

CSED232 Object Oriented Programming (Spring 2019)

Term Project : POSTECHIAN 관리 시스템

조 이름 : P-Manager (10조)

1. 강민호, 20180918, 무은재학부
2. 송수민, 20180085, 무은재학부
3. 조준형, 20180324, 창의it융합공학과

학번: 20180324

학과: 창의it융합공학과

이름: 조준형

POVIS ID: junhyeong99

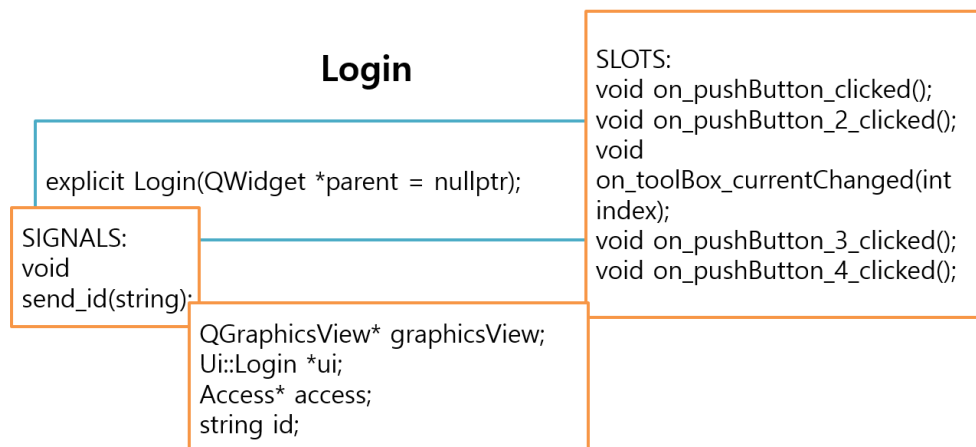
프로그램 기능에 대한 개요

이 프로그램은 POSTECH 학생이 학교를 다니면서 했던 활동들을 총체적으로 관리해주는 시스템이다. 포스테키안은 4년 동안 많은 활동들을 수행하고 졸업하게 되는데, 자신이 한 활동들을 체계적으로 관리할 수 있게 만들면 좋겠다고 생각하여 이러한 프로그램을 구상하였다.

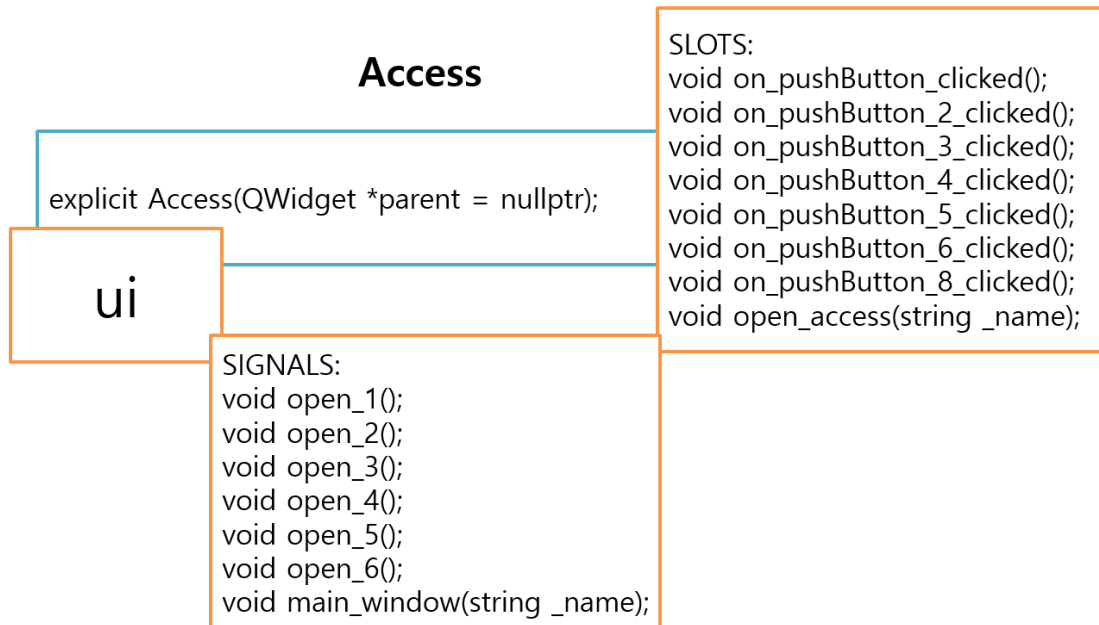
사용자는 회원가입을 진행하여 "ID / PASSWORD / 학과"에 대한 정보를 입력할 수 있다. 회원가입에 성공하면, 해당 아이디와 비밀번호를 입력해 로그인함으로써 메인 화면에 접근할 수 있게 된다. 메인 화면에는 "학점관리 / 자치단체 / 동아리 / 대외활동 / 준비위원회" 버튼이 존재한다. 각각의 버튼을 누르면 그에 해당하는 Widget들이 나타나고, 그곳에서 정보를 입력할 수 있다. 그리고 그렇게 입력한 정보를 바탕으로 "Portfolio" Widget에서 학년별로 자신이 무엇을 했는지를 visual하게 확인할 수 있다.

프로그램 구조 및 알고리즘

* Login : Login 클래스는 로그인 창 의 GUI를 구현하는 클래스이다. Member 클래스의 코드를 이용하여 로그인, 로그아웃, 회원가입 등의 기능이 구현되어 있다. 로그인이 되어 "P-Manager" 버튼을 누르면 Access 화면을 띄우고 "P-Manager"의 기능을 사용할 수 있다.



* Access : Access 클래스는 로그인 화면에서 성공적으로 로그인한 후에 띄울 수 있는 위젯이다. P-manager에 접근하면 처음으로 뜨는 창으로 6개의 버튼이 있고 각각을 누르면 SIGNAL을 mainwindow에 전달해 알맞은 창(학점관리, 자치단체, 동아리, 대외활동, 준비위원회, 포트폴리오)을 띄운다.



- * Courses : Courses 클래스는 학점관리 탭의 기능을 구현한다.
- inf 3차원 배열에서 첫번째 index는 8학기를, 두번째 index는 10개의 과목, 세번째 index는 0에 과목명을 담고, 1에 단위, 2에 학점, 3에 계열 정보를 담는다.
 - Size_per_semester는 8개의 학기 별 과목 수를 데이터로 가진다.
 - Sum_per_category는 7개 계열과 복수전공 과목 각각의 이수 현황을 데이터로 하는 배열이다.
 - Dual_major 변수는 true 값을 가지면 복수전공자이고 false 이면 그렇지 않음을 알려준다.
 - Read_data는 로그파일을 읽어와 클래스의 데이터 멤버를 채운다.
 - Write_data는 클래스의 데이터 멤버를 가지고 로그파일을 쓴다.
 - Sum_category는 inf의 정보를 계열별로 구분해 합을 구하여 sum_per_category에 데이터를 넣는다.
 - Get_requirements는 입력한 학과에 해당하는 졸업요건을 알려준다.

Courses

```
void read_data(std::string);  
void write_data(std::string);  
int* sum_category();  
void get_requirements(int*, std::string&);  
std::string const get_inf(int, int, int);  
void set_inf(const std::string& str, int, int, int);  
std::string get_inf();  
void increment_size(int);  
bool& get_dual_major();  
int get_dual_req(std::string&);  
void sort();
```

```
string inf[8][10][4];  
int size_per_semester[8];  
int sum_per_category[8];  
bool dual_major;
```

* CourseWindow : CourseWindow 클래스는 학점관리 탭의 Gui를 구현하는 위젯 클래스이다.

- on_pushButton_clicked1 슬롯은 저장버튼을 누르면 Gui 상에 입력한 내용을 course클래스에 저장해준다.
- on_checkBox_stateChanged는 복수전공 체크박스를 체크했을 때 신호를 받는 슬롯으로 course에 이를 저장해준다.

CoursesWindow

```
explicit CourseWindow(QWidget* parent = nullptr);
```

ui

SLOTS:
void on_pushButton_clicked1();
void on_checkBox_stateChanged(int arg1);

* Member : Member 클래스는 회원가입한 멤버의 정보를 저장하고 멤버의 학점, 동아리 등과 관련된 정보를 클래스 포인터를 이용해 저장, 접근한다.

Member

```
Member(string id,string pw,string dept);  
    Member(string id,string pw,string dept, string  
_dual_dept);  
    Member():ID(""),Password(""),department("");  
    string& get_id();  
    string& get_pw();  
    string& get_dept();  
    string& get_dual_dept();  
    bool information_check(string id, string password);  
    Courses* get_courses();
```

```
string ID;  
string Password;  
string department;  
string dual_dept;  
Courses* courses;  
Sumin::ExtraCourse* municipality, *extra,  
*club;Sumin::Portfolio* portfolio;
```

* MemberCollection : MemberCollection 클래스는 멤버 클래스 포인터의 배열을 가지고 있고 메소드를 이용해 이 멤버들을 모두 관리한다. 회원가입, 로그인, 로그아웃, 회원탈퇴 등이 이루어지고 프로그램을 켜다 켜도 회원들의 정보가 사라지지 않게 로그 파일을 출력, 입력 한다.

MemberCollection

```
MemberCollection();  
bool Login(string id, string password);  
bool Logout();  
bool is_logged_in();  
bool Register(string id, string password, string dept, string  
dual_dept = "");  
bool Unsubscribe(string id, string password);  
Member* get_logged_in_member();  
void write_log();  
void read_log();
```

```
Member* Members[20];  
int n_members;  
int index_of_logged_in_member;
```

* Club : 동아리에 대한 정보를 담는 클래스이며, 이 클래스에서는 다음과 같은 기능을 수행한다.

1. 동아리 정보 추가 : 사용자가 textEdit란에 자신의 동아리명, 동아리에서의 직책, 활동했던 학기를 체크박스로 표시하고, Apply를 클릭하면, 아래의 표에 정보가 표시되며, 이를 Save하면 로그파일로 출력된다. 로그 입출력 형태는 다음과 같다.

[학년] [1학기 활동여부-> 1이면 활동 0이면 비활동] [2학기 활동여부]

[동아리명]

[직책]

2. 회원 정보 Load : 사용자가 처음 설정화면에서 동아리 메뉴를 클릭하면, 클릭과 동시에 로그에 입력되어있는 정보가 다음과 같은 순서로 Load된다. [이 과정은 Club 페이지의 widget에서 실행된다. 정보는 Club에 담는다.]

가. commmandLog_Club_[ID].txt 파일을 열고 getline을 통해 한 줄씩 읽어 해당하는 학년탭의 표에 추가한다.

나. 모든 정보를 입력하였다면 파일을 닫는다.

* Extra_Course, Municipality : 담는 정보의 구분만 달라졌으며, 전체적인 동작 과정 및 기능은 위와 동일하다.

* Committee : Committee 클래스는 준비위원회 정보를 기록하여 체크할 수 있는 Widget 클래스이다. 1학년 / 2학년 / 3학년 / 4학년으로 구분되어 있으며 상반기에는 "축준위 / 엠준위 / 생준위"를 체크할 수 있고 하반기에는 "포준위 / 새준위 / 졸준위 / 엠준위 / 생준위"를 체크할 수 있다. 체크박스를 클릭하면, log를 출력할 때 체크박스의 상태를 1이라고 출력한다. UI에서 체크박스 아래에는 "역할"버튼이 있는데, 이 버튼을 클릭하면 새로운 widget이 나오게 되고 그곳에 정보를 입력할 수 있다. 오른쪽 상단부에는 "Save"버튼이 있으며, 이 버튼을 눌렀을 때 로그 파일이 만들어진다.

Committee

```
explicit Committee(string, QWidget* parent=nullptr);
~Committee();
void role(QWidget* widget, QTextEdit* textEdit, QLineEdit* lineEdit,
QPushButton* pushButton);
void DoTask();
```

```
Ui::committee* ui;
string str1[8], str2[8],str3[8],str4[8];
string chk1[8], chk2[8],chk3[8],chk4[8];
QWidget *widget1[8], *widget2[8], *widget3[8], *widget4[8];
QTextEdit *textEdit1[8], *textEdit2[8], *textEdit3[8], *textEdit4[8];
QLineEdit *lineEdit1[8], *lineEdit2[8], *lineEdit3[8], *lineEdit4[8];
QPushButton *pushButton1[8], *pushButton2[8], *pushButton3[8], *pushButton4[8];
int n;
ofstream output;
ifstream input;
string test;
stringstream ss;
string tokens[4];
string enter, name;
```

```
SIGNALS:
void role_widget11();
void role_widget12();
void role_widget13();
void role_widget14();
void role_widget15();
void role_widget16();
void role_widget17();
void role_widget18();
void role_widget21();
void role_widget22();
void role_widget23();
void role_widget24();
void role_widget25();
void role_widget26();
void role_widget27();
void role_widget28();
void role_widget31();
void role_widget32();
void role_widget33();
void role_widget34();
void role_widget35();
void role_widget36();
void role_widget37();
void role_widget38();
void role_widget41();
void role_widget42();
void role_widget43();
void role_widget44();
void role_widget45();
void role_widget46();
void role_widget47();
void role_widget48();
void save1();
void save2();
void save3();
void save4();
```

```
SIGNALS:
void hide0();
void save_str1();
void role11();
void role12();
void role13();
void role14();
void role15();
void role16();
void role17();
void role18();
void role21();
void role22();
void role23();
void role24();
void role25();
void role26();
void role27();
void role28();
void role31();
void role32();
void role33();
void role34();
void role35();
void role36();
void role37();
void role38();
void role41();
void role42();
void role43();
void role44();
void role45();
void role46();
void role47();
void role48();
void on_pushButton_7_clicked();
void on_pushButton_2_clicked();
void on_pushButton_3_clicked();
void on_pushButton_8_clicked();
void on_pushButton_6_clicked();
void on_pushButton_1_clicked();
```

```
void on_pushButton_4_clicked();
void on_pushButton_5_clicked();
void on_pushButton_16_clicked();
void on_pushButton_10_clicked();
void on_pushButton_13_clicked();
void on_pushButton_9_clicked();
void on_pushButton_14_clicked();
void on_pushButton_15_clicked();
void on_pushButton_11_clicked();
void on_pushButton_12_clicked();
void on_pushButton_21_clicked();
void on_pushButton_22_clicked();
void on_pushButton_18_clicked();
void on_pushButton_19_clicked();
void on_pushButton_33_clicked();
void on_pushButton_27_clicked();
void on_pushButton_28_clicked();
void on_pushButton_25_clicked();
void on_pushButton_29_clicked();
void on_pushButton_31_clicked();
void on_pushButton_32_clicked();
void on_pushButton_50_clicked();
void on_pushButton_51_clicked();
void on_pushButton_52_clicked();
void on_pushButton_53_clicked();
```

```
void on_checkBox_11_stateChanged(int arg1);
void on_checkBox_12_stateChanged(int arg1);
void on_checkBox_13_stateChanged(int arg1);
void on_checkBox_14_stateChanged(int arg1);
void on_checkBox_15_stateChanged(int arg1);
void on_checkBox_16_stateChanged(int arg1);
void on_checkBox_17_stateChanged(int arg1);
void on_checkBox_18_stateChanged(int arg1);
void on_checkBox_21_stateChanged(int arg1);
void on_checkBox_22_stateChanged(int arg1);
void on_checkBox_23_stateChanged(int arg1);
void on_checkBox_24_stateChanged(int arg1);
void on_checkBox_25_stateChanged(int arg1);
void on_checkBox_26_stateChanged(int arg1);
void on_checkBox_27_stateChanged(int arg1);
void on_checkBox_28_stateChanged(int arg1);
void on_checkBox_31_stateChanged(int arg1);
void on_checkBox_32_stateChanged(int arg1);
void on_checkBox_33_stateChanged(int arg1);
void on_checkBox_38_stateChanged(int arg1);
void on_checkBox_37_stateChanged(int arg1);
void on_checkBox_36_stateChanged(int arg1);
void on_checkBox_34_stateChanged(int arg1);
void on_checkBox_35_stateChanged(int arg1);
void on_checkBox_41_stateChanged(int arg1);
void on_checkBox_42_stateChanged(int arg1);
void on_checkBox_43_stateChanged(int arg1);
void on_checkBox_44_stateChanged(int arg1);
void on_checkBox_45_stateChanged(int arg1);
void on_checkBox_46_stateChanged(int arg1);
void on_checkBox_47_stateChanged(int arg1);
void on_checkBox_48_stateChanged(int arg1);
```

* Portfolio : 이 Widget에서는 사용자의 대학 생활을 한눈에 볼 수 있게 한다. 해당 하는 사용자의 정보를 받아오는 과정은 이 사용자의 로그 파일을 사용한다. 이수과목, 동아리, 자치단체, 준비위원회, 대외활동 총 4가지의 정보를 보여주며, 이수과목은 최대 20개, 동아리 최대 2개, 자치단체 최대 2개, 준비위원회 및 대외활동은 각각 최대 4개씩 보여준다. 학기당이 아닌 매 학년을 기준으로 보여준다. 최대 항목수가 제한되어 있는데, 이는 우선 로그파일에 작성된 순서를 제 1 우선순위로 판별하여 정보를 나타낸다. 만약, 활동 수가 부족할 때는 공란인 상태로 둔다. 다음은 로그 파일에서 정보를 읽어오는 방법이다. 이수과목, 동아리, 자치단체, 준비위원회와 대외활동의 정보를 읽어오는 방법은 정보의 종류만 다를 뿐, 전체적인 컨셉은 같다.

가. 해당하는 정보가 담긴 로그파일을 열고, 정해진 기준에 따라 먼저 정보를 판별한다.

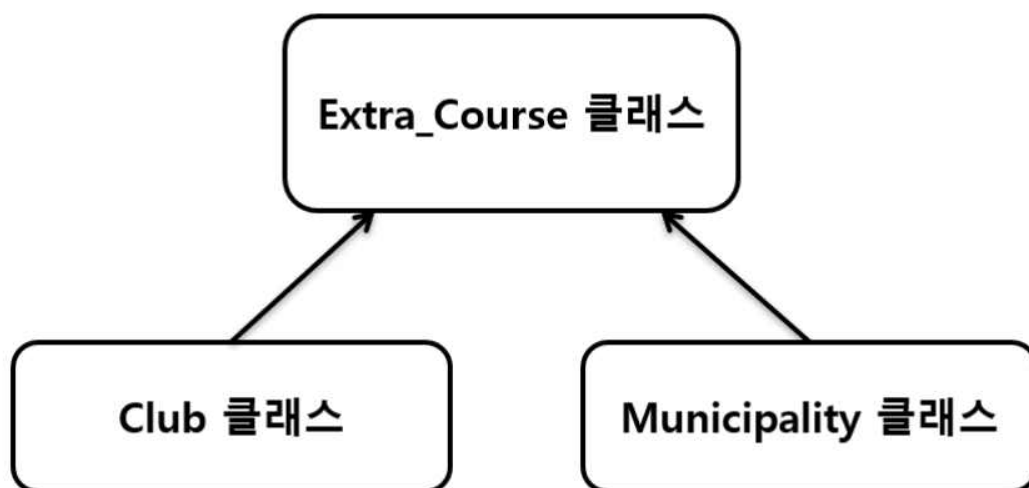
* 이수과목 : 기초필수 -> 기초선택 -> 전공필수 -> 전공선택 -> 교양필수 -> 교양선택 -> 자유선택 -> 복수전공

* 이수과목 외 항목: 1학년 -> 2학년 -> 3학년 -> 4학년

나. 각 항목들은 최대 항목수가 제한되어있으므로, 정보를 담을 때마다, count를 세어 제한 count가 되는 순간 while문을 종료한다.

다. 이 과정은 텍스트 파일을 처음부터 끝까지 다 읽고, 이를 통해 해당하는 정보를 얻는다.

클래스 상속관계



기타 프로그램을 이해하는데 필요한 내용

학점관리 widget에서 과목의 계열을 입력할 때, 다음과 같이 입력을 진행할 수 있다.

계열 입력 Manual

"교양필수" "교양 필수" "교필"
"교양선택" "교양 선택" "교선"
"기초필수" "기초 필수" "기필"
"기초선택" "기초 선택" "기선"
"전공필수" "전공 필수" "전필"
"전공선택" "전공 선택" "전선"
"자유선택" "자유 선택" "자선"
"복수전공" "복수 전공" "복전"

위에서 첫 번째 줄을 보면, "교양필수" "교양 필수" "교필"이라고 나와 있다. 즉, 과목의 계열을 적을 때, "교양필수"라고 적어도 되고, "교양 필수"라고 적어도 되며 "교필"이라고 적어도 된다.

회원가입 화면에서 학과의 이름을 적을 때, 복수전공 table에 학과의 이름을 적을 때는 다음과 같이 입력을 진행할 수 있다.

학과 입력 Manual

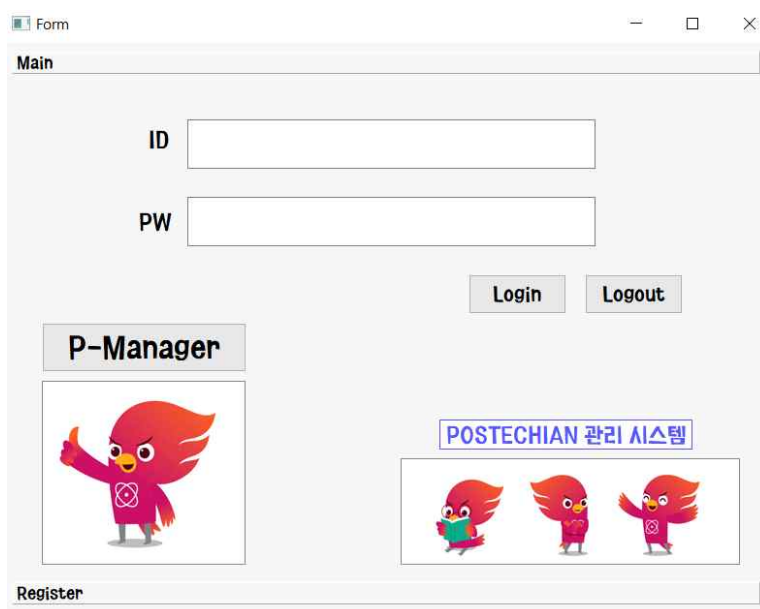
"기계" "기계과" "기계 공학과" "기계공학과" "기공"
"물리" "물리과" "물리 학과" "물리학과"
"산경" "산경과" "산업 경영 공학과" "산업경영공학과"
"생명" "생명 공학과" "생명공학과" "생명과"
"수학과" "수학"
"소재" "소재과" "신소재 공학과" "신소재공학과" "신소재"
"신소재과"
"전자" "전자과" "전자 공학과" "전자공학과"
"창공" "창의IT융합공학과" "창의it융합공학과" "창의IT공학과"
"창의it공학과" "창공과"
"컴공" "컴공과" "컴퓨터 공학과" "컴퓨터공학과"
"화학" "화학과"
"화공" "화학 공학과" "화학공학과" "화공과"

회원관리에 대한 정보와 각 회원이 "학점관리 / 자치단체 / 동아리 / 대외활동 / 준비위원회"에 입력한 정보는 로그로 작성하여 .txt파일 형식으로 저장하게 된다. 그리고 메인화면에서 "학점관리 / 자치단체 / 동아리 / 대외활동 / 준비위원회"에 해당하는 버튼을 눌렀을 때, 이 .txt파일을 읽어서 데이터를 자동으로 집어넣는다. .txt 파일의 이름은 회원마다 구별하기 위해 "_memberName"을 추가하였다.

- Members.txt
- courseslog_junhyeong.txt
- commandLog_Municipality_junhyeong.txt
- commandLog_ExtraCourse_junhyeong.txt
- commandLog_Committee_junhyeong.txt
- commandLog_Club_junhyeong.txt

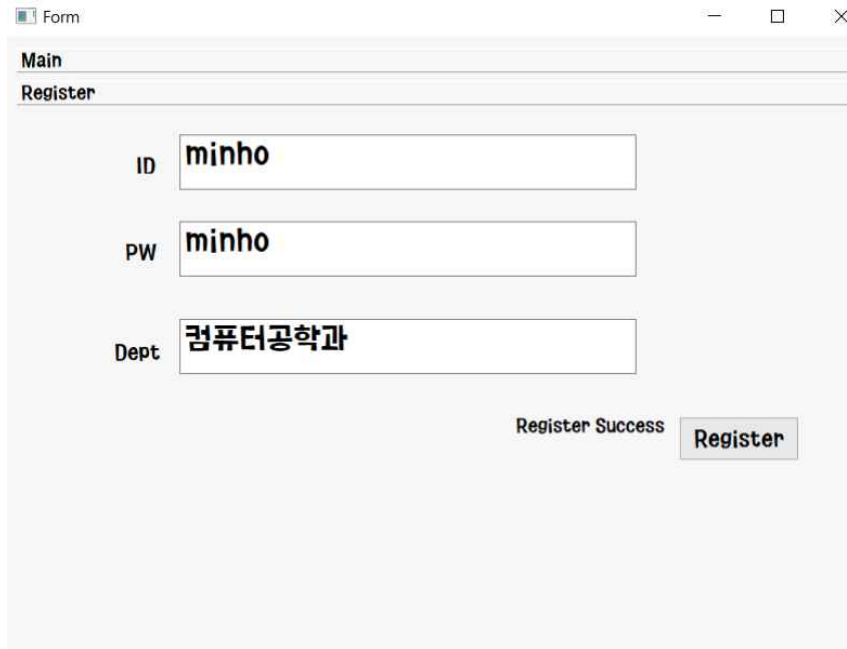
즉, 회원의 이름이 "junhyeong"이라면 위의 사진처럼 "_junhyeong"이 붙은 .txt파일 일이 만들어지는 것이다. 로그 파일은 각 widget에서 오른쪽 상단부의 "Save" 버튼을 눌렀을 때 생성된다.

실행 방법



맨 처음에 프로그램을 실행하면, 위와 같은 화면이 뜬다. 왼쪽 하단의 Register 버

튼을 누르면 회원가입을 진행할 수 있다.



Form

Main

Register

ID minho

PW minho

Dept 컴퓨터공학과

Register Success

Register

회원가입에 성공한 모습이다.



Form

Main

ID minho

PW minho

Login Logout

Login Success

P-Manager

POSTECHIAN 관리 시스템

Register

그런 다음에 Login 버튼을 누르면, "Login Success"라는 메시지가 나온다. 이 상태에서 왼쪽 중앙에 있는 "P-Manager" 버튼을 클릭하면 메인 화면으로 접근할 수 있다.

Form

학점관리

자치단체


동아리

대외활동

준비위원회

포트폴리오

P-Manager



Logged in as: minho

회원 탈퇴

메인 화면은 위와 같다. “학점관리 / 자치단체 / 동아리 / 대외활동 / 준비위원회 / 포트폴리오” 버튼이 있으며, 각 버튼을 누를 때 그에 해당하는 Widget이 나온다.

예제

MainWindow

1학기 2학기 3학기 4학기 5학기 6학기 7학기 8학기

수강과목

과목명	단위	학점	비고
일반물리실험1	1	P	기초필수
일반생명과학	3	A-	기초필수
프로그래밍과 문제해결	3	A+	기초필수
체육관리	1	S	교양필수
인문기술융합	3	A0	전공필수
미적분학1	3	A+	기초필수
대생생1	1	S	교양필수

Save

☒ 복수전공

졸업요건

학과	창의IT융합공학과
필수구분	이수 총
교양필수	2 13
교양선택	0 18
기초필수	10 25
기초선택	0 5
전공필수	3 29
전공선택	0 31
자유선택	0 0

복수 전공 0 이수 43 총

학과 산경과

학점관리 Widget에서 정보를 입력한 화면이다.

MainWindow

1 2 3 4

자치단체

이름: 중립위

역할: 부원

1학기

2학기

Save

Apply

1학기	이름	부원	역할
1.	중립위	부원	
2.			
3.			
4.			

2학기	이름	부원	역할
1.	중립위	부원	
2.			
3.			
4.			

자치단체 Widget에서 정보를 입력한 화면이다.

MainWindow

1 2 3 4

동아리

이름: 권디

역할: 부원

1학기

2학기

Save

Apply

1학기	이름	부원	역할
1.	권디	부원	
2.			
3.			
4.			

2학기	이름	부원	역할
1.			
2.			
3.			
4.			

동아리 Widget에서 정보를 입력한 화면이다.

대외활동 Widget에서 정보를 입력한 화면이다.

준비위원회 Widget에서 정보를 입력한 화면이다.

포트폴리오 Widget에서 학년별로 정보를 시각적으로 보여주는 화면이다. (학점관리 Widget에서 2학기에도 정보를 넣어놓았다.)

제한 사항

- * 회원가입을 통해 등록한 회원 수의 Maximum은 20이다.
- * 매 학기당 입력할 수 있는 과목 수의 Maximum은 10이다.
- * 매 학기당 입력할 수 있는 자치단체 수의 Maximum은 4이다.
- * 매 학기당 입력할 수 있는 동아리 수의 Maximum은 4이다.
- * 매 학기당 입력할 수 있는 준비위원회 수의 Maximum은 4이다.
- * 정보를 입력할 때, "띄어쓰기" 대신에 "_"를 사용한다(공백으로 token을 구분).
(학점관리 화면에서는 띄어쓰기를 해도 된다.)
- * 포트폴리오 Widget에서 각 학년 당 visual하게 보여주는 동아리의 수 / 자치위원회의 수는 2개씩이며, 준비위원회의 수 / 대외활동의 수는 4개씩이다.
(기존에 저장되어 있는 데이터에서 정렬된 순서대로 위에서부터 차례대로 정보가 들어가게 된다.)

Time-Line

기본적으로 매주 화요일과 일요일 9시~11시에 만나서 회의를 지속적으로 진행했다. 모임 때마다 여태까지 해온 것들을 정리하고, 다음 모임 때 각자가 해야 할 부분들을 정하고 헤어졌다. 그 구체적인 일정은 다음과 같다.

5월 7일 : 전체적인 인터페이스 구상

5월 12일 : 회원가입, 로그인 구현 / "학점 관리" TAB에서 과목명, 학점, 구분 등의 데이터 저장 / "동아리&자치단체" TAB과 "교외 활동" TAB에서 데이터 저장 / "준비 위원회" TAB 데이터 저장 (체크박스 형식)

5월 14일 : "학점 관리" TAB에서 졸업요건 GUI 구현 / "학점 관리" TAB, "동아리&자치단체" TAB, "교외 활동" TAB, "준비 위원회" TAB에서 Log 저장하는 기능 구현

5월 15일 : 중간발표 준비

5월 21일 : 보완하기

5월 24일 : 예외 처리 / Visual Map으로 시각화

5월 29일 : 디버깅

6월 5일 : 디버깅

6월 6일 : 프로그램 완성 및 최종발표 준비

역할 분배

강민호 : 인터페이스 구상, "학점관리" 구현, "로그인창" 구현, Member / Member Collection 클래스 디자인

송수민 : 인터페이스 구상, "동아리" 구현, "자치단체" 구현, "대외활동" 구현, "포트폴리오" 구현

조준형 : 프로젝트 아이디어 구상, proposal 및 보고서 작성, ppt 제작, 인터페이스 구상, "준비위원회" 구현, UI 디자인, 코드 수합 및 연결(클래스 간의 관계 수정), 영상 제작

프로그램 제작 기여도

처음에 역할을 나눌 때, 각자 비슷한 분량을 갖도록 하였다. 주기적으로 만나서 프로그램을 구현하였으며, 서로 코드를 작성하며 어려움이 있을 때는 적극적으로 도와주었다. 협업이 매우 잘 이루어졌고, 각자가 맡은 기능들을 구현하기 위해 소요한 시간도 비슷하다. 이러한 이유로, 모든 팀원이 이 프로그램 제작에 동일하게 기여했다고 합의했다.

강민호 : 33.33333% (1/3)

송수민 : 33.33333% (1/3)

조준형 : 33.33333% (1/3)

토론

1. 학번마다 졸업요건이 달라서, "학점 관리" TAB에서 default로 생성되는 졸업요건을 18학번 기준으로 잡았다.

2. "학점 관리" TAB에는 복수전공을 체크할 수도 있다. 복수전공을 선택할 시에는, 본과의 졸업요건에는 해당하지 않고 복수전공의 졸업요건에만 해당하는 과목들을 분류에 "복전"이라고 적어 정보를 입력한다.

ex) 컴퓨터공학과 학생이 산업경영공학과를 복수전공하는 경우, "공학기초통계"는 컴퓨터공학과와 전공필수 과목이므로 분류에 "전필"이라고 적는다. 그러나, "재무회계"는 산업경영공학과와 전공필수 과목이므로 분류에 "복전"이라고 적는다. 이 둘을 따로 관리하는 이유는, 본과의 졸업요건과 복수조건을 충족시키는 요건을 모두 사용자에게 visual하게 보여주기 위함이다.

3. 각 팀원들이 클래스와 widget을 만들었기 때문에, 이를 수합하고 연결하는 과정에서 어려움이 있었다. 그래서 클래스 간의 관계들과 구조들에 많은 수정을 가했고, 최대한 QObject::connect()을 이용해 정보를 주고받았다. 이를 통해 문제를 해결할 수 있었다.

결론

- 한 학기 동안 배운 C++ 지식을 모두 접목시켜 Qt creator를 이용해 프로그램을 구현할 수 있었다. 기존의 Command 창과는 달리, 직접 UI를 디자인할 수 있어서 생각했던 디자인과 그에 해당하는 기능들을 모두 구현할 수 있었다.
- "POSTECHIAN 관리 시스템"을 만듦으로써, 학기/학년별로 어떤 활동들을 했는지를 체계적으로 기록할 수 있게 되었다. 이는 매우 실용적이어서 실제로도 사용할 계획이며, 친구들에게 이 프로그램을 보급할 의향도 있다.