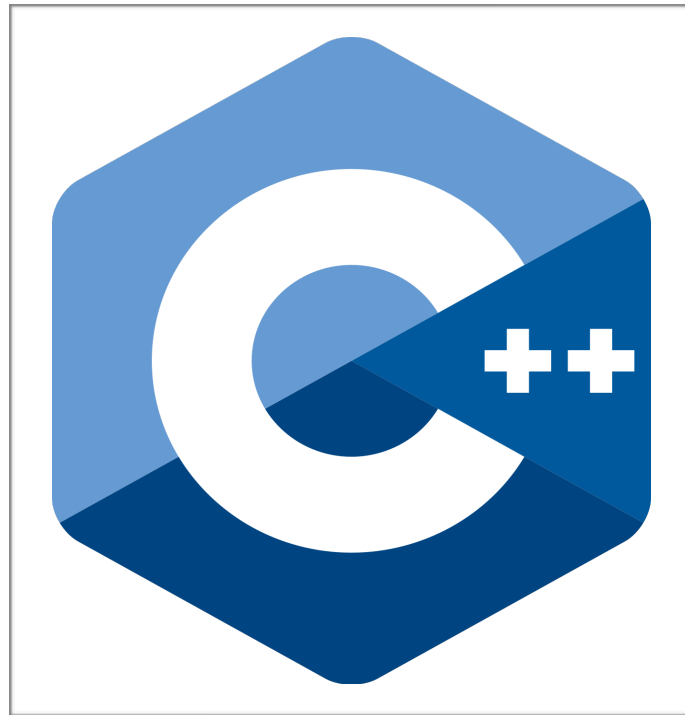


Student Information Management system

Project#1 Problem2



Team 4

20185659 김혜성 (Team Leader)

20186912 김지수 (Presenter)

20172774 송정우

20183032 임창성

20182592 장원범

20182966 조준오 (Presenter)

목차

Table of contents

1. Summary
 - 1.1. Project Title
 - 1.2. List of Team Members
 - 1.3. Presenter Name
 - 1.4. Project Description
2. Compilation and Execution
 - 2.1. How to Compile and Execute
 - 2.2. System Requirement for Compilation and Execution
3. Functionality
 - 3.1. Functionality that was Implemented
4. Implementation
 - 4.1. Important Implementation Issues
5. Result of SW system design
 - 5.1. UML Diagrams
6. Execution results
 - 6.1. Real Examples of Program Execution
 - 6.2. Function Correctness
7. Evaluation
 - 7.1. Object-Oriented Concepts
 - 7.2. Impressions
8. Conclusion

1. Summary

1.1. Project Title

Student Information Management System

1.2. List of Team Members

20185659 김혜성 (Team Leader)

20186912 김지수

20172774 송정우

20183032 임창성

20182592 장원범

20182966 조준오

1.3. Presenter Name

20186912 김지수

20182966 조준오

1.4. Project Description

Our project, “Student Information Management System”, is development of the management system for student information. **This program supports inserting, searching, modifying, displaying, deleting student information and Thanos’ finger snap function.** After terminating the program, the inserted, searched, modified, and deleted information will be stored in a file.

2. Compilation and Execution

2.1. How to Compile and Execute

There are multiple ways of running the program.

a. Use CLion (tested on CLion 2019.2)

Click 'Checkout from Git Repository' - 'Git'. Click the run arrow on the top right.

b. Use CMake and Makefile (tested on macOS 10.15)

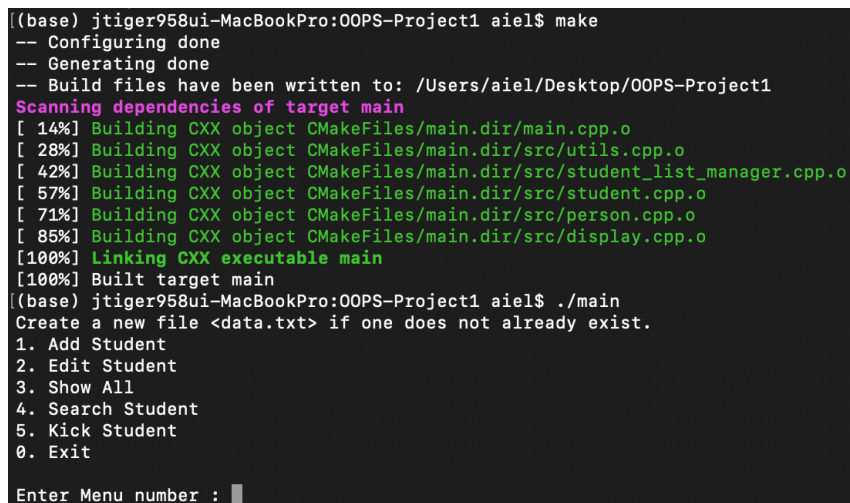
In bash terminal, enter the following commands consecutively to compile and execute the program. The user must install a compiler compatible with CMake(in this case, XCode), and CMake must also be installed beforehand.

```
cmake CMakeFiles.txt
```

```
make
```

```
./main
```

The result should look like **[picture 1]** in bash terminal.



```
(base) jtiger958ui-MacBookPro:OOPS-Project1 aiel$ make
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/aiel/Desktop/OOPS-Project1
Scanning dependencies of target main
[ 14%] Building CXX object CMakeFiles/main.dir/main.cpp.o
[ 28%] Building CXX object CMakeFiles/main.dir/src/utls.cpp.o
[ 42%] Building CXX object CMakeFiles/main.dir/src/student_list_manager.cpp.o
[ 57%] Building CXX object CMakeFiles/main.dir/src/student.cpp.o
[ 71%] Building CXX object CMakeFiles/main.dir/src/person.cpp.o
[ 85%] Building CXX object CMakeFiles/main.dir/src/display.cpp.o
[100%] Linking CXX executable main
[100%] Built target main
(base) jtiger958ui-MacBookPro:OOPS-Project1 aiel$ ./main
Create a new file <data.txt> if one does not already exist.
1. Add Student
2. Edit Student
3. Show All
4. Search Student
5. Kick Student
0. Exit

Enter Menu number : █
```

▲ [picture 1] bash terminal

c. Use g++

In bash terminal, enter the following command.

```
g++ -std=c++11 main.cpp src/display.cpp src/person.cpp src/student.cpp src/student_list_manager.cpp src/utls.cpp -o main
```

2.2. System Requirement for Compilation and Execution

- Operating system : Linux, Unix (such as macOS)
- Compiler : g++
- Version of the language : C++11

3. Functionality

3.1. Functionality that was Implemented

Insertion, Search, Modification, Display, Deletion, and Thanos' Finger Snap function is implemented in our SW system.

In the Insertion function, users can insert student information into the system. If the user inserts the information, the system checks the format of the information and then writes it in a file. The information format is like **[picture 2]**.

Name	Age	Id	Department	Phone number
------	-----	----	------------	--------------

▲ **[picture 2] Information format of student informations**

In the Search function, users can search stored student informations. There are 4 options for searching informations. Searching by name, searching by age, searching by Id, and searching by department.

In the Modification function, users can modify student informations. If the user tries to change the student information, the system searches the student information by student Id and modifies the corresponding student's information. Then the modified information is written into the file.

In the Display function, users are presented with a clean and succinct interface. If the user wants to inspect the information, the system displays five entries per page. It plays a crucial role in enhancing the user experience when interacting with studentListManager.

In the Deletion function, users can remove stored student information. If the user chooses to delete a certain entry, the system deletes the student information distinguished by the student Id. Also, the deleted entry is removed from the file.

Thanos' Finger Snap function was inspired from a brainwave of the team. Getting the inspiration from the movie, "Avengers" we tried to implement a method that randomly removes half of the student information.

Functions in our system are for managing student information. With basic functions, Insertion, Search, Modification, Display, and Deletion, we add Thanos' Finger Snap function for fun. Users would be impressed by the ascii painting of Thanos' Finger.

4. Implementation

4.1. Important Implementation Issues

There are several important implementation issues. The following is the issues on a basis of priority.

First of all, for efficiency, our system sorts the information by using lower bound in adding student level. The lower_bound function, defined in the algorithm header, indicates where objects can be inserted without breaking the alignment. We can assume that the empty data file is aligned by name. So, by using lower_bound algorithm we can insert the student object in order. We suppose that there are n information to insert. The average time complexity quick sort algorithm is $O(n * \log(n))$. But the time complexity of inserting information using lower bound is at most $O(n * \log(n))$.

Secondly, our system searches information by linear search. There are many search categories. If it were binary search, it should be aligned for each categories. Then, the sorting task would have too much cost. Using linear search, we get another advantage. Since our student information is initially aligned by name, search results are also aligned by name.

Third, we use lambda function in copy if function of that is responsible for searching. It is for readability and simplicity of the source code. On that latter point, defining additional function would makes the code messy since each comparing function is only used one time in each search menu. Because lambda function is recently defined grammar, we use c++11 which is reformed one.

Fourth, we overload operation in Student, Person class. Our system often sort information by name. By using comparison operation overloading, our system automatically sorts the information when executing algorithms, lower bound.

Fifth, our system has high security of information. For search, the function return copied student vector instead of address of it. This reduces the likelihood of arbitrary information changes.

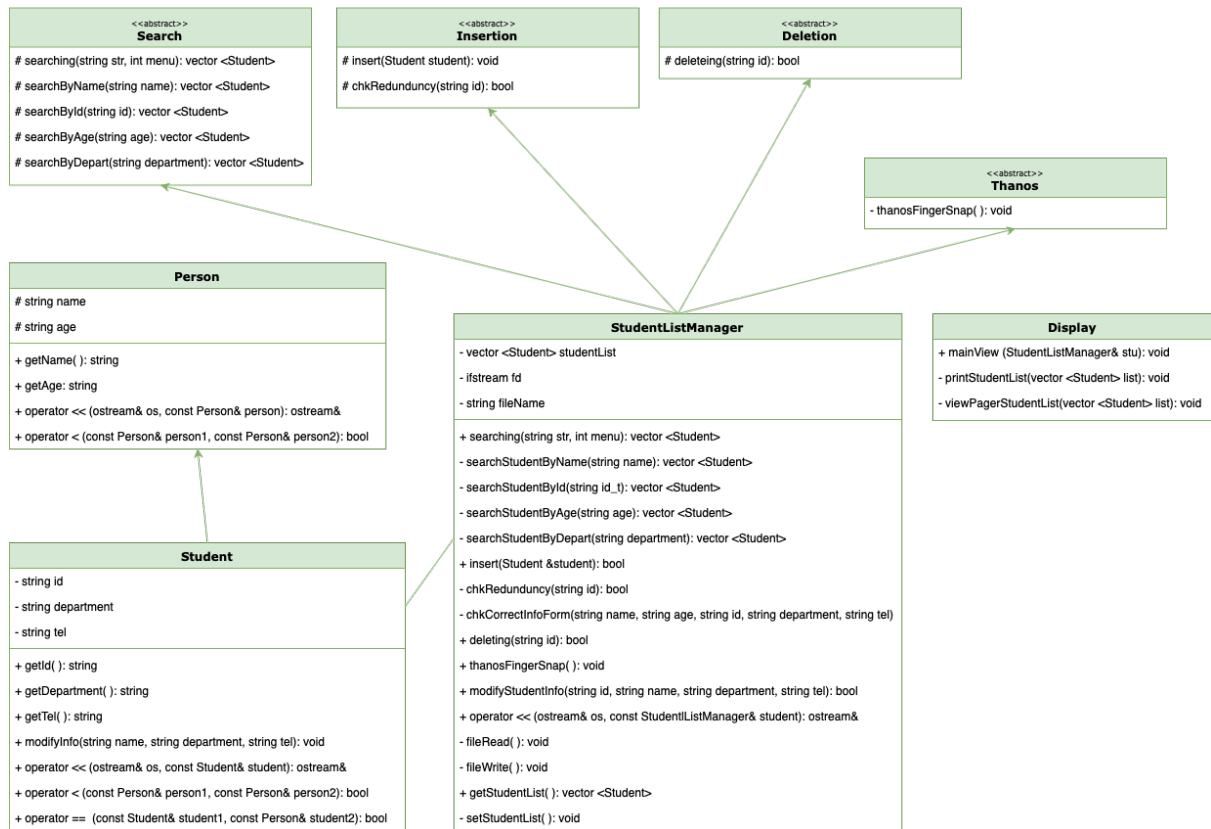
Sixth, our system has high usability considering user's interface. We grouped all search functions into single search function and use enum for consisting searchMenu. Our system has a very intuitive interface, even being a console application.

Seventh, our system implement viewPagerStudentList. The viewPagerStudentList is responsible of displaying information. It shows five students per page when displaying a list.

Lastly, our system uses regular expressions for input formats.

5. Result of SW system design

5.1. UML Diagrams



▲ [picture 3] UML diagram

6. Execution results

6.1. Real Examples of Program Execution

a. Add Student

Users add student by entering the number 1(Add Student) of Menu from [\[picture 1\]](#) in bash terminal. If the number 1 is entered, the form of the student information appears to users and users can store the information, just like [\[picture 4\]](#).

```
1. Add Student
2. Edit Student
3. Show All
4. Search Student
5. Kick Student
0. Exit

Enter Menu number : 1
Enter Name : Jang Wonbeom
Enter Age : 21
Enter ID : 2018259200
Enter Department : Computer Science
Enter Telephone : 01012341234

Error : Already inserted
1. Add Student
2. Edit Student
3. Show All
4. Search Student
5. Kick Student
0. Exit

Enter Menu number : 1
Enter Name : Jang Wonbeom
Enter Age : 21
Enter ID : 2018259200
Enter Department : Computer Science
Enter Telephone : 01012341234
```

▲ [\[picture 4\]](#) Real Example of Program Execution for add student

b. Edit Student And Search Student

Users can modify student by entering the number 2(Edit Student) of Menu from [\[picture 1\]](#) in bash terminal. If the number 2 is entered, the stored form of the student information appears to users and users could edit the information, just like [\[picture 4\]](#).

Users can search student by entering the number 4(Search Student) of Menu from [\[picture 1\]](#) in bash terminal. If the number 4 is entered, the 4 forms appears to users for which form to search, just like [\[picture 5\]](#).

```
Enter Menu number : 2
Enter ID : 20182592

1. Add Student
2. Edit Student
3. Show All
4. Search Student
5. Kick Student
0. Exit

Enter Menu number : 2
Enter ID : 2018259200
Enter new Name : Jang Wonbeom -> Jang Beomwon
Enter new Department : Computer Science -> Physics
Enter new Telephone : 01012341234 -> 01012341234

1. Add Student
2. Edit Student
3. Show All
4. Search Student
5. Kick Student
0. Exit

Enter Menu number : 4
Search for...?
1. ID
2. Name
3. Age
4. Department
5. Quit
Enter Mode : 1
Enter ID : 2018259200

LIST
ID | NAME | AGE | DEPARTMENT | TEL |
2018259200 | Jang Beomwon | 21 | Physics | 01012341234 |
```

▲ [\[picture 5\]](#) Real Example of Program Execution for edit and search student

c. Kick Student

Users can delete student by entering the number 5(Kick Student) of Menu from **[picture 1]** in bash terminal. If the number 5 is entered, input line of the student Id to delete appears to users. Then the student information of the Id will be deleted, just like **[picture 6]**.

```
1. Add Student
2. Edit Student
3. Show All
4. Search Student
5. Kick Student
0. Exit

Enter Menu number : 5
Enter Id : 2018239200
Wrong input. Try again.

1. Add Student
2. Edit Student
3. Show All
4. Search Student
5. Kick Student
0. Exit

Enter Menu number : 5
Enter Id : 2018259200
Goodbye, student.
```

▲ [picture 6] Real Example of Program Execution for kick student

d. Thanos' Finger Snap

If users entered the word, ‘thanos’, on the input line of the Menu from [\[picture 1\]](#) in bash terminal, Thanos’ Finger Snap function is executed along with the beautiful ascii art, just like [\[picture 7\]](#). Then, half of the entries in the file, selected randomly, are deleted.

[illegible]

▲ [picture 7] Real Example of Program Execution for Thanos' Finger Snap

6.2. Function Correctness

a. Data file

```
Albert Einstein|76|1879123400|Play|01043214321
Barack Obama|58|1980123400|Law|01088888888
Bill Gates|63|1975123400|Business|01099999999
Changsung Lim|23|2018303200|Play|01044444444
Chayeong|23|2011123400|Dance|01077778888
Cho Junoh|22|2018296600|Computer Science|01066666666
Dwayne Jhonson|47|1992123400|Chemistry|01077887788
Jang Wonbeom|21|2018259200|Economy|01055555555
Kim Hyeseong|25|2018565900|Mathematics|01011111111
Kim Jisu|20|2018691200|English|01022222222
Linus Torvalds|49|2000123400|Design|01000000000
Lionel Messi|32|2010123400|Play|01012341234
Michael Jackson|51|1978123400|Play|01098769876
Michael Jordan|56|1999123400|Physical Education|01077777777
Robert DowneyJr|54|1985123400|Acting|01011113333
Song Jungwoo|19|2017277400|Physics|01033333333
~
~
~
~
~
```

▲ [picture 8] Data file

7. Evaluation

7.1. Object-Oriented Concepts

In our project, we mainly considered the object-oriented concept, “SOLID(SRP, OCP, LSP, ISP, DIP)”. Especially, SRP, OCP, ISP are the important considerations.

For SRP, Single Responsibility Principle, each class in our system has only one responsibility each. In other words, the factor affecting classes to change must be only one thing. In view of SRP, each class works as what the class's name expresses. To design right names is the best way to show the responsibility.

For OCP, Open Close Principle, our system is open to the expansion of software components(component, class, module, function) and close to the modification. That is, the cost for the modification should be minimized, and the cost for the expansion should be maximized. Therefore, existing components must not be modified, though any modifications or additional needs arise. Reusability is the key point of this concept.

For ISP, Interface Segregation Principle, functions that use pointers or references to base classes are able to use objects of derived classes without knowing it. In other words, subtype should always be compatible with base type.

In terms of design, there could be some questions below.

First of all, ‘why some classes are designed in abstract(in other words, interface)?’ By the abstract functions, adding services, polymorphic transition exists through overriding. Also, in terms of ISP(Interface Segregation Principle) concept of SOLID, we divided a manager interface into 4 functions. Insertion, Search, Modification, and Deletion. Besides, if the interfaces were concrete class, studentList should be in class or passed by parameter call. In this case, all of the informations were not be encapsulated. Thus, we designed studentList class only to be managed by studentManager class.

Secondly, ‘why all the member variables in the student class are string type?’ For input validation, we set all member variables to string type in order to prevent the entire design from being changed by input processing, even if other items are added. Also, for flexible design, we executed the function of each function as the member variables in same type, and a polymorphic structure. Therefore, it was easy to make functions of each function in abstract. Through this, we could build a polymorphic structure, enabling flexible implementation.

7.2. Impressions

We were impressed on what professor, Son, emphasized. We, normally, ignore the importance of Object-Oriented Programming(OOP) concepts while we were developing little projects, especially, assignment. However, this time, we mostly considered OOP concepts. **In studying OOP concepts, for the assignment, we learned about SOLID concept,**

an acronym for 5 important design principles when doing OOP. The intention of these principles make software designs more understandable, easier to maintain and extend.

8. Conclusion

Our project, “Student Information Management System”, is development of the management system for student informations, supporting Insertion, Search, Modify, Display, Delete, and Thanos’ finger snap function. We mainly considered Object-Oriented concepts. Especially SOLID, an acronym for 5 important design principles when doing OOP. **By using this, our system has gained extension. Besides, as we mentioned before in the implement issues, our system has gained efficiency, in terms of system, and conciseness, in terms of users.**