

REPORT

전자공학도의 윤리 강령 (IEEE Code of Ethics)

(출처: <http://www.ieee.org>)

나는 전자공학도로서, 전자공학이 전 세계 인류의 삶에 끼치는 심대한 영향을 인식하여 우리의 직업, 동료와 사회에 대한 나의 의무를 짐에 있어 최고의 윤리적, 전문적 행위를 수행할 것을 다짐하면서, 다음에 동의한다.

1. **공중의 안전, 건강 복리에 대한 책임:** 공중의 안전, 건강, 복리에 부합하는 결정을 할 책임을 질 것이며, 공중 또는 환경을 위협할 수 있는 요인을 신속히 공개한다.
2. **지위 남용 배제:** 실존하거나 예기되는 이해 상충을 가능한 한 피하며, 실제로 이해가 상충할 때에는 이를 이해 관련 당사자에게 알린다. (이해 상충: conflicts of interest, 공적인 지위를 사적 이익에 남용할 가능성)
3. **정직성:** 청구 또는 견적을 함에 있어 입수 가능한 자료에 근거하여 정직하고 현실적으로 한다.
4. **뇌물 수수 금지:** 어떠한 형태의 뇌물도 거절한다.
5. **기술의 영향력 이해:** 기술과 기술의 적절한 응용 및 잠재적 영향에 대한 이해를 높인다.
6. **자기계발 및 책무성:** 기술적 능력을 유지, 증진하며, 훈련 또는 경험을 통하여 자격이 있는 경우이거나 관련 한계를 전부 밝힌 뒤에만 타인을 위한 기술 업무를 수행한다.
7. **엔지니어로서의 자세:** 기술상의 업무에 대한 솔직한 비평을 구하고, 수용하고, 제공하며, 오류를 인정하고 수정하며, 타인의 기여를 적절히 인정한다.
8. **차별 안하기:** 인종, 종교, 성별, 장애, 연령, 출신국 등의 요인에 관계없이 모든 사람을 공평하게 대한다.
9. **도덕성:** 허위 또는 악의적인 행위로 타인, 타인의 재산, 명예, 또는 취업에 해를 끼치지 않는다.
10. **동료애:** 동료와 협력자가 전문분야에서 발전하도록 도우며, 이 윤리 헌장을 준수하도록 지원한다.

위 IEEE 윤리헌장 정신에 입각하여 report를 작성하였음을 서약합니다.

학 부: 전자공학부

마감일: 2021. 06. 22.

과목명: 전자공학프로그래밍

교수명: 이정원 교수님

학 번: 201620837

성 명: 안민규

1. 함수 기능 및 코드 분석

1) init

```
package assignment5;

public class init {
    public static void Init_Program() {
        management.students.add(new student("유동연", "dongs0125", "1234"));
        management.students.add(new student("장현웅", "sas1200", "1234")); //학생 선언

        management.professors.add(new professor("이미연", "mylee", "1234"));
        management.professors.add(new professor("이정원", "jungwony", "1234"));
        management.professors.add(new professor("이세돌", "saedol", "1234")); //교수님 선언

        management.professors.get(0).register_lecture("전프", "월B목B");
        management.professors.get(0).register_lecture("전프", "수B금B");

        management.professors.get(1).register_lecture("자료구조", "화A금A");
        management.professors.get(1).register_lecture("자료구조", "화C금C");

        management.professors.get(2).register_lecture("C++", "월B금B");
        management.professors.get(2).register_lecture("Python", "화C금C"); //교수님의 강의 목록 선언
    }
}
```

학생과 교수님, 수업 등을 미리 선언하여 이들을 management 안에 저장한다. 프로그램 실행 전에 학생과 교수님을 미리 선언하여야 프로그램 시 로그인이 가능하다. 미리 선언하지 않은 사람이나 강의 경우 프로그램 내에서 회원가입과 강의등록을 통해 등록이 가능하다.

2) lecture

```
package assignment5;

public class lecture {
    private String lecture_name;
    private String lecture_prof;
    private String lecture_time; //문자열을 저장하기 위해 선언
    private int time_num1;
    private int time_num2; //숫자 저장하기 위해 선언

    public lecture(String lecture_name, String professor, String time) {
        this.lecture_name = lecture_name;
        this.lecture_prof = professor;
        this.lecture_time = time; //메소드를 선언할 때 들어오는 매개변수를 저장
        time_num1 = day_to_int(time.substring(0,1)) * 5 +
alphabet_to_int(time.substring(1,2));
        time_num2 = day_to_int(time.substring(2,3)) * 5 +
alphabet_to_int(time.substring(3,4)); //매개변수 time의 substring을 보고 등록되어있는 요일과 교시를 저장
    }

    public String get_name()
    {
        return this.lecture_name; //강의 이름을 리턴
    }
    public String get_prof()
    {
        return this.lecture_prof; //강의를 하시는 교수님의 이름을 리턴
    }
    public String get_time()
    {
        return this.lecture_time; //강의 시간을 리턴
    }
}
```

```

public int get_time1()
{
    return this.time_num1; //time1을 리턴
}
public int get_time2()
{
    return this.time_num2; //time2을 리턴
}

public static int day_to_int(String day) //매개변수 day를 숫자로 변경
{
    if(day.equals("월"))
        return 0;
    else if(day.equals("화"))
        return 1;
    else if(day.equals("수"))
        return 2;
    else if(day.equals("목"))
        return 3;
    else if(day.equals("금"))
        return 4;
    else
    {
        System.out.println("잘못된 입력입니다. (월~금)으로 입력해주세요"); //강의
        등록 시 월~금이 아닌 다른 요일 입력했을 시 출력 -> 이클립스 내에서 출력
        return -1;
    }
}

public static int alphabet_to_int(String alphabet) //교시를 숫자로 변경
{
    if(alphabet.equals("A"))
        return 0;
    else if(alphabet.equals("B"))
        return 1;
    else if(alphabet.equals("C"))
        return 2;
    else if(alphabet.equals("D"))
        return 3;
    else if(alphabet.equals("E"))
        return 4;
    else
    {
        System.out.println("잘못된 입력입니다. (A~E)으로 입력해주세요"); //강의
        등록 시 A~E교시가 아닌 다른 교시 입력했을 시 출력
        return -1;
    }
}
}

```

강의와 관련된 클래스로 강의가 있는 요일, 교시, 강의 이름, 교수님 성함을 저장하고 요일과 교시는 숫자로 변경하여 계산해놓고, 올바르지 못한 요일과 교시가 선언되면 이에 대한 문구를 출력한다. 또한 교수님의 성함, 강의 이름, 요일과 교시를 정해진 식에 따라 치환한 값을 반환하는 메소드를 다른 클래스 내에서 목록을 만들거나할 때 사용한다.

3) login

```

package assignment5;

public class login {
    static String id, pw, name; //id, pw, name을 저장하기 위함

    public static boolean check_duplicate_id(String id) //아이디 중복을 확인하는 메소드
    {
        for(int i = 0; i < management.students.size(); i++)
        {
            if(id.equals(management.students.get(i).get_id())) //management에 있는
            학생의 이름과 같으면

```

```

        return false; //false를 반환
    }

    for(int i = 0; i < management.professors.size(); i++)
    {
        if(id.equals(management.professors.get(i).get_id())) //management에 있는
교수님의 성함과 같으면
        return false; //false를 반환
    }

    return true; //그렇지 않다면 true
반환
}

public static member check_id_pw(String id, String pw) //입력된 아이디와 비밀번호가
저장되어있는 아이디와 비밀번호와 맞는지를 확인하고 member 형을 반환하는 메소드
{
    for(int i = 0; i < management.students.size(); i++)
    {
        if(id.equals(management.students.get(i).get_id()) &&
pw.equals(management.students.get(i).get_pw())) //학생의 아이디와 비밀번호가 매개변수로 들어온
값과 같다면
        return management.students.get(i); //맞는 학생을
반환
    }

    for(int i = 0; i < management.professors.size(); i++)
    {
        if(id.equals(management.professors.get(i).get_id()) &&
pw.equals(management.professors.get(i).get_pw())) //교수님의 아이디와 비밀번호가 매개변수로 들어온
값과 같다면
        return management.professors.get(i); //맞는 교수님을
반환
    }

    return null; //같은 사람이 없으므로 매개변수가 잘못 들어온 것이므로 null 반환
}
}

```

login 클래스는 두 가지 메소드를 이용하는데 둘 다 매개변수로 들어오는 값과 management에 저장되어 있는 값과 비교를 한다. check duplicate id 메소드는 id를 중복으로 사용하게되는 것을 방지하기 위한 메소드이고, check id pw 메소드는 매개변수로 들어오는 id와 pw가 member이 id와 pw가 맞는지를 판별하여 맞다면 그 member를 반환해주는 메소드이다.

4) LoginScreen

```

package assignment5;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.event.ActionListener;

public class LoginScreen extends JFrame implements ActionListener{

    JTextField ID = new JTextField(20); //id 입력 위한 텍스트필드 생성
    JTextField PW = new JTextField(20); //pw 입력 위한 텍스트필드 생성
    JLabel icon = new JLabel();
    JLabel text1 = new JLabel();
    JLabel text2 = new JLabel(); //레이블 생성
    JButton button1 = new JButton("Login");
    JButton button2 = new JButton("New"); //버튼 2개 생성

    public LoginScreen() {
        Container c1 = getContentPane(); //컨테이너 생성
        ImageIcon ajou = new ImageIcon("assignment5/ajou2.png"); //아이콘 생성
    }
}

```

```

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //화면에 x 자 생성

GridLayout b1 = new GridLayout(7,1,10,10); //gridlayout 사용
c1.setLayout(b1);

icon.setIcon(ajou);
icon.setText("아주대학교 수강신청 프로그램 입니다."); //icon에 text를 가져다 씀

text1.setText("ID를 입력해 주세요");
text2.setText("PW를 입력해 주세요"); //앞서 선언한 레이블에 문구 저장

button1.setText("Login");
button2.setText("회원가입"); //버튼 2개에 문구 저장

c1.add(icon);
c1.add(text1);
c1.add(ID);
c1.add(text2);
c1.add(PW);
c1.add(button1);
c1.add(button2); //앞서 선언한 요소들 컨테이너에 추가

button1.addActionListener(this);
button2.addActionListener(this); //버튼 누르면 이벤트 발생하므로 listener 선언

ID.addActionListener(this);
PW.addActionListener(this); //id와 pw 입력 시 이벤트 발생하므로 listener 선언

setTitle("수강신청 프로그램"); //창의 이름은 수강신청 프로그램
setSize(500,400);
setVisible(true);
}
public void actionPerformed(ActionEvent ae)
{
    member Member;
    int flag;
    String id = ID.getText();
    String pw = PW.getText(); //텍스트필드에 입력된 값 저장

    if(ae.getActionCommand() == "Login") { //Login 눌렀다면

        id = ID.getText();
        pw = PW.getText();

        Member = login.check_id_pw(id, pw); //입력된 id, pw를 가지고 맞는지
check해서 member에 저장

        if(Member == null) //만약 null이라면 잘못 입력
        {
            flag = 0; //flag를 0으로 설정
            new New_Window(flag); //new_window 메소드 매개 변수 0으로
호출

        }
        else
        {
            dispose();
            management.now_member = Member; //management.now
member에 저장

            if(management.now_member instanceof student) //now member가
student에 속하면

                new StudentScreen(); //studentscreen 띄움
            else if(management.now_member instanceof professor)
//professor에 속하면

                new ProfessorScreen(); //professorscreen 띄움

        }
    }
}

```

```

    }
    else if(ae.getActionCommand() == "회원가입") { //회원가입이 눌렀다면
        new MakeNewScreen(); //makenewscreen 생성
    }
}

class New_Window extends JFrame{
    New_Window(int flag) {
        setTitle("Alert"); //Alert으로 이름 지정

        JPanel NewWindowContainer = new JPanel(); //Panel 생성
        setContentPane(NewWindowContainer);

        if(flag == 0) //매개변수 0이라면 아이디 틀린 것이므로
        {
            NewWindowContainer.add(new JLabel("입력한 정보와 일치하는 아이디는
존재하지 않습니다"));
            NewWindowContainer.add(new JLabel("다시 한번 입력해 주세요"));
        }
        else if(flag == 1) //매개변수 1이라면 회원가입시 중복된 아이디를 입력한 것이므로
        {
            NewWindowContainer.add(new JLabel("중복된 id가 존재합니다"));
            NewWindowContainer.add(new JLabel("다른 아이디를 사용해 주세요"));
        }
        else if(flag == 2) //매개변수 2라면 회원가입 성공
        {
            NewWindowContainer.add(new JLabel("회원가입이 완료 되었습니다!"));
        }
        setSize(300,100);
        setResizable(false);
        setVisible(true);
    }
}

class MakeNewScreen extends JFrame implements ActionListener{ //회원가입 클래스

    JLabel jl1 = new JLabel("당신의 신분은? ");
    ButtonGroup bgp = new ButtonGroup(); //그룹버튼 생성
    JRadioButton jrb1 = new JRadioButton("학생", true);
    JRadioButton jrb2 = new JRadioButton("교수"); //radio버튼 2개 생성

    JTextField NAME = new JTextField(5); //이름 입력 위한 텍스트필드
    JTextField ID = new JTextField(20); //아이디 입력
    JTextField PW = new JTextField(20); //비밀번호 입력
    JButton BTN = new JButton("회원가입"); //회원가입 버튼 생성

    public MakeNewScreen() {

        Container c1 = getContentPane();
        GridLayout g1 = new GridLayout(9,2,10,10);
        c1.setLayout(g1);

        setTitle("회원가입"); //이름은 회원가입

        bgp.add(jrb1);
        bgp.add(jrb2); //버튼그룹에 앞서 생성한 radio 버튼 2개 추가

        c1.add(jl1);
        c1.add(new JLabel(" "));
        c1.add(jrb1);
        c1.add(jrb2);
        c1.add(new JLabel("이름을 입력해 주세요"));
        c1.add(NAME);
        c1.add(new JLabel("ID 를 입력해 주세요"));
        c1.add(ID);
        c1.add(new JLabel("PW 를 입력해 주세요"));
        c1.add(PW);
        c1.add(new JLabel(" "));
    }
}

```

```

버튼 추가      c1.add(BTN); //순서대로 신분 버튼 -> 이름 -> 아이디 -> 비밀번호 -> 회원가입

                BTN.setText("회원가입"); //버튼 문구 회원가입으로 설정
                BTN.addActionListener(this); //버튼 눌리면 이벤트 발생하므로 listener 선언

                setSize(300,400);
                setResizable(false);
                setVisible(true);

            }
            public void actionPerformed(ActionEvent ae) {
                member Member;
                System.out.println(ae.getActionCommand());
                if(ae.getActionCommand() == "회원가입") { //회원가입 눌렀다면
                    if(!login.check_duplicate_id(ID.getText())) //아이디가 같다면
                    {
                        new New_Window(1); //회원가입 실패
                        ID.setText(""); //아이디 입력 창 비움
                    }
                    else {
                        new New_Window(2); //회원가입 성공
                        if(jrb1.isSelected() == true)
                        {
                            Member = new student(NAME.getText(), ID.getText(),
PW.getText());
//management에 저장
                            management.students.add( (student) Member);
                        }
                        else
                        {
                            Member = new professor(NAME.getText(), ID.getText(),
PW.getText());
//management에 저장
                            management.professors.add( (professor) Member);
                        }
                    }
                    this.dispose();
                }
            }
        }
    }
}

```

loginScreen 클래스는 프로그램을 실행했을 때 처음 나오는 창이다. 로그인해야 하는 창으로 이미 회원가입이 되어있는 상태라면 아이디와 비밀번호를 입력하여 로그인하고 등록되지 않은 사람이라면 회원가입을 하고 그 후 아이디와 비밀번호를 입력해서 프로그램을 사용한다. 기본적으로 입력하는 텍스트 필드에 무엇을 입력해야 하는지 명시적으로 알리고 버튼과 입력이 되는 텍스트 필드는 모두 ActionListener로 선언하였다. 눌린 버튼에 따라서 동작을 달리하는데 로그인 버튼이 눌렸을 때는 텍스트 필드에 입력된 문자열을 매개변수로 하여 login 클래스의 check id pw 메소드를 호출해 맞는지를 확인 후 맞다면 신분에 따라 창을 달리 보여주고 맞지 않는다면 다시 입력하라는 문구가 출력되게 설정하였다. 만약 회원가입 버튼이 눌렸다면 makeNewScreen 객체를 생성한다. new Window 클래스의 경우 매개변수 0~2에 따라 동작을 달리하는데 매개변수가 0이라면 아이디가 틀린 것으로 다시 입력하라는 문구가 들어있는 창을 띄운다. 1과 2는 회원가입 시 나오는데, 회원가입 시 중복된 아이디를 입력 시 1이, 회원가입 성공 시 2가 나온다. makeNewScreen은 회원가입 창으로 신분과 이름 아이디 pw를 입력받아 member에 저장한다. 이 때도 입력된 아이디가 같으면은 앞서 new Window를 매개변수 1로 하여 호출하고 성공 시 매개변수 2로 하여 new Window를 호출한다.

5) management

```

package ajou_service;

import java.util.Scanner;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

import java.util.ArrayList;

public class management {
    public static ArrayList<student> students = new ArrayList<student>(); //학생 저장 위한
    public static ArrayList<professor> professors = new ArrayList<professor>(); //교수님 저장
    public static ArrayList<lecture> lectures = new ArrayList<lecture>(); //강의 저장 위한 배열
    public static Scanner scanner = new Scanner(System.in); //입력받을 때
    //member클래스 형을 가지는 now_member 객체 선언
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        init.Init_Program(); //프로그램 실행
        new LoginScreen(); //로그인
    }
}

```

management 클래스로 학생, 교수님, 강의를 저장하기 위해 ArrayList 클래스를 이용해 배열을 선언하고 키보드로 입력하는 것을 받기 위해 scanner 클래스 또한 사용한다. main 문에서 앞서 init 클래스의 Init_Program() 메소드를 실행하고 LoginScreen 클래스를 호출해 실제 프로그램 이용 시 미리 init 클래스에서 선언한 것이 저장되어있는 상태로 login 화면이 나타난다.

6) member

```

package assignment5;

public class member {
    private String name;
    private String id;
    private String pw;

    public member(String name, String id, String pw)
    {
        this.name = name; //이름
        this.id = id; //아이디
        this.pw = pw; //비밀번호
    }

    public String get_id()
    {
        return this.id; //아이디를 리턴 -> login 클래스 내에서 확인 위해 사용
    }

    public String get_pw()
    {
        return this.pw; //비밀번호를 리턴 -> login 클래스 내에서 확인 위해 사용
    }

    public void change_pw(String pw) {
        this.pw = pw; //매개변수 비밀번호를 저장
    }

    public String get_name()
    {
    }
}

```



```

        return this.name; //이름을 리턴
    }
}

```

member 클래스는 이름 그래도 프로그램에 저장되는 member에 대한 값들을 저장하고 리턴해주는 클래스이다. 학생이나 교수님이 member에 들어가고 이름, 아이디, 비밀번호를 저장한다. 또한 여러 메소드를 통해 이름, 비밀번호, 아이디 등을 리턴해주어 다른 클래스 내에서 조건문에서 사용한다.

7) professor

```

package assignment5;

import java.util.ArrayList;

public class professor extends member{
    private ArrayList<lecture> lectures = new ArrayList<>(); //등록한 강의 저장위한 배열 생성

    public professor(String name, String id, String pw)
    {
        super(name, id, pw); //교수님의 이름, id, 비밀번호
    }

    public int get_num_of_lectures()
    {
        return this.lectures.size(); //등록한 강의 숫자 반환
    }

    public lecture get_index_lecture(int index)
    {
        return lectures.get(index); //강의의 index 반환
    }

    public boolean register_lecture(lecture lec) { //강의 등록 메소드

        if(schedule_check(lec)) //schedule_check메소드 실행 시 true가 반환됐다면
        {
            lectures.add(lec); //lectures에 lec 추가
            management.lectures.add(lec);
            return true; //true 반환 -> 강의등록 성공
        }
        return false; //false 반환 -> 강의등록 실패
    }

    public void register_lecture(String lecture_name, String time) { //강의 등록 메소드

        lecture new_lecture = new lecture(lecture_name, this.get_name(), time); //new
        if(schedule_check(new_lecture)) //new_lecture를 매개변수로 메소드 실행 시 true
        {
            lectures.add(new_lecture); //lectures에 추가
            management.lectures.add(new_lecture);
        }
    }

    public boolean schedule_check(lecture check) //중복되는 시간이 있는지 체크
    {
        int time1, time2;
        time1 = check.get_time1();
        time2 = check.get_time2();
        for (lecture temp : this.lectures) {

            if(time1 == temp.get_time1() || time1 == temp.get_time2() || time2 ==
temp.get_time1() || time2 == temp.get_time2())
                //temp의 time1, time2가 check의 time1, time2와 하나라도
                겹치는지 확인
        }
    }
}

```

```

        return false; //false 반환 -> 강의등록 실패
    }
    return true; //true 반환 -> 강의등록 성공
}
}

```

professor 클래스는 교수님의 정보를 저장하고 여러 메소드를 통해 조건에 맞는 값들을 반환하는 클래스이다. 이름과 아이디, 비밀번호를 저장하고, 메소드에 따라 등록한 강의의 숫자 혹은 index를 반환한다. 강의 등록을 할 때는 schedule check 메소드를 이용해 강의 시간이 겹치는지를 확인한 후 겹치면 false를 반환하고 겹치지 않으면 management에 저장한 후 true를 반환한다. 강의 이름과 시간을 매개변수로 받는 강의 등록 메소드에서도 같은 동작을 하지만 return 값이 없는 void 함수를 사용한다. schedule check 메소드에서는 앞서 time1, time2와 매개변수로 들어온 check의 time1, tim2가 겹치는지를 한 개라도 겹치는지 확인하여 true, false를 반환한다.

8) ProfessorScreen

```

package assignment5;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*.*;

public class ProfessorScreen extends JFrame implements ActionListener{

    private Pf_RegisterScreen rsc; //St_RegisterScreen 형식의 rsc 선언
    private JTabbedPane jtp;      //JTabbedPane jtp -> panel

    public ProfessorScreen() {

        jtp = new JTabbedPane(); //Panel 선언
        rsc = new Pf_RegisterScreen(); //RegisterScreen 선언
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //화면에 close 버튼 (x)

        JMenuBar jmb = new JMenuBar(); //메뉴바 생성

        JMenu menu1 = new JMenu("기능"); //메뉴1 - 기능 - 로그아웃, 종료
        JMenu menu2 = new JMenu("정보"); //메뉴2 - 정보 - 프로그램 정보

        jmb.add(menu1);
        jmb.add(menu2); //menu1,2를 메뉴바에 추가

        JMenuItem jmi1 = new JMenuItem("로그아웃");
        JMenuItem jmi2 = new JMenuItem("종료"); //메뉴1에 들어갈 아이템 선언

        menu1.add(jmi1);
        menu1.add(jmi2); //메뉴1에 아이템 2개 추가

        JMenuItem jmi3 = new JMenuItem("프로그램 정보"); //메뉴2에 들어갈 아이템 선언
        menu2.add(jmi3); //메뉴2에 아이템 추가

        jmi1.addActionListener(this);
        jmi2.addActionListener(this);
        jmi3.addActionListener(this); //목록 3개 모두 클릭이라는 이벤트에 반응해야하므로
listener 선언

        setJMenuBar(jmb); //메뉴바 set

        jtp.addTab("강의등록", rsc); //panel에 강의등록이라는 이름의 rsc 추가
        getContentPane().add(jtp);
    }
}

```

```

setTitle("Professor Panel");//이름은 Professor Panel
setSize(400,500);
setVisible(true);

}

public void actionPerformed(ActionEvent ae) //이벤트에 반응
{
    String s = ae.getActionCommand();
    if (s == "로그아웃") { //이벤트가 발생한 곳에 저장된 문자열이 로그아웃이라면
        LoginScreen login = new LoginScreen(); //새로운 로그인 창 선언
        login.setVisible(true); //새로운 로그인 창 보이게 설정
        setVisible(false); //본래 있던 창은 안보이게 설정
    }
    else if (s == "종료") { //종료라면
        new New_Window2(2); //New_Window2에 매개변수 2를 넣고 메소드
    }
    else if (s == "프로그램 정보") { //프로그램 정보라면
        Program_info info = new Program_info(); //Program_info() 선언
        info.setVisible(true); //프로그램 정보 보이는 창 보이게
    }
}

}

class Pf_RegisterScreen extends JPanel implements ActionListener{

    professor pf;
    JLabel name = new JLabel("강의 이름 : ");
    JLabel time = new JLabel("강의 시간 : ");
    JLabel info = new JLabel(" "); //강의 등록 레이블 추가
    JTextField lec_name = new JTextField(10); //강의 이름 입력하기 위한 텍스트 필드
    JTextField lec_time = new JTextField(10); //강의 시간 입력하기 위한 텍스트 필드

    Pf_RegisterScreen() {
        pf = (professor) management.now_member; //management에 저장되어있는 현재
        멤버를 professor형으로 바꾸어 pf에 저장
        // TODO Auto-generated constructor stub
        JButton register = new JButton("강의등록"); //강의등록이라는 이름의 버튼 생성

        add(name);
        add(lec_name);
        add(time);
        add(lec_time);
        add(register);
        add(info); //앞서 선언한 레이블과 버튼 추가
        register.addActionListener(this); //버튼은 클릭 시 이벤트 발생하므로 listener 선언
    }

    public void actionPerformed(ActionEvent ae) {
        String s = ae.getActionCommand();
        lecture temp; //lecture 형식으로 temp 객체 선언
        if(s == "강의등록") //이벤트가 발생한 곳에 저장된 문자열이 강의등록이라면
        {
            pf = (professor) management.now_member;
            temp = new lecture(lec_name.getText(), pf.get_name(),
            lec_time.getText()); //temp에 강의 이름과 교수님 성함, 강의 시간을 메소드 이용해 가져와 객체 생성
            if(pf.register_lecture(temp)) //temp를 매개변수로 메소드를 호출해 1이
            반환됐으면
                info.setText("정상적으로 강의가 등록되었습니다."); //강의등록 성공
            문구 알림
            else
                info.setText("시간이 겹치는 과목이 있습니다. 다시 확인해 주세요.");
            //실패 시 다시 입력하게끔 설정

        }
    }
}

```

```
}
```

```
}
```

professor screen 클래스는 professor 계정으로 로그인했을 때 나타나는 창에 관한 클래스이다. student screen 클래스와 같이 메뉴바에 기능과 정보가 있고 기능에는 로그아웃, 종료 기능이 올바르게 구현이 되어있고, 정보에는 프로그램 정보가 들어가있다. student screen과 다르게 강의를 등록할 수 있고, 강의명과 시간을 입력하는 텍스트필드가 존재한다. professor 클래스의 register_lecture 메소드를 이용해 입력한 강의명과 시간을 이미 등록된 강의와 비교해 강의등록의 성공 실패를 다루고 강의를 등록한다.

9) Program_info

```
package assignment5;

import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class Program_info extends JFrame{
    Program_info() {
        setTitle("Program Info"); //팝업창의 이름을 Program Info로 지정

        JPanel WindowContainer = new JPanel(); //panel 선언
        setContentPane(WindowContainer); //panel에 content를 넣기위한 작업

        JLabel j1 = new JLabel("<html><body style='text-align:center;'>만든이 : 안민규<br />학번 : 201620837<br />업데이트 날짜 : 2021-06-15</body></html>"); //레이블 선언

        WindowContainer.add(j1); //선언한 레이블을 panel에 추가

        setSize(300,100); //사이즈는 300,100
        setResizable(false); //사이즈 변경 불가
        setVisible(true); //창 보이게 설정
    }
}
```

Program_info 클래스는 StudentScreen, ProfessorScreen 클래스에서 프로그램 사용자가 메뉴에서 프로그램 정보를 눌렀을 때 프로그램에 대한 정보가 나오는 창을 띄어주는 클래스이다. Program Info라는 이름의 새로운 창을 만들어서 띄우게끔 설정하였다. 또한 <html><body style='text-align:center>
</html>을 이용해서 입력한 글자들이 가운데 정렬되게끔 설정하였다.

10) student

```
package assignment5;

import java.util.ArrayList;

public class student extends member{
    private ArrayList<lecture> lectures = new ArrayList<>(); //강의를 저장하기 위해 배열 선언

    public student(String name, String id, String pw)
    {
        super(name, id, pw); //학생의 이름, id, 비밀번호
    }

    public int get_num_of_lectures()
    {
        return this.lectures.size(); //수강신청한 강의 숫자 반환
    }

    public lecture get_index_lecture(int index)
    {
        return lectures.get(index); //강의의 index 반환
    }
}
```

```

public boolean register_lecture(lecture lec)
{
    if(this.duplication_check(lec) && this.schedule_check(lec)) { //강의 명과 시간을 중복으로
선택하지 않았으면
        this.lectures.add(lec); //lec을 신청한 강의목록에 추가
        return true; //true 반환 -> 수강신청 성공
    }
    else
        return false; //false 반환 -> 수강신청 실패
}

public boolean duplication_check(lecture check) //같은 과목 수강신청했는지 확인
{
    for(lecture temp : this.lectures)
    {
        if(check == temp) //check와 temp가 같으면
        {
            System.out.println("중복신청 하셨습니다. 확인하시고 다시 신청해 주세요.");
            System.out.println("겹치는 수업명 : " + temp.get_name()); //문구 출력
            return false; //false 반환
        }
    }
    return true; //아니라면 true 반환
}

public boolean schedule_check(lecture check) //시간 겹치는지 확인
{
    int time1, time2;
    time1 = check.get_time1();
    time2 = check.get_time2();
    for (lecture temp : this.lectures) {
        if(time1 == temp.get_time1() || time1 == temp.get_time2() || time2 == temp.get_time1()
|| time2 == temp.get_time2())
        { //temp의 time1, time2가 check의 time1, time2와 하나라도 겹치는지 확인
            System.out.println("시간이 겹치는 수업이 존재합니다. 확인해보시고 다시 신청해
주세요.");
            System.out.println("겹치는 수업명 : " + temp.get_name());
            System.out.println(); //문구 출력
            return false; //false 반환 -> 겹치므로 수강신청 실패
        }
    }
    return true; //true 반환 -> 수강신청 성공
}
}

```

student 클래스는 professor 클래스와 비슷하게 학생의 정보를 저장하고 여러 메소드를 통해 적당한 값들을 반환한다. 학생의 경우에는 수강신청하려는 강의가 이미 신청되어있는 강의명 혹은 시간에 중복되지 않는지를 판단하고 이에 따라 수강신청 성공 실패를 반환한다. register lecture 메소드는 성공 시 lectures 배열에 저장하고 실패시 false를 반환한다. register lecture 안에서 사용되는 duplication check 메소드는 강의명이 중복되는지를 판단하고 중복되면 중복신청했다는 문구와 함께 false를 반환하고 올바르게 선택됐다면 true를 반환하고 수강신청이 성공한다. 또한 register lecture 내에서 사용되는 schedule check 메소드는 신청하려는 강의의 시간 중 어느 하나라도 이미 신청한 강의의 시간과 겹치면 실패라고 판단하고 문구와 함께 false를 반환한다. 이러한 두 메소드에서 모두 true가 나와야 register lecture 메소드에서 true를 반환한다.

11) StudentScreen

```

package assignment5;

import java.awt.*;

```

```

import java.awt.event.*;
import javax.swing.*;

public class StudentScreen extends JFrame implements ActionListener {
    private St_RegisterScreen rsc; //St_RegisterScreen 형식의 rsc 선언
    private JTabbedPane jtp; //JTabbedPane jtp -> panel

    public StudentScreen() {

        jtp = new JTabbedPane(); //Panel 선언
        rsc = new St_RegisterScreen(); //RegisterScreen 선언
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //화면에 close 버튼 (x)
        JMenuBar jmb = new JMenuBar(); //메뉴바 생성

        JMenu menu1 = new JMenu("기능"); //메뉴1 - 기능 - 로그아웃, 종료
        JMenu menu2 = new JMenu("정보"); //메뉴2 - 정보 - 프로그램 정보

        jmb.add(menu1);
        jmb.add(menu2); //menu1,2를 메뉴바에 추가

        JMenuItem jmi1 = new JMenuItem("로그아웃");
        JMenuItem jmi2 = new JMenuItem("종료"); //메뉴1에 들어갈 아이템 선언

        menu1.add(jmi1);
        menu1.add(jmi2); //메뉴1에 아이템 2개 추가

        JMenuItem jmi3 = new JMenuItem("프로그램 정보"); //메뉴2에 들어갈 아이템 선언
        menu2.add(jmi3); //메뉴2에 아이템 추가

        jmi1.addActionListener(this);
        jmi2.addActionListener(this);
        jmi3.addActionListener(this); //목록 3개 모두 클릭이라는 이벤트에 반응해야하므로

        setJMenuBar(jmb); //메뉴바 set

        jtp.addTab("수강신청", rsc); //panel에 수강신청이라는 이름의 rsc 추가
        getContentPane().add(jtp);

        setTitle("Student Panel"); //이름은 Student Panel
        setSize(400,500);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent ae) //이벤트에 반응
    {
        String s = ae.getActionCommand();
        if (s == "로그아웃") { //이벤트가 발생한 곳에 저장된 문자열이 로그아웃이라면
            LoginScreen login = new LoginScreen(); //새로운 로그인 창 선언
            login.setVisible(true); //새로운 로그인 창 보이게 설정
            setVisible(false); //본래 있던 창은 안보이게 설정
        }
        else if (s == "종료") { //종료라면
            new New_Window2(2); //New_Window2에 매개변수 2를 넣고 메소드
        }
        else if (s == "프로그램 정보") { //프로그램 정보라면
            Program_info info = new Program_info(); //Program_info() 선언
            info.setVisible(true); //프로그램 정보 보이는 창 보이게
        }
    }
}

class St_RegisterScreen extends JPanel implements ActionListener { //rsc

```

listener 선언

실행

설정

```

int num_of_lectures = 6;
ButtonGroup gb = new ButtonGroup(); //버튼 그룹 생성
JCheckBox[] jc = new JCheckBox[num_of_lectures + 1]; //체크박스 생성
private student st; //student 형식 객체 생성

St_RegisterScreen() {
    st = (student) management.now_member; //management에 저장되어있는 현재
    //멤버를 student형으로 바꾸어 st에 저장
    num_of_lectures = management.lectures.size(); //강의의 사이즈를 num of lectures에
    //저장
    GridLayout b1 = new GridLayout(num_of_lectures + 2,4,10,10); //행 : 강의 목록 +2,
    //열 : 4, 줄간격 : 10,10 으로 하는 grid layout 설정
    setLayout(b1); //앞서 설정한 레이아웃으로 rsc 설정

    String[] content = {"선택","과목명","시간","교수명"}; //4열에 들어갈 목록 이름

    JButton btn = new JButton("수강신청"); //수강신청 적혀있는 버튼 생성

    for(int i = 0; i < 4; i++)
        add(new JLabel(content[i])); //목록 이름을 차례대로 레이블 선언하여 패널에
    //붙임 첫번째 행

    for(int i = 0; i < num_of_lectures ; i++)
    {
        jc[i] = new JCheckBox(""); //i에 체크박스 생성 -> 순서대로 체크박스
        // (선택칸)
        add(jc[i]); //생성한 체크박스 붙임
        add(new JLabel(management.lectures.get(i).get_name())); //강의 이름(과목명)
        add(new JLabel(management.lectures.get(i).get_time())); //시간(시간)
        add(new JLabel(management.lectures.get(i).get_prof())); //교수님
        // 성함(교수명)

        gb.add(jc[i]); //앞서 선언한 체크박스를 버튼 그룹으로 설정
        jc[i].addActionListener(this); //체크박스 클릭하면서 이벤트가 발생하므로
        // listener 선언

    }
    add(new JLabel(""));
    add(new JLabel(""));
    add(new JLabel("")); //num of lectures 행까지 다 채웠으므로 마지막 남은 한 행을
    //위해 공백 3개
    add(btn); //버튼 추가 -> 이로 인해 앞서 grid layout의 행을 num of
    //lectures+2로 설정

    btn.addActionListener(this); //버튼 또한 눌리면서 이벤트가 발생하므로 listener 선언

    // TODO Auto-generated constructor stub
}

public void actionPerformed(ActionEvent ae) {
    String s = ae.getActionCommand();
    lecture temp;
    if(s == "수강신청") //이벤트가 발생한 곳에 저장된 문자열이 수강신청이라면
    {
        st = (student) management.now_member; //st에 now_member 저장
        for(int i = 0; i < num_of_lectures; i++) //강의 수만큼 반복
        {
            if(jc[i].isSelected()) //선택된 체크박스 확인
            {
                temp = management.lectures.get(i); //선택된 체크박스
                // 번호 i에 맞는 과목 temp에 저장

                if(st.register_lecture(temp)) //temp를
                //student.register_lecture 메소드의 매개변수로 사용해 호출해서 1이라면 -> 수강 신청함
                {
                    new New_Window2(0); //new_window2
                    //메소드 매개변수0으로 호출

                }
                else
                //수강신청이 되지

```

않았다면

```
        {
            new New_Window2(1); //new_window2
        }
    }
}

class New_Window2 extends JFrame implements ActionListener{ //new_window2 메소드
    New_Window2(int flag) {
        setTitle("Alert"); //생성된 팝업 창의 이름은 Alert - 알림

        JPanel NewWindowContainer = new JPanel(); //Panel 생성
        setContentPane(NewWindowContainer);

        FlowLayout fl = new FlowLayout(FlowLayout.CENTER, 5, 5); //flowlayout으로 설정

        JButton btn1 = new JButton("종료"); //종료 버튼
        JButton btn2 = new JButton("취소"); //취소 버튼 -> 실수로 종료를 눌렀을 때 바로
        종료되지 않게끔 추가구현

        if(flag == 0) //new_window2 메소드를 호출하는 매개변수가 0이라면
        {
            NewWindowContainer.add(new JLabel("선택하신 강의가 정상적으로
            신청되었습니다."); //신청 성공 메세지

            setSize(300,100);
            setResizable(false);
            setVisible(true);
        }
        else if(flag == 1) //매개변수가 1이라면
        {
            NewWindowContainer.add(new JLabel("선택하신 강의를 신청할 수
            없습니다.");
            NewWindowContainer.add(new JLabel("시간 및 강의를 다시 확인해
            주세요."); //신청 실패 메세지

            setSize(300,100);
            setResizable(false);
            setVisible(true);
        }
        else if(flag == 2) //매개변수가 2라면
        {
            NewWindowContainer.setLayout(fl); //flowlayout 적용

            NewWindowContainer.add(new JLabel("프로그램을 종료하시겠습니까?"));
            //한번 더 물어보는 문구와 함께
            NewWindowContainer.add(btn1);
            NewWindowContainer.add(btn2); //앞서 생성한 버튼 추가

            setSize(200,100);
            setResizable(false);
            setVisible(true);
        }







        btn1.addActionListener(this);
        btn2.addActionListener(this); //두 버튼 모두 이벤트 listener로 선언
    }

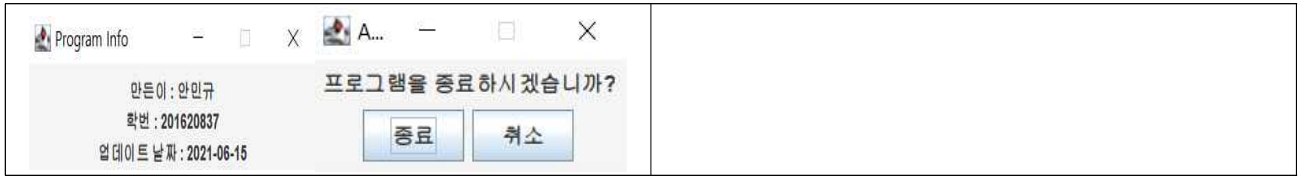
    public void actionPerformed(ActionEvent ae) {
        String s = ae.getActionCommand();
        if(s == "종료") //눌리게 종료라면
            System.exit(0); //시스템 종료
        else if(s == "취소") //눌리게 취소라면
```


<pre> } setVisible(false); //Alert 창 지움 } } </pre>	
<p>studentscreen 클래스에서는 student 신분의 계정으로 로그인했을 때 나타나는 창에 관한 클래스이다. 기능, 정보의 메뉴바를 생성하고 기능에는 로그아웃과 종료를, 정보에는 프로그램 정보를 추가하였다. 또한 메뉴바에 존재하는 3개의 아이템을 눌렀을 때 발생하는 이벤트에 응답하기 위해서 actionlistener를 선언하였다. 눌린 아이템의 문자열에 따라 로그아웃이라면 loginscreen을 새로 띄우고 원래 있던 창의 visible을 false로 바꾼다. 종료가 선택됐다면 new window2 메소드를 매개변수 2로 호출하고 프로그램 정보가 눌렀다면 program info 클래스 창이 보이게 설정하였다. 화면에 나타나는 registerscreen 클래스는 이미 등록되어 있는 강의들을 보고 선택할 수 있게 체크박스과 과목명 시간 교수명으로 나타나게 설정을 하였고, 체크박스과 수강신청 버튼은 actionlistener로 선언하였다. student 클래스의 register lecture 메소드를 호출하여 true를 반환했다면 new_window2를 매개변수 0으로 호출하고 수강신청에 실패했다면 new window2를 매개변수 1로 호출한다. new window2를 매개변수 0으로 호출 시 수강신청에 성공한 것이고 1로 호출 시 실패했다는 것이므로 다시 선택하게끔 문구를 출력한다. 매개변수가 2로 호출됐다면 프로그램 종료가 눌린 것으로 실수로 눌렀을 가능성을 생각해 정말 종료할건지 물어본 후 종료, 취소 버튼 중 하나를 누르게끔 설정하였다. 두 버튼 모두 actionlistener 선언하였고, 종료가 눌리면 시스템이 종료되고 취소가 눌리면 alert 창이 지워진다.</p>	

2. 결과화면 분석

1) Student

	
	
	1. 로그인 페이지에서 학생으로 로그인 시도
	2. 수강신청할 수 있는 목록이 나옴
	3. 이미연 교수님의 월B목B 전프 수업 클릭 후 수강신청 버튼 클릭
	4. 정상적으로 신청되었다는 alert 팝업 등장
	5. 앞서 선택한 전프와 월B 시간이 겹치는 C++ 수업을 선택하고 수강신청 버튼 클릭
	6. 신청할 수 없다는 alert 팝업 등장
	7. 기능 메뉴에는 로그아웃과 종료가 있음
	8. 정보 메뉴에서는 프로그램의 정보를 확인할 수 있음
	9. 프로그램 정보 클릭 시 Program Info 팝업 등장
	10. 종료 클릭 시 종료하시겠습니까?라는 문구와 함께 버튼 등장
	11. 종료 버튼 누를 시 프로그램이 종료되고 취소 누르면 팝업이 없어지고 프로그램 그대로 사용 가능
	12. 로그아웃은 클릭 시 다시 로그인 화면으로 돌아감



2) Professor

- 로그인 페이지에서 교수님으로 로그인 시도
- 강의등록할 수 있는 창 등장
- 자알 수업을 월B목B에 하려고 했으나 이미 그 시간대에 수업이 존재하므로 강의 등록 불가
- 월C목C로 시간 변경 시 강의가 정상적으로 등록 됨
- 학생과 마찬가지로 기능과 정보 모두 사용 가능
- 강의를 등록하고 나서 로그아웃 후 학생으로 다시 들어가서 확인 시 제대로 등록이 되어있는 것이 확인된다.

3) 회원가입

- 로그인 대신에 회원가입 누를 시 회원가입 창
- 학생과 신분을 고를 수 있고 이름과 ID, PW를 입력한다.
- 이름은 안민규, 아이디는 min11, pw는 1234로 회원가입 시도하면 회원가입이 완료되었다고 나온다.
- 그 후 새로 회원가입한 아이디로 로그인을 시도하면 로그인에 성공한다. 이를 통해 회원가입이 진행이 되면 올바르게 management에 저장되는 것을 확인할 수 있다.
- 이미 management에 들어가있는 mylee라는 아

