.

* LogIn Class Creation: Created a LogIn class with functionalities to manage user login credentials. The class included methods to validate user credentials (validate()), check user statuses, and display login information.
* LogInDataBase Class Implementation: Developed a LogInDataBase class responsible for storing instances of the LogIn class, essentially simulating a database of user login information. This class managed operations such as adding new users, displaying user information, and retrieving user details.
* TestLogIn.cpp for Functionality Verification: Implemented a test file to verify and test the functionalities of the LogIn and LogInDataBase classes. The test file included scenarios for signing up new users, displaying login details, and attempting user authentication.
* Integration with UserInterface.cpp: Integrated the login functionalities into the UserInterface.cpp. This involved creating instances of LogInDataBase, adding users to simulate a basic login database, and implementing authentication prompts before allowing access to the employee database functionalities.

However, nothing changes to the display menu when I run debug on testLogIn.cpp, hence, I tried editing UserInterface.cpp  
  
In summary,

* Integration of Login System: Added functionalities to handle user authentication and authorization using a login system.
* Login Classes Integration: Included the necessary headers (LogIn.h and LogInDataBase.h) and classes (LogIn and LogInDataBase) to support user authentication.
* Login Database Initialization: Created an instance of LogInDataBase and added users (with usernames, passwords, and roles) to simulate a basic login database.
* Login Prompt and Authentication: Implemented functions (authenticate() and promptLogin()) to prompt users for their credentials, validate them against the login database, and grant access based on successful authentication.
* Conditional Database Access: Modified the main function to conditionally allow access to the employee database functionalities (doHire(), doFire(), etc.) based on successful login/authentication.
* Loop for Menu Operations: Created a loop to continuously display the menu and execute functionalities related to the employee database after successful login.

A screenshot of a computer

Description automatically generated

However, despite all these attempts and changes, the output still isn’t what we want to solve this task, and this is as far as I can tackle this task.  
The output I provided seems to match the initial menu interface which doesn’t reflect the added functionalities of Manager and Employee login levels.

To implement the two levels of login and the associated functionalities, further alterations need to be made to the code.

* UserInterface.cpp addition code

A screen shot of a computer code

Description automatically generated

A computer screen shot of text

Description automatically generated

* LogIn.cpp

A screenshot of a computer program

Description automatically generatedA black background with many small colored lights

Description automatically generated with medium confidence

* LogIn.h

A computer screen shot of a program code

Description automatically generated

* LogInDataBase.cpp

A screenshot of a computer program

Description automatically generated

* LogInDataBase.h

A computer screen shot of a program

Description automatically generated

* testLogIn.cpp

A screen shot of a computer program

Description automatically generated