

# 객체지향프로그래밍II 결과 보고서

제목 : 1조 프로젝트 결과

제출일	2023년 11월 25일
9조	조원 : 싯홍재(20203131), 장준혁(20203230), 하태형(20203179), 이민우(20221380), 변정빈(20223086)

© 2018 동의대학교 컴퓨터소프트웨어공학과

이 결과 보고서 양식은 ISO/IEC/IEEE 29148:2011 요구공학 국제표준과 스크럼 애자일 프로세스를 적용하여 진행된 객체지향설계 교과목의 설계 프로젝트에 맞게 테일러링한 것입니다.

<제목 차례>

1. 프로젝트 개요 .....	3
2. 시스템 .....	3
2.1 소프트웨어 구조 .....	3
2.2 시스템 인터페이스 .....	3
2.3 사용자 인터페이스 .....	3
3. 시스템 설명 .....	4
3.1 초기에 지정 된 ID, PW로 로그인할 수 있으며, 아이디에 따라 교수, 학생, 학사를 구분할 수 있다. ....	4
3.1.1 분석 .....	4
3.1.2 설계 .....	4
3.1.3 구현 .....	4
3.1.4 테스트 .....	5
3.2 학생은 수강신청을 할 수 있고, 수강내역 조회 및 수강 취소를 할 수 있으며, 수강료 조회가 가능하다. ....	5
3.2.1 분석 .....	5
3.2.2 구현 .....	5
3.2.3 테스트 .....	5
3.3 신청 가능한 최대 인원을 초과 하거나, 신청한 학점이 18학점을 넘으면 강의를 더 신청할 수 없다. ....	5
3.3.1 분석 .....	5
3.3.2 구현 .....	5
3.3.3 테스트 .....	5
3.4 수업담당자는 강좌번호, 강좌이름, 담당학과, 학점, 강좌 설명을 지정하여 새 강좌를 등록할 수 있다. ....	5
3.4.1 분석 .....	5
3.4.2 구현 .....	5
3.4.3 테스트 .....	5
3.5 수업담당자는 등록된 강좌를 조회하고, 등록된 강좌 내에서 담당교수, 수강 가능 최소/최대 인원을 지정하여 학기별 강의를 개설할 수 있다. ....	5
3.5.1 분석 .....	5
3.5.2 구현 .....	5

3.6 수업담당자는 기존의 발급 시스템을 이용해 수강료 청구서를 발급할 수 있다.	5
3.6.1 분석	5
3.6.2 구현	5
3.7 교수는 담당한 강좌에 대한 학생의 성적 입력이 가능하다.	5
3.7.1 분석	5
3.7.2 구현	5
3.8 교수는 자신이 담당한 강의의 출석부 조회 결과로 검색을 통해 수강생의 학번, 이름, 취득 학점을 확인할 수 있다.	5
3.8.1 분석	5
3.8.2 구현	5
3.9 학사 담당자는 새 학생 및 교수를 등록 및 수정, 삭제할 수 있다.	5
3.9.1 분석	5
3.9.2 구현	5
3.10 학사 담당자는 학생과 교수 정보를 학번/교수 번호와 이름으로 검색 및 조회할 수 있다.	5
3.10.1 분석	5
3.10.2 구현	5
3.11 7자리 영문자 및 숫자로 구성된 비밀번호로 변경할 수 있다	5
3.11.1 분석	5
3.11.2 설계	5
3.11.3 구현	5
3.11.4 테스트	5
4. 프로젝트 평가	6
4.1 프로젝트 완성도	6
4.2 일정 계획 평가	6
4.3 역할 수행 평가	6
4.4 소스 코드 버전 제어 도구 사용	7
4.5 설계 구성요소	7
4.6 현실적 제한조건	7
5. 소감	8

# 1. 프로젝트 개요

## 1.1 비전

이번 프로젝트는 대학 구성원인 교수, 학생, 직원의 작업을 효율적으로 관리 및 지원하기 위한 **대학정보시스템(UIS)**이라는 전산 시스템을 구축함에 따라 효율적인 대학 학사정보시스템을 설계함에 그 목적이 있다.

**대학정보시스템(UIS)** 구축에 있어 **Swing**을 이용한 **GUI** 기반의 사용자 인터페이스를 제공함으로써 SW 사용에 편의를 제공하며, 특히 이번 프로젝트에서는 SW 개발 시 **V Model**에 기술되어 있는 개발과정 및 테스트 절차를 최대한 준수하여, SW 품질 향상 및 사후 유지·보수 관리 시 문제 발생 지점에 대한 신속한 대응방안을 마련하며 소프트웨어의 안정성까지 보장한다.

## 1.2 문제 기술

학사관리기능, 수업관리기능, 사용자 관리 기능 등을 제공하는 프로그램을 설계하여, 일일이 하나하나 수기로 자료를 받고, 기록하고, 보관하는 번거로운 과정 없이 소프트웨어를 사용하여 교수, 학생, 직원 모두의 입장에서 작업 수행시간을 단축 시켜줄 수 있는 효율적인 전산 시스템을 구축하려고 한다.

## 1.3 요구사항 목록

ID	기능요구사항	개발자
SFR-100	사용자 관리	하,변,장,신,이
SFR-101	사용자 추가 또는 삭제를 선택할 수 있다.	변
SFR-102	사용자 추가를 위한 정보를 입력한다.	변
SFR-103	java.io를 활용하여 txt 파일을 불러온다	하,변,장,신,이
SFR-104	사용자 추가 성공 또는 실패를 알린다.	변
SFR-105	사용자 제거를 위해 현재 사용자 목록을 보여준다.	변
SFR-106	제거할 사용자를 선택한다.	변
SFR-107	사용자 제거 성공 또는 실패를 알린다.	변
SFR-200	학사 담당자 요구사항	하,변,장,신,이
SFR-201	학사 담당자를 추가한다	신
SFR-202	학사 담당자-추가된 학생/교수의 정보를 검색할 수 있다.	신
SFR-203	학사 담당자-추가된 학생/교수의 정보를 수정할 수 있다.	신

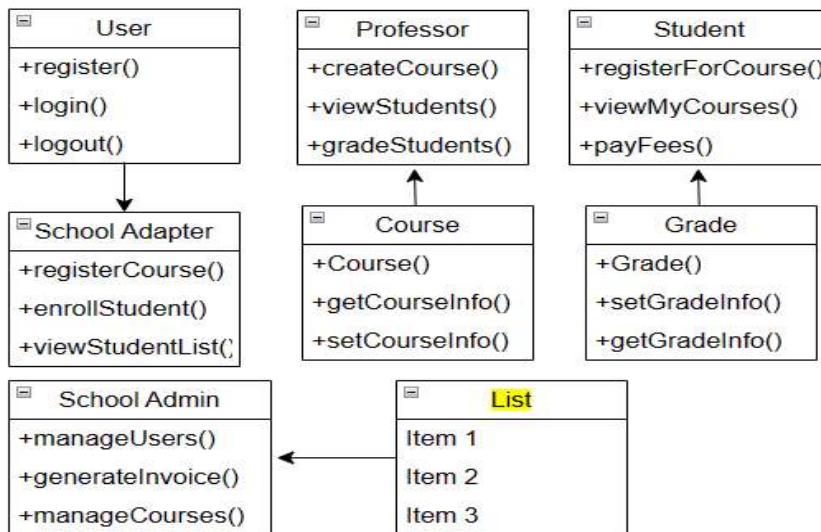
SFR-204	학사 담당자-추가된 학생/교수의 정보를 삭제할 수 있다.	하,변,장,신,이
SFR-300	수업 담당자 요구사항	이
SFR-301	수업 담당자를 추가한다	이
SFR-302	수업 담당자-새로운 강좌 정보를 추가할 수 있다.(강좌번호,이름,학과,학점,설명,학생 수)	이
SFR-303	수업 담당자-등록된 강좌에 대해서만 매 학기에 강의를 개설할 수 있다	이
SFR-304	수업 담당자-개설된 강의에 담당 교수 및 수강 가능 최소/최대 학생 수가 지정된다.	이
SFR-305	수업 담당자-강의가 개설되지 않은 강좌는 강좌 번호를 제외한 다른 정보에 대한 변경 및 강좌 자체에 대한 삭제가 가능하다.	이
SFR-306	수업 담당자-수강료 청구서를 발급할 수 있다.	하,변,장,신,이
SFR-400	학생 교수 요구사항	장
SFR-401	학생, 교수의 정보를 추가한다.	장
SFR-402	학생-수강하고 싶은 과목을 신청할 수 있다.	장
SFR-403	수강인원이 찬 강의의 수강신청과 중복 수강을 거부할 수 있다.	장
SFR-404	교수-자신의 강좌 학생 명단을 확인할 수 있다.	장
SFR-405	교수-자신의 강좌 학생의 성적을 입력할 수 있다.	장
SFR-406	학생-자신의 성적 정보를 확인할 수 있다	장
SFR-407	교수-자신의 담당 강의 출석부를 조회할 수 있다.(학생의 학번/이름/취득 학점)	장
SFR-500	로그인 및 화면 출력	하,변,장,신,이
SFR-501	교,학,직의 번호를 아이디로 하고 주민번호 뒷자리를 임시 비밀번호로 지정할 수 있다.	하
SFR-502	사용자 인증을 한다.	하
SFR-503	사용자에게 맞는 화면을 보여준다(학생/교수/직원)	하
SFR-504	모든 사용자-로그인 후 초기 암호를 수정할 수 있다.	하
SFR-505	로그인 실패 시 로그인 실패 화면을 보여준다.	하

## 2. 시스템

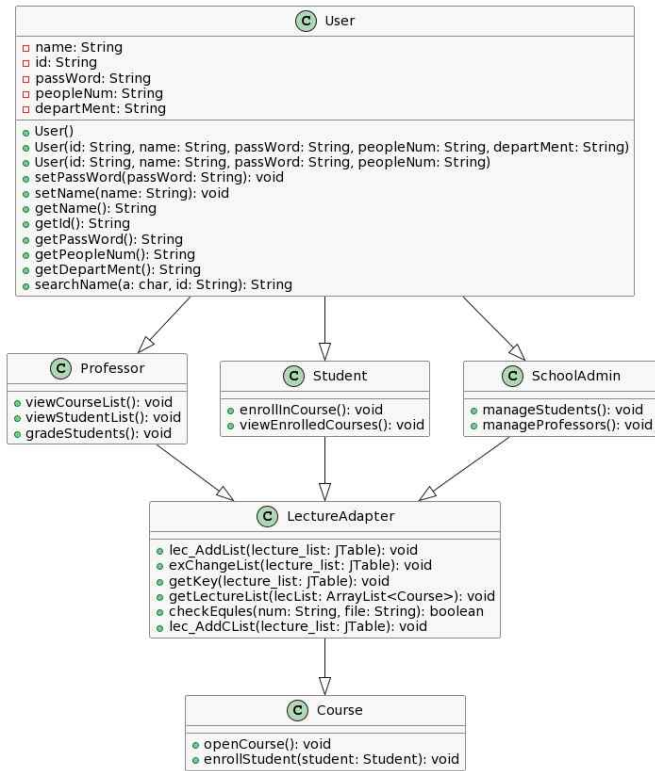
### 2.1 소프트웨어 구조



## 2.2 시스템 인터페이스



## 2.3 사용자 인터페이스



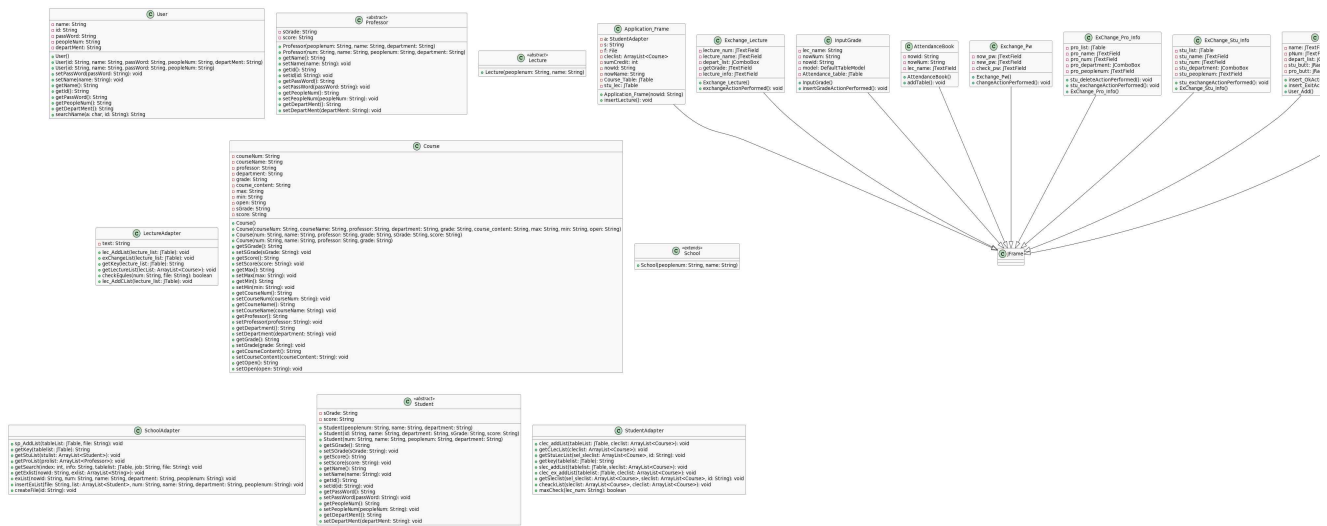
### 3. 시스템 설명

3.1 초기에 지정 된 ID, PW로 로그인을 할 수 있으며, 아이디에 따라 교수, 학생, 학사를 구분할 수 있다.

### 3.1.1 분석 및 설계

로그인을 할 수 있으며, 로그인 시 교수, 학생, 학사관리자, 수업관리자는 아이디 고유 문자를 가진다.

### 3.1.2 설계





### 3.1.3 구현

#### 관련 코드

1. 초기에 지정 된 ID,PW로 로그인 할 수 있으며, try-catch문을 사용하여 로그인에 실패하였을 때(파일이 발견되지 않았을 때) 예외처리를 한다.
2. 로그인 정보를 확인하여, 아이디 고유 문자가 P이면 교수, S이면 학생, H이면 학사 담당자, G이면 수업담당자로 인식 하며, 로그인 실패시 실패 문구가 뜬다.

```
private void Login_ButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    String id_Field = ID_Field.getText(); //ID 필드에서 값 받아오기  
    String pw_Field = new String(PW_Field.getPassword()); //PW 필드에서 값 받아오기  
    char a;  
    BufferedReader reader = null;  
    String str;  
    String[] key;  
  
    try {  
        if (ID_Field.getText().isEmpty() || PW_Field.getText().isEmpty()) { //입력받지 않았을 때  
            showMessageDialog(null, "아이디 또는 비밀번호를 입력해주세요.");  
        } else {  
            a = id_Field.charAt(0); //아이디를 입력하는 직군을 구분하기 위한 이니셜 저장  
            if (a == 'P') { //아이디 고유 문자가 p-> 교수  
                reader = new BufferedReader(new InputStreamReader(new FileInputStream("professor.txt"), text)); //읽을 파일 열기  
                while ((str = reader.readLine()) != null) { //마지막 문장이 아닐동안 반복  
                    key = str.split("/"); // "/"를 이용해 배열에 저장  
                    //table에 따라 [0] : id, [1] : name, [2] : pw, [3] : peopleNum, [4] : departMent  
                    if (key[0].equals(id_Field)) { //입력받은 아이디가 문장안에 있을시  
                        if (key[2].equals(pw_Field)) { //비밀번호가 같은지 검사  
                            Professor_Main_Frame pro = new Professor_Main_Frame(key[0]);  
                            //맞으면 해당 메뉴 출력  
                            pro.setVisible(true);  
                            dispose(); //현재창은 닫기  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```

} else if (a == 'S') { //아이디 고유 문자가 s -> 학생
    reader = new BufferedReader(new InputStreamReader(new FileInputStream("student.txt"), text));
    while ((str = reader.readLine()) != null) {
        if (str.contains(id_Field)) {
            key = str.split("/");
            if (key[2].equals(pw_Field)) {
                Student_Main_Frame stu = new Student_Main_Frame(key[0], a); //임시로 로그인 시 사용자 정보를 넘김
                stu.setVisible(true);
                //nowName = key[1];
                //new Student_Main_Frame(nowName).setVisible(true);
                dispose();
            }
        }
    }
}

} else if (a == 'H') { //아이디 고유 문자가 h -> 학사
    reader = new BufferedReader(new InputStreamReader(new FileInputStream("school.txt"), text));
    while ((str = reader.readLine()) != null) {
        if (str.contains(id_Field)) {
            key = str.split("/");
            if (key[2].equals(pw_Field)) {
                School_Main_Frame sch = new School_Main_Frame(key[0], a);
                sch.setVisible(true);
                dispose();
            }
        }
    }
}
}

```

```

    } else if (a == 'G') { //아이디 고유 문자가 g -> 수업
        reader = new BufferedReader(new InputStreamReader(new FileInputStream("lecture.txt"), "text"));
        while ((str = reader.readLine()) != null) {
            if (str.contains(id_Field)) {
                key = str.split("/");
                if (key[2].equals(pw_Field)) {
                    Lecture_Main_Frame lec = new Lecture_Main_Frame(key[0], a);
                    lec.setVisible(true);
                    dispose();
                }
            }
        }
    }
    } else {
        JOptionPane.showMessageDialog(null, "아이디 또는 비밀번호가 잘못 입력 되었습니다.\n" +
            "아이디와 비밀번호를 정확히 입력해 주세요..", "ERROR_MESSAGE", JOptionPane.ERROR_MESSAGE);
    }
}

} catch (FileNotFoundException ex) { //파일이 발견되지 않았을 때 예외처리
    Logger.getLogger(Login_Frame.class.getName()).log(Level.SEVERE, null, ex);
} catch (UnsupportedEncodingException ex) {
    Logger.getLogger(Login_Frame.class.getName()).log(Level.SEVERE, null, ex);
} catch (IOException ex) {
    Logger.getLogger(Login_Frame.class.getName()).log(Level.SEVERE, null, ex);
}
}

```

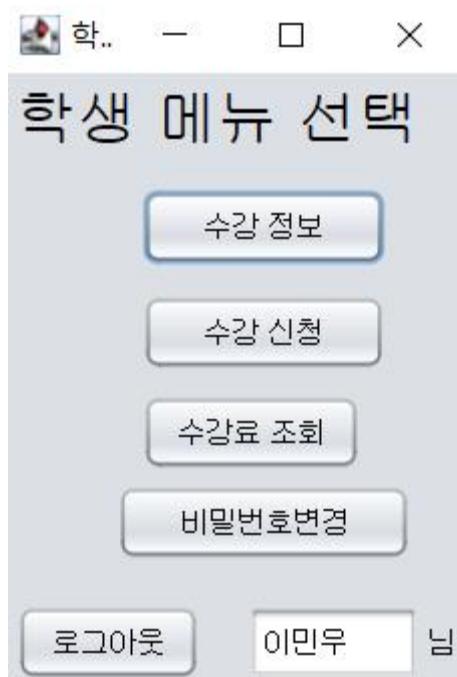
구현 결과

1. 학생

▶ 로그인 전



▶ 로그인 후



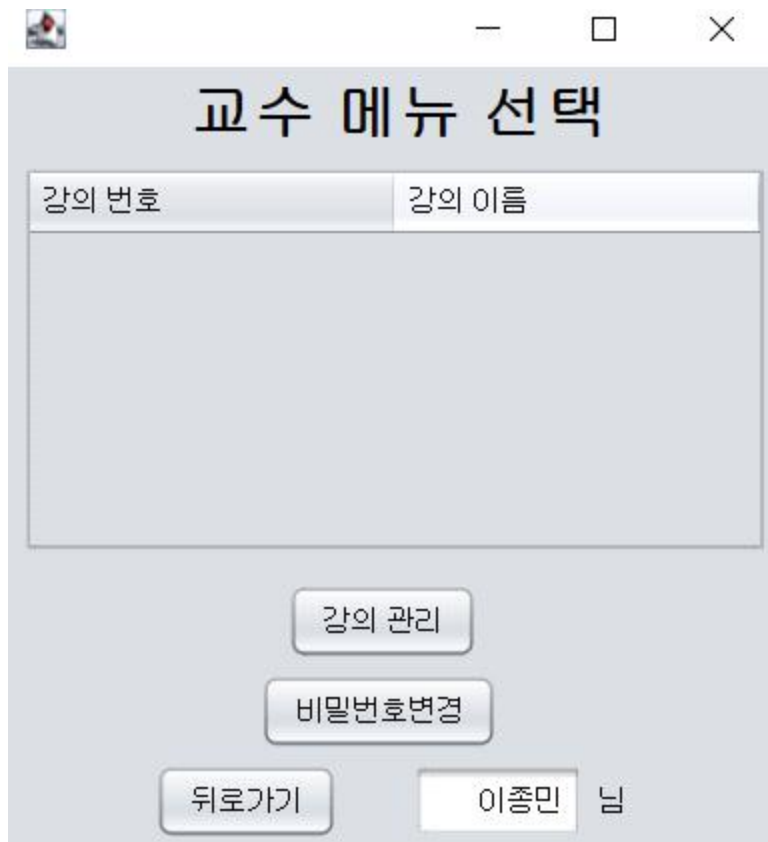
구현 결과

2. 교수

▶ 로그인 전



▶ 로그인 후



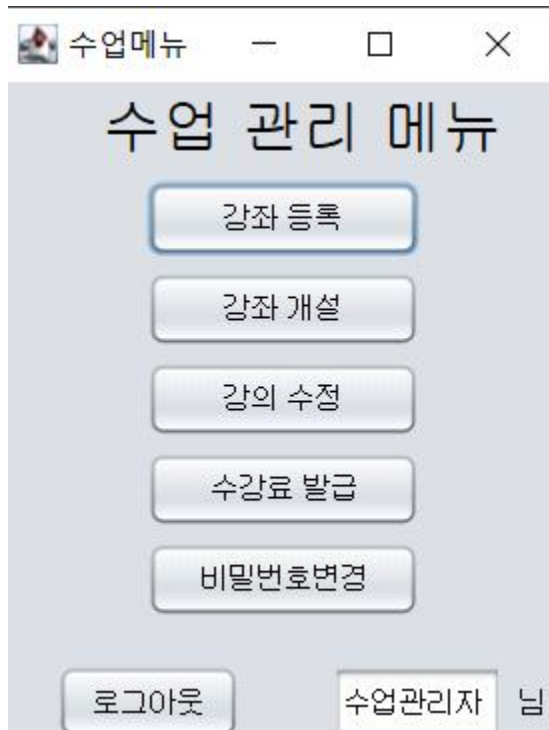
구현 결과

### 3. 수업 관리자

▶ 로그인 전



▶ 로그인 후





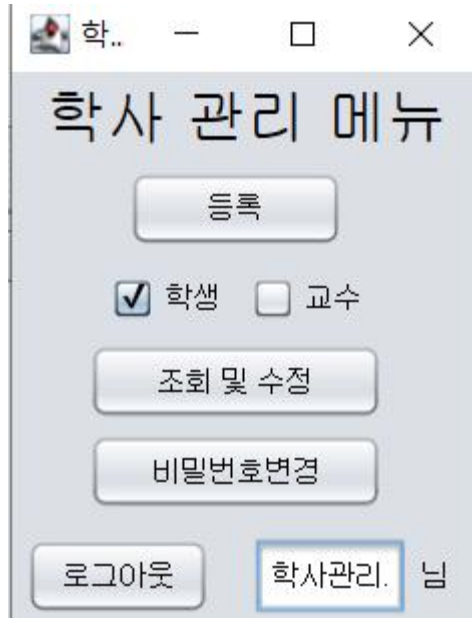
구현 결과

#### 4. 학사 관리자

▶ 로그인 전



▶ 로그인 후



### 3.1.4 테스트

#### 테스트 코드- 1

학생, 교수, 학사 담당자, 수업 담당자 객체를 각각 만들어 주고 각 객체마다 알파벳 1자리와 숫자 3자리를 포함한 아이디를 각각 만들어주고 학생은 S, 교수는 P, 학사 담당자는 H, 수업 담당자는 G로 시작하는 아이디를 만들어주는 테스트 코드

```
package cse.tests;
import java.util.Random;
class Person {
    private final String id;
    public Person(String id) {
        this.id = id;
    }

    public String getId() {
        return id;
    }
}

class Student extends Person {
    public Student() {
        super(Tests.generateId("S"));
    }
}

class Professor extends Person {
    public Professor() {
        super(Tests.generateId("P"));
    }
}

class AcademicAdministrator extends Person {
    public AcademicAdministrator() {
        super(Tests.generateId("H"));
    }
}

class CourseInstructor extends Person {
    public CourseInstructor() {
        super(Tests.generateId("G"));
    }
}

public class Tests {
    public static void main(String[] args) {
        // 학생 객체 생성 및 아이디 출력
        Student student = new Student();
        System.out.println("Student ID: " + student.getId());
    }
}
```



```

// 교수 객체 생성 및 아이디 출력
Professor professor = new Professor();
System.out.println("Professor ID: " + professor.getId());

// 학사 담당자 객체 생성 및 아이디 출력
AcademicAdministrator academicAdministrator = new AcademicAdministrator();
System.out.println("Academic Administrator ID: " + academicAdministrator.getId());

// 수업 담당자 객체 생성 및 아이디 출력
CourseInstructor courseInstructor = new CourseInstructor();
System.out.println("Course Instructor ID: " + courseInstructor.getId());
}

static String generateId(String prefix) {
    Random random = new Random();
    int randomNum = random.nextInt(900) + 100; // 100부터 999까지의 난수 생성
    return prefix + randomNum;
}
}

```

## ▶ 테스트 코드 결과

```

-----< cse:tests >-----
[ ] Building tests 1.0-SNAPSHOT
    from pom.xml
    -----[ jar ]-----
[ ] --- resources:3.3.1:resources (default-resources) @ tests ---
    skip non existing resourceDirectory C:\Users\walsdn\Documents\NetBeansProjects\tests\src\main\resources
[ ] --- compiler:3.11.0:compile (default-compile) @ tests ---
    Changes detected - recompiling the module! :source
    Compiling 1 source file with javac [debug target 17] to target\classes
[ ] --- exec:3.1.0:exec (default-cli) @ tests ---
    Student ID: S707
    Professor ID: P559
    Academic Administrator ID: H985
    Course Instructor ID: G563
    -----
    BUILD SUCCESS

```

## 테스트 코드- 2

학생, 교수, 학사 담당자, 수업담당자의 id로 로그인을 하였을 때 각각의 객체마다 다른 화면을 보

여주는 테스트 코드  
package cse.tests;

import java.util.Scanner;

// 공통 인터페이스

```
interface User {  
    String getId();  
    void displayScreen();  
}
```

// 학생 클래스

```
class Student implements User {  
    private final String id;  
  
    public Student(String id) {  
        this.id = id;  
    }  
  
    @Override  
    public String getId() {  
        return id;  
    }  
  
    @Override  
    public void displayScreen() {  
        System.out.println("Welcome, Student " + id + "! Student Dashboard is displayed.");  
    }  
}
```

// 교수 클래스

```
class Professor implements User {  
    private final String id;  
  
    public Professor(String id) {  
        this.id = id;  
    }  
  
    @Override  
    public String getId() {  
        return id;  
    }  
  
    @Override  
    public void displayScreen() {  
        System.out.println("Welcome, Professor " + id + "! Professor Dashboard is displayed.");  
    }  
}
```

// 학사 담당자 클래스

```
class AcademicAdministrator implements User {  
    private final String id;  
  
    public AcademicAdministrator(String id) {
```

```

        this.id = id;
    }

    @Override
    public String getId() {
        return id;
    }

    @Override
    public void displayScreen() {
        System.out.println("Welcome, Academic Administrator " + id + "! Academic
Dashboard is displayed.");
    }
}

// 수업 담당자 클래스
class CourseInstructor implements User {
    private final String id;

    public CourseInstructor(String id) {
        this.id = id;
    }

    @Override
    public String getId() {
        return id;
    }

    @Override
    public void displayScreen() {
        System.out.println("Welcome, Course Instructor " + id + "! Instructor Dashboard is
displayed.");
    }
}

public class UserLoginTest {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // 사용자 입력을 받아 로그인
        System.out.print("Enter your ID: ");
        String userId = scanner.nextLine();

        // 사용자의 역할에 따라 다른 객체 생성
        User user;
        if (userId.startsWith("S")) {
            user = new Student(userId);
        } else if (userId.startsWith("P")) {
            user = new Professor(userId);
        } else if (userId.startsWith("H")) {
            user = new AcademicAdministrator(userId);
        } else if (userId.startsWith("G")) {
            user = new CourseInstructor(userId);
        } else {

```

```

        System.out.println("Invalid ID. Exiting...");
        return;
    }

    // 로그인한 사용자의 화면을 출력
    user.displayScreen();
}
}

```

## ▶ 테스트 코드 결과

— 학생의 ID를 넣었을 경우

```

-----< cse:tests >-----
Building tests 1.0-SNAPSHOT
  from pom.xml
-----[ jar ]-----

--- resources:3.3.1:resources (default-resources) @ tests ---
skip non existing resourceDirectory C:\Users\walsdn\Documents\NetBeansProjects\tests\src\main\resources

--- compiler:3.11.0:compile (default-compile) @ tests ---
Changes detected - recompiling the module! :source
Compiling 1 source file with javac [debug target 17] to target\classes

--- exec:3.1.0:exec (default-cli) @ tests ---
Enter your ID: s001
Invalid ID. Exiting...

-----
BUILD SUCCESS
-----

Total time: 15.056 s
Finished at: 2023-11-24T19:10:21+09:00
-----

```

- 교수의 ID를 넣었을 경우

```
-----< cse:tests >-----  
Building tests 1.0-SNAPSHOT  
  from pom.xml  
-----[ jar ]-----  
  
--- resources:3.3.1:resources (default-resources) @ tests ---  
skip non existing resourceDirectory C:\Users\alsdn\Documents\NetBeansProjects\tests\src\main\resource  
  
--- compiler:3.11.0:compile (default-compile) @ tests ---  
Changes detected - recompiling the module! :source  
Compiling 1 source file with javac [debug target 17] to target\classes  
  
--- exec:3.1.0:exec (default-cli) @ tests ---  
Enter your ID: P001  
Welcome, Professor P001! Professor Dashboard is displayed.  
-----  
BUILD SUCCESS  
-----  
Total time: 17.073 s  
Finished at: 2023-11-24T19:21:43+09:00  
-----
```

- 학사 담당자를 넣을 경우

```
-----< cse:tests >-----
Building tests 1.0-SNAPSHOT
  from pom.xml
-----[ jar ]-----

--- resources:3.3.1:resources (default-resources) @ tests ---
skip non existing resourceDirectory C:\Users\walsdn\Documents\NetBeansProjects\tests\src\main\resource

--- compiler:3.11.0:compile (default-compile) @ tests ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ tests ---
Enter your ID: H001
Welcome, Academic Administrator H001! Academic Dashboard is displayed.
-----

BUILD SUCCESS
-----

Total time: 13.811 s
Finished at: 2023-11-24T19:30:36+09:00
-----
```

— 수업 담당자를 넣을 경우

```
-----< cse:tests >-----
Building tests 1.0-SNAPSHOT
  from pom.xml
-----[ jar ]-----

--- resources:3.3.1:resources (default-resources) @ tests ---
skip non existing resourceDirectory C:\Users\walsdn\Documents\NetBeansProjects\tests\src\main\resource

--- compiler:3.11.0:compile (default-compile) @ tests ---
Nothing to compile - all classes are up to date

--- exec:3.1.0:exec (default-cli) @ tests ---
Enter your ID: G001
Welcome, Course Instructor G001! Instructor Dashboard is displayed.
-----

BUILD SUCCESS
-----

Total time: 24.375 s
Finished at: 2023-11-24T19:33:56+09:00
-----
```

**3.2 학생은 수강신청을 할 수 있고, 수강내역 조회 및 수강 취소를 할 수 있으며, 수강료 조회가 가능하다.**

### **3.2.1 분석**

수강신청 및 수강내역 조회를 할 수 있고, 데이터베이스와 연동되어 학생의 수강정보를 등록, 조회, 취소 할 수 있고, 수강료를 조회할 수 있다.

### **3.2.2 구현**

## 관련 코드

```
private void createActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        sel_sleclist.clear();
        for(int i=0; i<selcliclist.size(); i++){ //학생이 수강신청 성공한 배열에 저장
            sel_sleclist.add(new Course(selcliclist.get(i).getCourseNum(), selcliclist.get(i).getCourseName(),
                selcliclist.get(i).getProfessor(), selcliclist.get(i).getGrade(), selcliclist.get(i).getScore()));
        }
        CreateFile();
        insertLecture();
        showMessageDialog(null, "수강신청에 성공하였습니다!!");
        Student_Main_Frame stu = new Student_Main_Frame(nowId, 'S');
        stu.setVisible(true);
        dispose();
    } catch (FileNotFoundException ex) {
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);
    } catch (UnsupportedEncodingException ex) {
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

위 코드는 사용자가 수강신청을 완료하면, 선택한 강의 정보를 배열에 저장하고 파일에 기록한 뒤, 메시지를 통해 성공 메시지를 표시하고 학생 메인 화면을 열어줍니다. 코드는 파일 입출력 및 예외 처리를 포함하고 있습니다.

```
private void deleteActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try {
        //선택한 강의 수강신청할 강의 배열에서 제거
        for (int i = 0; i < selcliclist.size(); i++) {
            if (selcliclist.get(i).getCourseNum().equals(lec_num.getText())) {
                selcliclist.remove(i);
            }
        }
        //선택된 강의 수강신청 가능한 강의 배열에 추가
        clecliclist.add(new Course(lec_num.getText(), lec_name.getText(), pro_name.getText(), lec_grade.getText()));
        //강의 리스트 테이블 업데이트
        a.clec_ex_addList(Course_Table, clecliclist);
        //수강신청한 강의 테이블 업데이트
        a.slec_addList(stu_lect, selcliclist);
        //삭제한 학점 빼기
        sumCredit -= Integer.parseInt(lec_grade.getText());
        Credit_Total.setText(Integer.toString(sumCredit));
    } catch (UnsupportedEncodingException ex) {
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

위 코드는 사용자가 강의를 취소하면 해당 강의를 수강 신청한 배열에서 제거하고, 수강신청 가능한 배열에 추가합니다. 그 후, 강의 목록과 학생이 수강한 강의 목록을 업데이트하고, 삭제한 강의에 해당하는 학점을 합계에서 빼서 화면에 표시합니다. 코드는 배열 조작과 테이블 업데이트, 예외 처리를 수행합니다.



#### 관련 코드

```
private void Course_TableMouseClicked(java.awt.event.MouseEvent evt) {  
    String str;  
    String[] key;  
    String num = a.getKey(Course_Table);  
    try {  
        BufferedReader read = new BufferedReader(new InputStreamReader(new FileInputStream(file), "euc-kr"));  
        while((str=read.readLine())!=null){  
            key = str.split("/");  
            if(key[0].equals(num)){  
                lec_num.setText(key[0]);  
                lec_name.setText(key[1]);  
                pro_name.setText(key[4]);  
                lec_grade.setText(key[3]);  
                lec_info.setText(key[7]);  
            }  
        }  
    } catch (FileNotFoundException ex) {  
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (UnsupportedEncodingException ex) {  
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (IOException ex) {  
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

위 코드는 테이블 셀 클릭 이벤트를 처리하는 메서드로, 파일에서 읽은 데이터를 한글 인코딩 ("euc-kr")으로 처리하여 강의 정보를 GUI에 표시합니다. 클릭한 행의 일련번호를 기준으로 파일을 탐색하며, 해당하는 정보를 각각의 GUI 요소에 할당합니다. 코드는 예외 처리를 통해 파일 읽기 과정에서 발생할 수 있는 오류를 처리합니다.

## 관련 코드

▶ 클릭시 수강신청 강의 리스트에서 수강신청한 강의 목록으로 이동시켜주는 메소드

```
private void insertActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    //if 현재 신청학점 + 추가할 강의 학점 >18  
    boolean maxcheck;  
    try {  
        maxcheck = a.maxCheck(lec_num.getText());  
        if ((sumCredit + Integer.parseInt(lec_grade.getText())) > 18) {  
            //강의를 신청할 수 없습니다.  
            showMessageDialog(null, "더 이상 수강신청을 할 수 없습니다.");  
        } else if (maxcheck) {  
            showMessageDialog(null, "수강 가능 인원이 초과되었습니다.");  
        } else {  
            //강의 리스트에서 선택한 강의 배열에서 삭제  
            for (int i = 0; i < cleclist.size(); i++) {  
                if (cleclist.get(i).getCourseNum().equals(lec_num.getText())) {  
                    cleclist.remove(i);  
                }  
            }  
            //선택된 강의 수강신청한 강의 배열에 추가  
            sleclist.add(new Course(lec_num.getText(), lec_name.getText(), pro_name.getText(), lec_grade.getText()));  
            //강의 리스트 테이블 업데이트  
            a.clec_ex_addList(Course_Table, cleclist);  
            //수강신청한 강의 테이블 업데이트  
            a.slec_addList(stu_lec, sleclist);  
            //수강신청한 강의 배열의 학점 수 만큼 신청학점 추가  
            sumCredit += Integer.parseInt(lec_grade.getText());  
            Credit_Total.setText(Integer.toString(sumCredit));  
        }  
    }  
  
    } catch (UnsupportedEncodingException ex) {  
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (IOException ex) {  
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

구현 결과

▶ 수강 신청 전(수강 정보)


— □ ×

## 수강정보

메인페이지로 돌아가기

강의 번호	강의이름	담당교수	학점	점수
-------	------	------	----	----

▶ 수강 신청 후(수강 정보)

— □ ×

## 수강정보

메인페이지로 돌아가기

강의 번호	강의이름	담당교수	학점	점수
004	지구과학	둘리		

## 구현 결과

### ▶ 수강 신청 전

#### 수강신청 강의 리스트

번호	강의이름	담당교수	학점
001	객체지향프로그래밍	손오공	3
002	파일처리론	손오공	3
004	지구과학	둘리	3

#### 선택강의 정보

강의 이름

지구과학

담당 교수

둘리

강의 번호

004

학점

3

점

강의 설명

지구과학 이론

이민우

신청 학점

0

/18

강의 선택

강의 삭제

수강신청 완료

수강신청 페이지 나가기

▶ 수강 신청 후(과목 클릭 후 강의 선택 누르면 수강신청 강의 리스트에서 수강 신청한 강의로 내려가며, 삭제를 누르면 반대 과정이 실행됨.)

#### 수강신청 강의 리스트

번호	강의이름	담당교수	학점
001	객체지향프로그래밍	손오공	3
002	파일처리론	손오공	3

#### 선택강의 정보

강의 이름

지구과학

담당 교수

둘리

강의 번호

004

학점

3

점

강의 설명

지구과학 이론

이민우

신청 학점

3

/18

강의 선택

강의 삭제

수강신청 완료

수강신청 페이지 나가기

## 구현 결과

▶ 강의 선택 또는 강의삭제 후 수강신청 완료를 누른 결과

**수강신청 강의 리스트**

번호	강의이름	담당교수	학점
001	객체지향프로그래밍	손오공	3
002	파일처리론	손오공	3
004	지구과학	둘리	3
004	지구과학	둘리	3

**선택강의 정보**

강의 이름: 지구과학

담당 교수: 둘리

강의 번호: 004 학점: 3 점

강의 설명: 지구과학 이론

이민우 신청 학점 0 /18

**수강 신청한 강의**

강의 번호	강의 이름	담당 교수	학점
-------	-------	-------	----

강의 선택 강의 삭제

수강신청 완료

수강신청 페이지 나가기

**메시지**

수강신청에 성공하였습니다!!

확인

▶ 수강신청 페이지 나가기 버튼을 누른 결과(메뉴화면으로 복귀)

**학생 메뉴 선택**

수강 정보

수강 신청

수강료 조회

비밀번호변경

로그아웃 이민우

### 3.2.4 테스트

#### ▶ 테스트 코드

```
students.add(new TestPerson("S001"));
students.add(new TestPerson("S002"));

// 학생이 강좌 선택 및 학점 총 합 출력
for (TestPerson student : students) {
    System.out.println("Student ID: " + student.getId());
    student.chooseCourse(courses.get(0)); // Introduction to Computer Science
    student.chooseCourse(courses.get(1)); // Linear Algebra
    student.chooseCourse(courses.get(2)); // Introduction to Psychology
    student.chooseCourse(courses.get(3)); // World History
    student.chooseCourse(courses.get(4)); // English Composition
    System.out.println("Total Credits: " + student.getTotalCredits());
    System.out.println();
}
}

class TestPerson {
    private final String id;
    private List<TestCourse> courses; // 강좌 리스트 추가

    public TestPerson(String id) {
        this.id = id;
        this.courses = new ArrayList<>();
    }

    public String getId() {
        return id;
    }

    public List<TestCourse> getCourses() {
        return courses;
    }

    // 강좌 선택 메서드 추가
    public void chooseCourse(TestCourse course) {
        if (!courses.contains(course) && getTotalCredits() + course.getCredits() <= 18) {
            courses.add(course);
            System.out.println("Course chosen: " + course.getName());
        } else {
            System.out.println("Failed to choose course: " + course.getName());
        }
    }
}
```

▶ 테스트 결과

Student ID: S001  
Course chosen: Introduction to Computer Science  
Course chosen: Linear Algebra  
Course chosen: Introduction to Psychology  
Course chosen: World History  
Course chosen: English Composition  
Total Credits: 15

Student ID: S002  
Course chosen: Introduction to Computer Science  
Course chosen: Linear Algebra  
Course chosen: Introduction to Psychology  
Course chosen: World History  
Course chosen: English Composition  
Total Credits: 15

임의의 학생 객체인 S001과 S002가 수강 신청을 한 모습입니다.

### 3.3 신청 가능한 최대 인원을 초과 하거나, 신청한 학점이 18학점을 넘으면 강의를 더 신청할 수 없다.

#### 3.3.1 분석

신청한 학점이 18학점을 넘으면 강의를 더 신청할 수 없게 하기 위해 학생의 수강정보에 수강학점을 추가하여 조건에 따라 수강신청 가능 여부를 알 수 있게 하였으며, 강의 생성 시 가용 최대인원과 최소인원을 설정하여 최대인원을 초과하였을 경우 강의 신청이 불가능하게 설계 하였다.

### 3.3.2 구현

#### 관련 코드

```
private void insertActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    //if 현재 신청학점 + 추가할 강의 학점 >18  
    boolean maxcheck;  
    try {  
        maxcheck = a.maxCheck(lec_num.getText());  
        if ((sumCredit + Integer.parseInt(lec_grade.getText())) > 18) {  
            //강의를 신청할 수 없습니다.  
            showMessageDialog(null, "더 이상 수강신청을 할 수 없습니다.");  
        } else if (maxcheck) {  
            showMessageDialog(null, "수강 가능 인원이 초과되었습니다.");  
        } else {  
            //강의 리스트에서 선택한 강의 배열에서 삭제  
            for (int i = 0; i < cleclist.size(); i++) {  
                if (cleclist.get(i).getCourseNum().equals(lec_num.getText())) {  
                    cleclist.remove(i);  
                }  
            }  
            //선택된 강의 수강신청한 강의 배열에 추가  
            sleclist.add(new Course(lec_num.getText(), lec_name.getText(), pro_name.getText(), lec_grade.getText()));  
            //강의 리스트 테이블 업데이트  
            a.clec_ex_addList(Course_Table, cleclist);  
            //수강신청한 강의 테이블 업데이트  
            a.slec_addList(stu_lec, sleclist);  
            //수강신청한 강의 배열의 학점 수 만큼 신청학점 추가  
            sumCredit += Integer.parseInt(lec_grade.getText());  
            Credit_Total.setText(Integer.toString(sumCredit));  
        }  
    } catch (UnsupportedEncodingException ex) {  
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (IOException ex) {  
        Logger.getLogger(Application_Frame.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

위 코드는 sel\_sleclist라는 배열에 있는 각 강의 정보를 가져와 강의 번호를 파일명으로 하는 파일에 학생의 아이디, 이름, 학년, 성적 정보를 기록합니다. 파일은 EUC-KR 인코딩으로 작성되며, 파일이 없으면 새로 생성하고 이미 존재하는 경우에는 기존 파일에 내용을 덧붙입니다. 코드는 파일 입출력 및 예외 처리를 다루고 있습니다.



## 구현 결과

▶ 신청 가능한 학점인 18학점을 넘으면 다음과 같이 “더 이상 수강신청을 할 수 없습니다.” 문구가 뜨며 수강신청이 되지 않는다.



The screenshot shows a web application for course selection. On the left, there is a table titled "수강신청 강의 리스트" (Course Selection Course List) with columns: 번호 (No.), 강의이름 (Course Name), 담당교수 (Instructor), and 학점 (Credits). The table lists three courses: 010 (알자역학, 홍길중, 3), 009 (신소재 이론, 마이클, 2), and 005 (화학개론, 딘거, 3). The course 005 is selected. On the right, there is a form titled "선택강의 정보" (Selected Course Information) with fields for 강의이름 (Course Name), 학과개론 (Department Introduction), 담당 교수 (Instructor), 딘거 (Dean), 강의 번호 (Course No.), 학점 (Credits), and 강의 설명 (Course Description). The form shows 005 and 3 credits. Below the form, there is a section titled "수강 신청한 강의" (Courses Applied For) with a table listing applied courses: 004 (지구과학, 돌리, 3), 002 (파일처리론, 손오공, 3), 001 (객체지향프로그래밍, 손오공, 3), 007 (통계, 현우진, 2), 008 (전기회로, 김성우, 3), and 006 (전자회로, 에디슨, 2). A message box is displayed in the center, saying "메시지" (Message) and "더 이상 수강신청을 할 수 없습니다." (No more course applications allowed). The message box has a blue 'i' icon and a "확인" (Confirm) button. At the bottom right, there are buttons for "강의 선택" (Select Course), "강의 삭제" (Delete Course), "수강신청 완료" (Complete Course Selection), and "수강신청 페이지 나가기" (Go to Course Selection Page).

▶ 수강 인원이 초과된 강의를 신청하면 다음과 같이 “수강 가능 인원이 초과되었습니다.” 라는 문구가 뜨며 수강신청이 되지 않는다.



The screenshot shows the same web application for course selection. The "수강신청 강의 리스트" (Course Selection Course List) table now includes an additional course: 011 (회로이론, 권순각, 3). The course 011 is selected. The "선택강의 정보" (Selected Course Information) form shows 011 and 3 credits. The "강의 설명" (Course Description) field now contains the text "회로에 대해 배운다." (Learn about circuits). The "수강 신청한 강의" (Courses Applied For) table now includes an additional course: 012 (수치해석학, 이종민, 3). A message box is displayed in the center, saying "메시지" (Message) and "수강 가능 인원이 초과되었습니다." (The number of students who can take the course has exceeded). The message box has a blue 'i' icon and a "확인" (Confirm) button. At the bottom right, there are buttons for "강의 선택" (Select Course), "강의 삭제" (Delete Course), "수강신청 완료" (Complete Course Selection), and "수강신청 페이지 나가기" (Go to Course Selection Page).

### 3.3.3 테스트

#### ▶ 테스트 코드

```
// 강좌 선택 메서드 추가
public void chooseCourse(TestCourse course) {
    if (!courses.contains(course) && getTotalCredits() + course.getCredits() <= 18) {
        courses.add(course);
        System.out.println("Course chosen: " + course.getName());
    } else {
        System.out.println("Failed to choose course: " + course.getName());
    }
}

// 학점 총 합 계산 메서드 추가
public int getTotalCredits() {
    int totalCredits = 0;
    for (TestCourse course : courses) {
        totalCredits += course.getCredits();
    }
    return totalCredits;
}
}
```

#### ▶ 테스트 결과

```
Student ID: S002
Course chosen: Introduction to Computer Science
Course chosen: Linear Algebra
Course chosen: Introduction to Psychology
Course chosen: World History
Course chosen: English Composition
Total Credits: 15
```

-----  
**BUILD SUCCESS**  
-----

임의의 학생 객체인 S002가 18학점을 초과하지 않게 자동으로 수강신청을 한 모습입니다.

### 3.4 수업담당자는 강좌번호, 강좌이름, 담당학과, 학점, 강좌 설명을 지정하여 새 강좌를 등록할 수 있다.

#### 3.4.1 분석

수업담당자는 강좌의 번호, 이름, 학과, 학점, 설명 등을 적은 후 강좌 등록을 할 수 있어야 한다.

#### 3.4.2 구현

##### 관련 코드

```
private void insertActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    boolean check; //이미 생성한 강좌 비교를 위한 체크 변수  
    String str;  
    DefaultTableModel model = (DefaultTableModel) lecture_list.getModel();  
    try {  
        check = a.checkEquals(lecture_num.getText(), "insertlecturelist.txt"); //이미 생성된 강좌 체크를 위한 Adapter에서 함수 사용  
        if (lecture_num.getText().isEmpty() || lecture_info.getText().isEmpty() || getLec_name.getText().isEmpty() || getCredit.getText().isEmpty()) {  
            //정보가 비어있을 때  
            showMessageDialog(null, "정보를 입력해 주세요");  
        } else if (check) { //이미 강좌번호가 생성되어있을 때  
            showMessageDialog(null, "이미 생성한 강좌 번호입니다.");  
        } else { //강좌 등록  
            FileOutputStream file = new FileOutputStream("insertlecturelist.txt", true); //등록된 강좌 파일 열기  
            OutputStreamWriter output = new OutputStreamWriter(file, text);  
            BufferedWriter writer = new BufferedWriter(output);  
            str = String.format("%s/%s/%s/%s/%s/%s", lecture_num.getText(), getLec_name.getText(),  
                                , depart_list.getSelectedItem(), getCredit.getText(), lecture_info.getText(), false);  
            //강좌 번호, 강좌 이름, 담당 학과, 학점, 강의 설명  
            writer.write(str); // 입력  
            writer.close(); //파일 닫기  
            showMessageDialog(null, "강좌가 등록되었습니다.");  
        }  
        model.setNumRows(0); //테이블 초기화  
        a.lec_AddList(lecture_list); //업데이트 된 리스트 출력  
    } catch (UnsupportedEncodingException ex) {  
        Logger.getLogger(Insert_Lecture.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (IOException ex) {  
        Logger.getLogger(Insert_Lecture.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

위 코드는 사용자가 입력한 강좌 정보를 검증하고, 이미 등록된 강좌인지 확인한 후, 정보가 유효하면 "insertlecturelist.txt" 파일에 새로운 강좌 정보를 추가합니다. 추가된 강좌 정보를 테이블에 업데이트하고 성공 메시지를 표시합니다. 코드는 파일 입출력, 예외 처리, 그리고 테이블 업데이트를 수행합니다.

## 구현 결과

### ▶ 강좌 등록 전

강좌 등록

강좌 정보 입력

강좌 번호

013

강좌 이름

파이썬 기초

담당 학과

전자공학과

학 점

3

강좌 설명

파이썬의 기초에 대해 배운다.

등록

취소하기

강좌 번호	강좌 이름	담당 학과	학점	강의 설명
001	객체지향프로그래밍	전산학과	3	java를 이용한 객체지...
002	파일처리론	전산학과	3	파일처리 이론
003	물리	기계공학과	3	물리실습
004	지구과학	항공우주공학과	3	지구과학 이론
005	화학개론	화학공학과	3	화학 실습 및 이론
006	전자회로	전자공학과	2	회로도 그리기
007	통계	전산학과	2	통계학 개론
008	전기회로	전자공학과	3	전기 공학 실습
009	신소재 이론	항공우주공학과	2	신소재에 관한 이론 ...
010	양자역학	항공우주공학과	3	양자역학 이론 습득
011	e스포츠 과학	전산학과	2	e스포츠 속 과학이야기
012	인문사회학	전산학과	2	인문사회 교양

### ▶ 강좌 등록 후

## 3.4.3 테스트

### ▶ 테스트 코드

```

public class Tests {
    private static final List<TestPerson> students = new ArrayList<>();
    private static final List<TestCourse> courses = new ArrayList<>();

    public static void main(String[] args) {
        // 강좌 생성
        courses.add(new TestCourse(1, "Introduction to Computer Science", "Computer Science", 3));
        courses.add(new TestCourse(2, "Linear Algebra", "Mathematics", 4));
        courses.add(new TestCourse(3, "Introduction to Psychology", "Psychology", 3));
        courses.add(new TestCourse(4, "World History", "History", 3));
        courses.add(new TestCourse(5, "English Composition", "English", 2));
    }

    class TestCourse {
        private final int courseNumber;
        private final String name;
    }
}

```

```
private final String department;
private final int credits;

public TestCourse(int courseNumber, String name, String department, int credits) {
    this.courseNumber = courseNumber;
    this.name = name;
    this.department = department;
    this.credits = credits;
}

public int getCourseNumber() {
    return courseNumber;
}

public String getName() {
    return name;
}

public String getDepartment() {
    return department;
}

public int getCredits() {
    return credits;
}
@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    TestCourse course = (TestCourse) o;
    return courseNumber == course.courseNumber;
}
```

▶ 테스트 결과

```
Student ID: S001
Course chosen: Introduction to Computer Science
Course chosen: Linear Algebra
Course chosen: Introduction to Psychology
Course chosen: World History
Course chosen: English Composition
Total Credits: 15
```

우리가 수업들을 만든 후 임의의 학생 객체인 S001 이 만든 수업들을 고른 모습입니다.

**3.5 수업담당자는 등록된 강좌를 조회하고, 등록된 강좌 내에서 담당교수, 수강 가능 최소/최대 인원을 지정하여 학기별 강의를 개설할 수 있다.**

#### **3.5.1 분석**

등록된 강좌 내에서 담당교수, 수강 가능 최소/최대 인원을 지정하여 학기별 강의를 개설할 수 있다.

#### **3.5.2 구현**

## 관련 코드

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    DefaultTableModel model = (DefaultTableModel) lecture_list.getModel();
    DefaultTableModel model1 = (DefaultTableModel) clecture_list.getModel();
    int row = -1; //행선택 변수
    boolean check; // 이미 개설되어 있는지 확인
    boolean proCheck;
    String str;
    row = lecture_list.getSelectedRow(); //테이블 행 선택 함수
    FileOutputStream file;
    try {
        proCheck = checkPro(); //강의를 담당할 교수의 존재 여부
        check = a.checkEquals(model.getValueAt(row, 0).toString(), "lecturelist.txt");
        //이미 개설된 강좌 체크
        if (row == -1) { // 강좌 미선택시
            showMessageDialog(null, "강좌를 선택해 주세요.");
        } else if (lecture_pro.getText().isEmpty() || max_num.getText().isEmpty() || min_num.getText().isEmpty()) {
            //강좌에 대한 정보를 미입력시
            showMessageDialog(null, "정보를 입력해 주세요.");
        } else if (check) { // 이미 개설된 강좌번호 선택시
            showMessageDialog(null, "이미 개설된 강좌입니다.");
        } else if (!proCheck) { //교수의 존재 여부 확인
            showMessageDialog(null, "존재하지 않는 교수입니다.");
        }
    } else { //강좌 개설
        a.getLectureList(lecList); //개설전 강좌 정보를 lecList에 담기
        file = new FileOutputStream("insertlecturelist.txt");
        BufferedWriter nWriter = new BufferedWriter(new OutputStreamWriter(file, text));
        for (int i = 0; i < lecList.size(); i++) { //lecList만큼 반복
            if (lecList.get(i).getCourseNum().equals((String) model.getValueAt(row, 0))) {
                lecList.get(i).setOpen("true"); //개설한 강좌와 같은 강의 번호가 있으면 개설 여부를 true로 변경
            }
            str = String.format("%s/%s/%s/%s/%s/%s/%s", lecList.get(i).getCourseNum(), lecList.get(i).getCourseName(),
                lecList.get(i).getDepartment(), lecList.get(i).getGrade(), lecList.get(i).getCourse_content(), lecList.get(i).getOpen());
            //개설여부 저장할 위해 다시 파일에 저장
            nWriter.write(str);
        }
        nWriter.close(); // 파일 닫기

        //개설된 강좌를 메모장에 저장
        file = new FileOutputStream("lecturelist.txt", true); //개설 된 강좌 파일 열기
        OutputStreamWriter output = new OutputStreamWriter(file, text);
        BufferedWriter writer = new BufferedWriter(output);
        str = String.format("%s/%s/%s/%s/%s/%s/%s/%s/%s", model.getValueAt(row, 0), model.getValueAt(row, 1),
            model.getValueAt(row, 2), model.getValueAt(row, 3), lecture_pro.getText(),
            max_num.getText(), min_num.getText(), model.getValueAt(row, 4));
        //강좌 번호, 강좌 이름, 담당 학과, 학점, 담당 교수, 최대 인원, 최소 인원, 강의 설명, 개설여부
        writer.write(str);
        writer.close();
        str = (String) model.getValueAt(row, 0) + ".txt";
        File newFile = new File(str); //강좌번호로 이루어진 파일 생성
        newFile.createNewFile();
    }
}
```

```

        showMessageDialog(null, "강좌가 개설되었습니다.");
        model1.setNumRows(0); //테이블 초기화
        a. lec_AddCList(clecture_list); //테이블 업데이트
    }
} catch (FileNotFoundException ex) {
    Logger.getLogger(Insert_Lecture.class.getName()).log(Level.SEVERE, null, ex);
} catch (IOException ex) {
    Logger.getLogger(Insert_Lecture.class.getName()).log(Level.SEVERE, null, ex);
}
}

```

위 코드는 사용자가 선택한 강좌 정보와 개설에 필요한 정보(담당 교수, 최대 인원, 최소 인원)를 검증하고, 이미 개설된 강좌인지와 해당하는 교수의 존재 여부를 확인합니다. 이후, 개설이 가능한 경우 "insertlecturelist.txt" 파일과 "lecturelist.txt" 파일을 업데이트하고, 새로운 파일을 생성하여 강좌 정보를 저장합니다. 마지막으로 테이블을 업데이트하고 성공 메시지를 표시합니다. 코드는 파일 입출력, 예외 처리, 테이블 업데이트 등을 수행합니다.



## 구현 결과

### ▶ 강의 개설 전

Create Lecture

강의 개설

등록된 강의

강좌 번호	강좌 이름	담당학과	학점	강의 설명
001	객체지향프...	인문학과	3	java를 이용...
002	파일처리론	전산학과	3	파일처리 이...
003	물리	기계공학과	3	물리실습
004	지구과학	항공우주공...	3	지구과학 이...
005	화학개론	화학공학과	3	화학 실습 ...
006	전자회로	전자공학과	2	회로도 그리...
007	통계	전산학과	2	통계학 개론 ...
008	전기회로	전자공학과	3	전기 공학 ...
009	신소재 이론	항공우주공...	2	신소재에 관...
010	알자역학	항공우주공...	3	알자역학 이...
011	e스포츠 과학	전산학과	2	e스포츠 속 ...
012	인문사회학	전산학과	2	인문사회 교...

개설된 강의

강좌 번호	강좌 이름	담당 학과	학점	담당 교수	MAX	MIN	강의 설명
001	객체지향프...	전산학과	3	손오공	12	22	java를 이용...
002	파일처리론	전산학과	3	손오공	32	22	파일처리 이론
004	지구과학	항공우주공...	3	물리	20	10	지구과학 이론
006	전자회로	전자공학과	2	에디슨	30	20	회로도 그리기
010	알자역학	항공우주공...	3	홍길동	30	20	알자역학 이...
009	신소재 이론	항공우주공...	2	마이클	30	20	신소재에 관...
007	통계	전산학과	2	현우진	40	20	통계학 개론 ...
005	화학개론	화학공학과	3	딩거	40	20	화학 실습 및...
008	전기회로	전자공학과	3	김성우	40	20	전기 공학 실...
003	물리	기계공학과	3	아이번	1	0	물리실습
011	e스포츠 과학	전산학과	2	페이커	30	15	e스포츠 속 ...

담당 교수

개설

최대 인원

뒤로가기

최소 인원

### ▶ 강의 개설 후

Create Lecture

강의 개설

등록된 강의

강좌 번호	강좌 이름	담당학과	학점	강의 설명
001	객체지향프...	인문학과	3	java를 이용...
002	파일처리론	전산학과	3	파일처리 이...
003	물리	기계공학과	3	물리실습
004	지구과학	항공우주공...	3	지구과학 이...
005	화학개론	화학공학과	3	화학 실습 ...
006	전자회로	전자공학과	2	회로도 그리...
007	통계	전산학과	2	통계학 개론 ...
008	전기회로	전자공학과	3	전기 공학 ...
009	신소재 이론	항공우주공...	2	신소재에 관...
010	알자역학	항공우주공...	3	알자역학 이...
011	e스포츠 과학	전산학과	2	e스포츠 속 ...
012	인문사회학	전산학과	2	인문사회 교...

개설된 강의

강좌 번호	강좌 이름	담당 학과	학점	담당 교수	MAX	MIN	강의 설명
001	객체지향프...	인문학과	3	손오공	12	22	java를 이용...
002	파일처리론	전산학과	3	손오공	32	22	파일처리 이론
004	지구과학	항공우주공...	3	물리	20	10	지구과학 이론
006	전자회로	전자공학과	2	에디슨	30	20	회로도 그리기
010	알자역학	항공우주공...	3	홍길동	30	20	알자역학 이...
009	신소재 이론	항공우주공...	2	마이클	30	20	신소재에 관...
007	통계	전산학과	2	현우진	40	20	통계학 개론 ...
005	화학개론	화학공학과	3	딩거	40	20	화학 실습 및...
008	전기회로	전자공학과	3	김성우	40	20	전기 공학 실...
003	물리	기계공학과	3	아이번	1	0	물리실습
011	e스포츠 과학	전산학과	2	페이커	30	15	e스포츠 속 ...
012	인문사회학	전산학과	2	윤해경	30	15	인문사회 교양

담당 교수

개설

최대 인원

뒤로가기

최소 인원

### 3.6 수업담당자는 기존의 발급 시스템을 이용해 수강료 청구서를 발급할 수 있다.

#### 3.6.1 분석

수강료는 (신청 학점 X 10000) 원으로 지정 후 리스트에 있는 학생을 클릭 후 발급 버튼을 누르면 해당 학생의 청구서가 발급된다.

#### 3.6.2 구현

##### 관련 코드

//청구서 등록

```
private void editActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    FileOutputStream file;  
    boolean check;  
    try {  
        check = checkBill(); //청구서가 있는지 확인  
        if (!check) {  
            file = new FileOutputStream("bill.txt", true); //청구서 파일 열기  
            OutputStreamWriter output = new OutputStreamWriter(file, text);  
            BufferedWriter writer = new BufferedWriter(output);  
            String l = String.format("%s%n", num.getText()); //학번 저장  
            writer.write(l);  
            writer.close();  
            showMessageDialog(null, name.getText() + "님의 청구서 발급이 완료되었습니다.");  
        } else  
            showMessageDialog(null, name.getText() + "님의 청구서가 이미 발급 되었습니다.");  
        num.setText(" ");  
        name.setText(" ");  
        grade.setText(" ");  
        price.setText(" ");  
    } catch (FileNotFoundException ex) {  
        Logger.getLogger(All_Stu_Bill.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (UnsupportedEncodingException ex) {  
        Logger.getLogger(All_Stu_Bill.class.getName()).log(Level.SEVERE, null, ex);  
    } catch (IOException ex) {  
        Logger.getLogger(All_Stu_Bill.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

위 코드는 학번, 이름, 학년, 가격 등의 정보를 입력 받은 후, 이미 청구서가 발급되었는지 확인하고, 발급되지 않았다면 "bill.txt" 파일에 해당 학생의 정보를 기록하여 청구서를 발급합니다. 발급이 완료되면 성공 메시지를 표시하고, 입력 필드를 초기화합니다. 코드는 파일 입출력, 예외 처리, 입력 확인 등을 수행합니다.

구현 결과

▶ 수강료 청구서 발급 전

수강료 청구서 발급

학번	이름	신청 학점	수강료
S001	이민우	9	90000
S754	하태형	3	30000
S605	변정빈	0	0
S315	신홍재	9	90000
S315	신홍재	9	90000

학번

이름

신청 학점

수강료

발급

뒤로가기

▶ 수강료 청구서 발급 후

수강료 청구서 발급

학번	이름	신청 학점	수강료
S001	이민우	9	90000
S754	하태형	3	30000
S605	변정빈	0	0
S315	신홍재	9	90000
S315	신홍재	9	90000

학번

이름

신청 학점

수강료

발급

뒤로가기

메시지



신홍재님의 청구서 발급이 완료되었습니다.

확인

## 3.7 교수는 담당한 강좌에 대한 학생의 성적 입력이 가능하다

### 3.7.1 분석

담당 교수가 해당 과목을 클릭 후 강의 정보 버튼을 누르면 나오는 출석부에서 성적 부여 할 학생을 클릭한 후 성적 입력 버튼을 누르면 A,B,C,D,F 중에 하나를 골라서 성적 부여를 할 수 있다.

### 3.7.2 구현

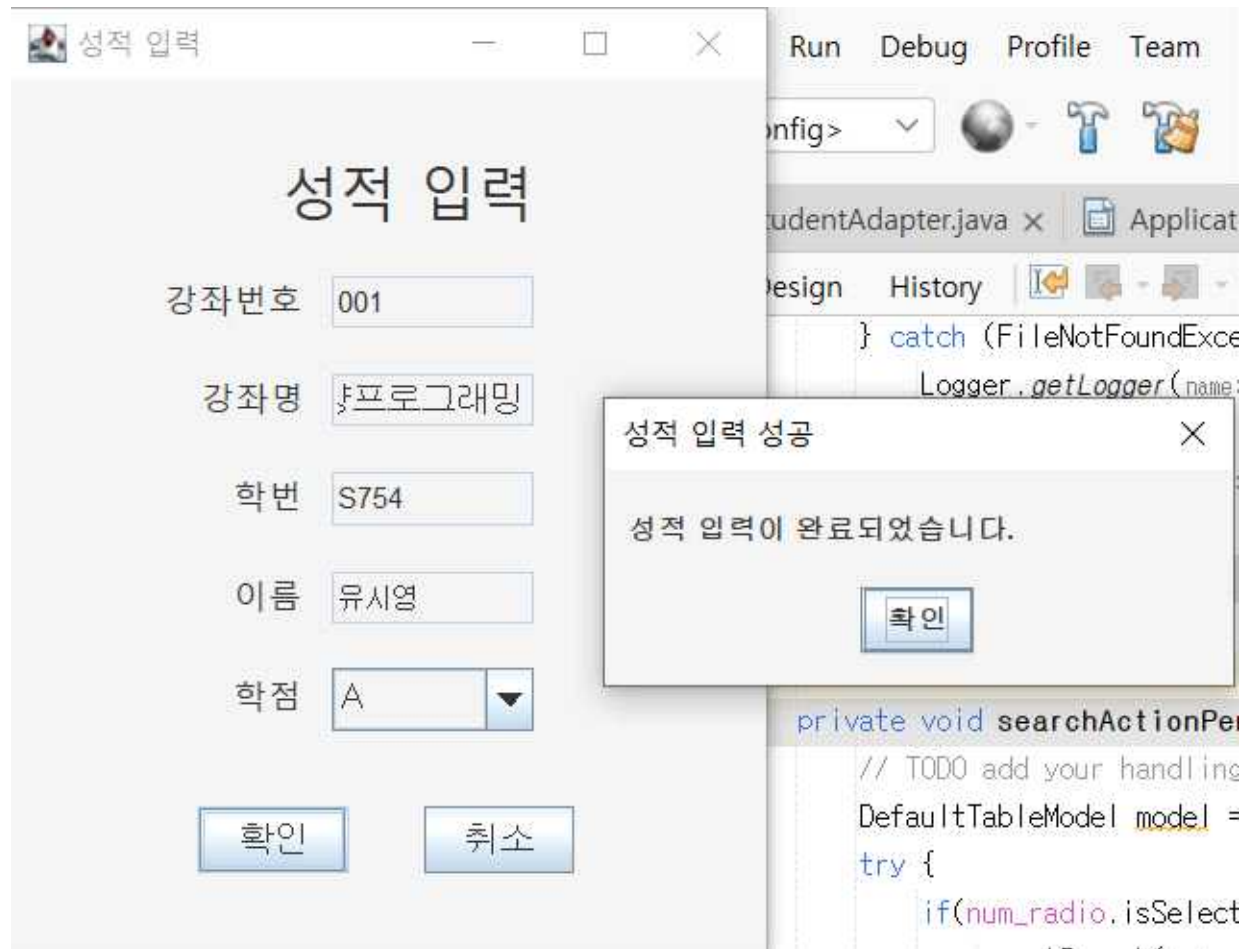
#### 관련 코드

```
private void OK_BtnActionPerformed(java.awt.event.ActionEvent evt) {  
323     // 확인 버튼 기능  
324     int bindex = getBook();  
325     int sindex = getStu();  
326     if (causeNum.getText().isEmpty() || causeName.getText().isEmpty() || studentNum.getText().isEmpty()  
327         || studentName.getText().isEmpty()) {  
328         // 값이 하나라도 입력되지 않았을 경우  
329         JOptionPane.showMessageDialog(parentComponent,null, "정보를 모두 입력해 주세요.", title: "성적 입력 실패", messageType:JOptionPane.WARNING_MESSAGE);  
330     } else {  
331         // 학점별 학점수 초기화  
332         if(grade.getSelectedItem()=="A") {//A학점  
333             bookList.get(index: bindex).setScore(score: "4.0");  
334             bookList.get(index: bindex).setsGrade(sgrade: "A");  
335             stuList.get(index: sindex).setScore(score: "4.0");  
336             stuList.get(index: sindex).setsGrade(sgrade: "A");  
337         }else if(grade.getSelectedItem()=="B") {//B학점  
338             bookList.get(index: bindex).setScore(score: "3.0");  
339             bookList.get(index: bindex).setsGrade(sgrade: "B");  
340             stuList.get(index: sindex).setScore(score: "3.0");  
341             stuList.get(index: sindex).setsGrade(sgrade: "B");  
342         }else if(grade.getSelectedItem()=="C") {//C학점  
343             bookList.get(index: bindex).setScore(score: "2.0");  
344             bookList.get(index: bindex).setsGrade(sgrade: "C");  
345             stuList.get(index: sindex).setScore(score: "2.0");  
346             stuList.get(index: sindex).setsGrade(sgrade: "C");  
347         } else if (grade.getSelectedItem() == "D") {//D학점  
348             bookList.get(index: bindex).setScore(score: "1.0");  
349             bookList.get(index: bindex).setsGrade(sgrade: "D");  
350             stuList.get(index: sindex).setScore(score: "1.0");  
351             stuList.get(index: sindex).setsGrade(sgrade: "D");  
352         } else if (grade.getSelectedItem() == "F") {//F학점  
353             bookList.get(index: bindex).setScore(score: "0.0");  
354             bookList.get(index: bindex).setsGrade(sgrade: "F");  
355             stuList.get(index: sindex).setScore(score: "0.0");  
356             stuList.get(index: sindex).setsGrade(sgrade: "F");  
357         }  
358         insertList();//파일에 저장  
359         JOptionPane.showMessageDialog(parentComponent,null, "성적 입력이 완료되었습니다.", title: "성적 입력 성공", messageType:JOptionPane.PLAIN_MESSAGE); // 성  
360         AttendanceBook b = new AttendanceBook(nowId, lecNum: nowNum, lecName);  
361         b.setVisible(b: true);  
362         dispose();  
363     } // 성적 입력 실패
```

위 코드는 사용자가 입력한 정보(책 번호, 학번, 학점 등)를 검증하고, 입력이 완료되면 해당 학생과 책의 성적 정보를 업데이트하고, 파일에 저장합니다. 학점에 따라 학점수와 학점 등급이 설정되며, 입력이 완료되면 성공 메시지를 표시하고 새로운 창을 엽니다. 코드는 파일 입출력, 예외 처리, 입력 확인 등을 수행합니다.

구현 결과

▶ 성적 부여 결과



### 3.8 교수는 자신이 담당하 강의의 출석부 조회 결과로 검색을 통해 수강생의 학번, 이름, 취득학점을 확인할 수 있다.

#### 3.8.1 분석

담당 교수가 해당 과목을 클릭 후 강의 정보 버튼을 누르면 학생들의 학번, 이름, 취득학점을 확인할 수 있다.

#### 3.8.2 구현


##### 관련 코드

```
18 public class AttendanceBook extends javax.swing.JFrame{
19     DefaultTableModel model;
20     String nowId;
21     String nowNum;
22
23     public AttendanceBook(String nowId, String lecNum, String lecName) {
24         initComponents();
25         this.nowId = nowId;
26         this.nowNum = lecNum;
27         lec_name.setText(text: lecName);
28         addTable();// 테이블에 정보 추가
29     }
30     public void addTable(){ // 테이블에 강의에대한 출석부 추가
31         model = (DefaultTableModel) Attendance_table.getModel();
32         model.setNumRows(rowCount:0);
33         String str;
34         String[] key;
35         try {
36             String file = String.format(format: "%s.txt", args: nowNum);
37             BufferedReader read = new BufferedReader(new InputStreamReader(new FileInputStream(name:file), charsetName:"euc-kr"));
38             while((str = read.readLine())!= null){
39                 key = str.split(regex: "/");
40                 String[] list = {key[0], key[1],key[2],key[3]};
41                 model.addRow(rowData:list);
42             }
43         } catch (FileNotFoundException ex) {
44             Logger.getLogger(name:AttendanceBook.class.getName()).log(level:Level.SEVERE, msg:null, thrown: ex);
45         } catch (UnsupportedEncodingException ex) {
46             Logger.getLogger(name:AttendanceBook.class.getName()).log(level:Level.SEVERE, msg:null, thrown: ex);
47         } catch (IOException ex) {
48             Logger.getLogger(name:AttendanceBook.class.getName()).log(level:Level.SEVERE, msg:null, thrown: ex);
49         }
50     }
```

위 코드는 클래스는 강의 번호(lecNum), 강의 이름(lecName), 현재 사용자의 아이디(nowId), 그리고 현재 사용자의 학번(nowNum)을 받아 초기화됩니다. addTable 메서드는 파일에서 출석부 정보를 읽어와 테이블에 추가하는 역할을 수행합니다. Attendance\_table은 Swing에서 제공하는 테이블 컴포넌트로, 강의에 대한 출석 정보를 표시합니다. 코드는 파일 입출력 및 테이블 업데이트를 다루고 있습니다.

구현 결과

▶ 출석부 결과

— □ ×

객체지향프로그래밍

학번	이름	학점	점수
S315	신홍재		
S754	유시영		
S605	변정빈		

성적입력

뒤로가기



## 3.9 학사 담당자는 새 학생 및 교수를 등록 및 수정, 삭제할 수 있다.

### 3.9.1 분석

학사 담당자는 등록 버튼을 누르고 학생과 교수중 추가하고 싶은 직군의 라디오 버튼을 고른 뒤 학과, 이름, 주민 번호를 누르면 등록이 가능하고, 메인 화면에서 직군을 선택 후 조회 및 수정 버튼을 클릭한 뒤 수정 및 삭제를 하고싶은 사람을 클릭한 뒤 내용 수정 후 수정 및 삭제 버튼을 누르면 해당 기능이 실행된다.

### 3.9.2 구현

#### 관련 코드

##### ▶ 사용자 등록

```
private void insert_OkActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    String str;
    if(name.getText().isEmpty()){
        showMessageDialog(null,"이름을 입력해주세요!!");
    }else if(pNum.getText().isEmpty() || pNum.getText().length() != 13){
        showMessageDialog(null,"주민번호를 입력해주세요!!");
    }else{
        if(stu_butt.isSelected()){ //학생 파일에 입력
            try{
                FileOutputStream file = new FileOutputStream("student.txt",true);
                OutputStreamWriter output = new OutputStreamWriter(file,"euc-kr");
                BufferedWriter writer = new BufferedWriter(output);
                Student stu = new Student(pNum.getText(), name.getText(),
depart_list.getSelectedItemAt(0).toString());
                str = String.format("%s/%s/%s/%s/%s\n",stu.getId(),stu.getName(),
stu.getPassWord(),stu.getPeopleNum(),stu.getDepartMent());
                writer.write(str); //파일에 객체 정보를 저장
                writer.close();
                showMessageDialog(null,stu.getId() + " " + stu.getName() +
"학생이 등록 되었습니다.");
                a.createFile(stu.getId());
            }catch(IOException e){
                e.printStackTrace();
            }
        }else if(pro_butt.isSelected()){ //교수 파일에 입력
            try{
```



```

        FileOutputStream file = new FileOutputStream("professor.txt",true);
        OutputStreamWriter output = new OutputStreamWriter(file,"euc-kr");
        BufferedWriter writer = new BufferedWriter(output);
        Professor pro = new Professor(pNum.getText(), name.getText(),
depart_list.getSelectedItem().toString());

        str =
String.format("%s/%s/%s/%s/%s%n",pro.getId(),pro.getName(),pro.getPassWord(),pro.getPeopl
eNum(), pro.getDepartMent());

        writer.write(str); //파일에 객체 정보를 저장
        writer.close();
        showMessageDialog(null,pro.getId() + " " + pro.getName() + "교수가 등
록 되었습니다.");
    }catch(IOException e){
        e.printStackTrace();
    }
}
else{
    showMessageDialog(null,"직군을 선택하지 않았습니다.");
}
}
}
}

```

위 코드는 사용자가 이름, 주민번호, 학과를 입력하고 직군(학생 또는 교수)을 선택한 경우, 해당 정보를 파일에 추가하여 새로운 학생 또는 교수를 등록합니다. 등록된 정보에 대한 메시지를 표시하고, 필요한 경우 파일에 대한 예외 처리를 수행합니다. 코드는 파일 입출력 및 예외 처리를 다루고 있습니다.

#### ▶ 사용자 수정

```

private void stu_exchangeActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    FileOutputStream wfile;
    String str;
    String id = a.getKey(stu_list);
    try {
        a.getStuList(stulist);
        if(stu_num.getText().isEmpty()){
            showMessageDialog(null, "학생을 선택해 주세요.");
        }
        else if(stu_num.getText().length()!=4){
            showMessageDialog(null, "학번이 3자리가 아닙니다.");
        }
    }
}

```

```

    }
    else if(stu_peoplenum.getText().length()!=13){
        showMessageDialog(null, "주민번호의 자리수가 일치하지않습니다.");
    }
    else{
        wfile = new FileOutputStream(file);
        BufferedWriter writer = new BufferedWriter(new
OutputStreamWriter((wfile), "euc-kr"));
        for(int i = 0; i < stulist.size(); i++){//id이랑 일치하는 객체 정보 변경
            if(stulist.get(i).getId().equals(id)){
                stulist.get(i).setName(stu_name.getText());
                stulist.get(i).setId(stu_num.getText());

stulist.get(i).setDepartMent(stu_department.getSelectedItem().toString());
                stulist.get(i).setPeopleNum(stu_peoplenum.getText());
            }
            str = String.format("%s/%s/%s/%s/%s%n",
stulist.get(i).getId(), stulist.get(i).getName(),
stulist.get(i).getPassWord(),stulist.get(i).getPeopleNum(),stulist.get(i).getDepartMent());
            writer.write(str); }
            a.exList(id, stu_num.getText(),stu_name.getText(),
stu_department.getSelectedItem().toString(),stu_peoplenum.getText());
            writer.close(); //파일 닫기
            clearStuInfo();
            a.sp_AddList(stu_list, file);
        }
    } catch (FileNotFoundException ex) {
        Logger.getLogger(ExChange_Stu_Info.class.getName()).log(Level.SEVERE, null, ex);
    } catch (UnsupportedEncodingException ex) {
        Logger.getLogger(ExChange_Stu_Info.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(ExChange_Stu_Info.class.getName()).log(Level.SEVERE, null, ex);
    }
}
}

```

위 코드는 사용자가 입력한 학생 정보로 파일을 업데이트하고 목록을 갱신합니다.  
 선택한 학생의 기존 정보를 새로 입력한 정보로 변경하고, 이를 파일에 업데이트합니다.  
 변경된 정보를 테이블에 반영하고, 사용자에게 성공적으로 변경되었다는 메시지를 표시합니다.  
 유효성 검사를 통해 학번이 4자리가 아니거나 주민번호의 자리수가 일치하지 않는 경우에는 적절  
 한 경고 메시지를 표시합니다.

## 관련 코드

### ▶ 사용자 삭제

```
private void stu_deleteActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    String str;
    String key = null;
    FileOutputStream wfile;
    try {
        if(stu_num.getText().isEmpty()){
            showMessageDialog(null, "학생을 선택해 주세요");
        }
        else{
            key = a.getKey(stu_list);
            a.getStuList(stulist);
            for(int i = 0; i<stulist.size(); i++){
                if(key.equals(stulist.get(i).getId()))
                    stulist.remove(i);
            }
            wfile = new FileOutputStream(file);
            BufferedWriter writer = new BufferedWriter(new OutputStreamWriter((wfile), "euc-kr"));
            for(int i = 0; i< stulist.size();i++){ //리스트의 크기만큼 실행
                str = String.format("%s/%s/%s/%s/%s/%s\n",
stu_list.get(i).getId(), stu_list.get(i).getName(), stu_list.get(i).getPassWord(),stu_list.get(i).
getPeopleNum(),stu_list.get(i).getDepartMent());
                writer.write(str);//메모장에 쓰기
            }
            writer.close(); //닫기
            clearStuInfo();
            a.sp_AddList(stu_list, file);
            showMessageDialog(null, "삭제가 완료되었습니다.");
        }
    } catch (FileNotFoundException ex) {
        Logger.getLogger(ExChange_Stu_Info.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(ExChange_Stu_Info.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

위 코드는 선택한 학생의 정보를 목록에서 삭제하고, 해당 정보를 파일에서도 제거합니다. 삭제된 정보를 테이블에 반영하고, 사용자에게 삭제가 완료되었다는 메시지를 표시합니다. 유효성 검사를 통해 학번이 입력되지 않은 경우에는 적절한 경고 메시지를 표시합니다.

## 구현 결과

### ▶ 사용자 등록

The image shows a Windows-style application window titled "사용자 등록" (User Registration). It contains the following elements:

- Radio buttons for "학생" (Student) and "교수" (Professor), with "학생" selected.
- A dropdown menu for "전산학과" (Computer Science Department).
- Input fields for "이름" (Name) with the value "아무개" (Aimagae) and "주민번호" (Resident Number) with the value "1234567891234".
- Buttons for "등록" (Register) and "취소" (Cancel).

Overlaid on the bottom of the registration window is a "메시지" (Message) dialog box with an information icon. It displays the text: "S384 아무개학생이 등록 되었습니다." (S384 Aimagae student has been registered.) and includes an "확인" (OK) button.

구현 결과

▶ 사용자 수정 전

☐ 학번
 ☐ 이름
 ☒ 전체

검색

학번

S384

이름

아무개

학과

전산학과

주민등록번호

1234567891234

수정

삭제

뒤로가기

학번	이름	학과	주민등록번호
S754	유시영	전자공학과	1234567891234
S605	변정빈	화학공학과	1234567891234
S315	신홍재	기계공학과	1234567891234
S915	장준혁	항공우주공학과	1234567891234
S778	김민석	기계공학과	1234567891234
S384	아무개	전산학과	1234567891234

▶ 사용자 수정 후

☐ 학번
 ☐ 이름
 ☒ 전체

검색

학번

S384

이름

홍길동

학과

전자공학과

주민등록번호

1234567891234

수정

삭제

뒤로가기

학번	이름	학과	주민등록번호
S754	유시영	전자공학과	1234567891234
S605	변정빈	화학공학과	1234567891234
S315	신홍재	기계공학과	1234567891234
S915	장준혁	항공우주공학과	1234567891234
S778	김민석	기계공학과	1234567891234
S384	홍길동	전자공학과	1234567891234

## 구현 결과

### ▶ 사용자 삭제 전

☐ 학번
 ☐ 이름
 ☒ 전체

검색

학번

S384

이름

홍길동

학과

전자공학과 ▼

주민등록번호

1234567891234

수정

삭제

뒤로가기

학번	이름	학과	주민등록번호
S754	유시영	전자공학과	1234567891234
S605	변정빈	화학공학과	1234567891234
S315	신홍재	기계공학과	1234567891234
S915	장준혁	항공우주공학과	1234567891234
S778	김민석	기계공학과	1234567891234
S384	홍길동	전자공학과	1234567891234

### ▶ 사용자 삭제 후

☐ 학번
 ☐ 이름
 ☒ 전체

검색

학번

이름

학과

전산학과 ▼

주민등록번호

수정

삭제

뒤로가기

학번	이름	학과	주민등록번호
S754	유시영	전자공학과	1234567891234
S605	변정빈	화학공학과	1234567891234
S315	신홍재	기계공학과	1234567891234
S915	장준혁	항공우주공학과	1234567891234
S778	김민석	기계공학과	1234567891234

메시지

i

삭제가 완료되었습니다.

확인

### 3.10 학사 담당자는 학생과 교수 정보를 학번/교수 번호와 이름으로 검색 및 조회할 수 있다.

#### 3.10.1 분석

학사 담당자는 학번/교수 번호로 검색할 때는 직군을 제외한 숫자 3자리를 통하여 검색할 수 있고 이름으로도 검색을 할 수 있다.

#### 3.10.2 구현

##### 관련 코드

##### ▶ 사용자 조회

```
private void searchActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    DefaultTableModel model = (DefaultTableModel)stu_list.getModel();
    try {
        if(num_radio.isSelected()){ //학번으로 찾기 검색
            a.getSearch(0, searchfield.getText(), stu_list,"S",file);
        }
        else if(name_radio.isSelected()){ //이름으로 찾기 검색
            a.getSearch(1, searchfield.getText(), stu_list,"",file);
        }
        else if(all_radio.isSelected()){ //전체 선택시 학생 전체리스트 띄우기
            a.sp_AddList(stu_list, file);
        }
        searchfield.setText(null);
    } catch (UnsupportedEncodingException ex) {
        Logger.getLogger(ExChange_Stu_Info.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(ExChange_Stu_Info.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

위 코드는 사용자가 선택한 검색 옵션(학번, 이름, 전체)에 따라 검색을 수행합니다.

검색 결과를 학생 목록 테이블에 반영하여 표시합니다.

getSearch 메서드를 사용하여 검색을 처리하며, 파일에서 정보를 읽어와 테이블에 업데이트합니다.

유효성 검사를 통해 검색어가 입력되지 않은 경우에는 아무 작업도 수행하지 않습니다.

▶ 사용자 조회 (학번/교수 번호) 검색 버튼 누르기 전

▶ 사용자 조회 (학번/교수 번호) 검색 버튼 누른 후

## 학생 정보 조회 및 수정

☒ 학번    ☐ 이름    ☐ 전체

학번

이름

학과

전산학과
▼

주민등록번호

수정

삭제

뒤로가기

학번	이름	학과	주민등록번호
S754	유시영	전자공학과	1234567891234



▶ 사용자 조회 (이름) 검색 버튼 누르기 전

▶ 사용자 조회 (이름) 검색 버튼 누른 후

[illegible]

## 3.11 7자리 영문자 및 숫자로 구성된 비밀번호로 변경할 수 있다

### 3.11.1 분석

학사 관리자가 사용자를 추가할 때 초기 비밀번호가 주민번호 뒷자리로 되어 있으므로 사용자가 자신이 쓰고싶은 비밀번호로 설정 할 수 있어야 한다.

### 3.11.3 구현

관련 코드

#### ▶ 비밀번호 변경

```
private void changeActionPerformed(java.awt.event.ActionEvent evt)
{
    int index = 0; // 사용자 목록에서 현재 사용자의 인덱스를 저장하는 변수

    try {
        // TODO add your handling code here:
        if (now_pw.getText().isEmpty() || new_pw.getText().isEmpty() ||
        check_pw.getText().isEmpty())
            showMessageDialog(null, "빈칸을 입력해주세요.");
        else {
            // 현재 사용자의 인덱스를 찾기 위해 사용자 목록을 순회
            for (int i = 0; i < userList.size(); i++) {
                if (userList.get(i).getId().equals(nowId)) {
                    index = i;
                }
            }
            // 현재 비밀번호 확인
            if (userList.get(index).getPassWord().equals(now_pw.getText())) {
                // 새로운 비밀번호와 확인 비밀번호가 일치하는지 확인
                if (new_pw.getText().equals(check_pw.getText())) {
                    // 비밀번호 길이 확인 (7자리로 제한)
                    if (new_pw.getText().length() == 7) {
                        // 비밀번호 변경
                        userList.get(index).setPassWord(new_pw.getText());
                        insertList(filename);
                        showMessageDialog(null, "비밀번호가 변경되었습니다.");
                    } else {
                        showMessageDialog(null, "비밀번호는 7자리로 입력해주세요. 변경되지
                        않았습니다.");
                    }
                } else {
                    } else {
```

```

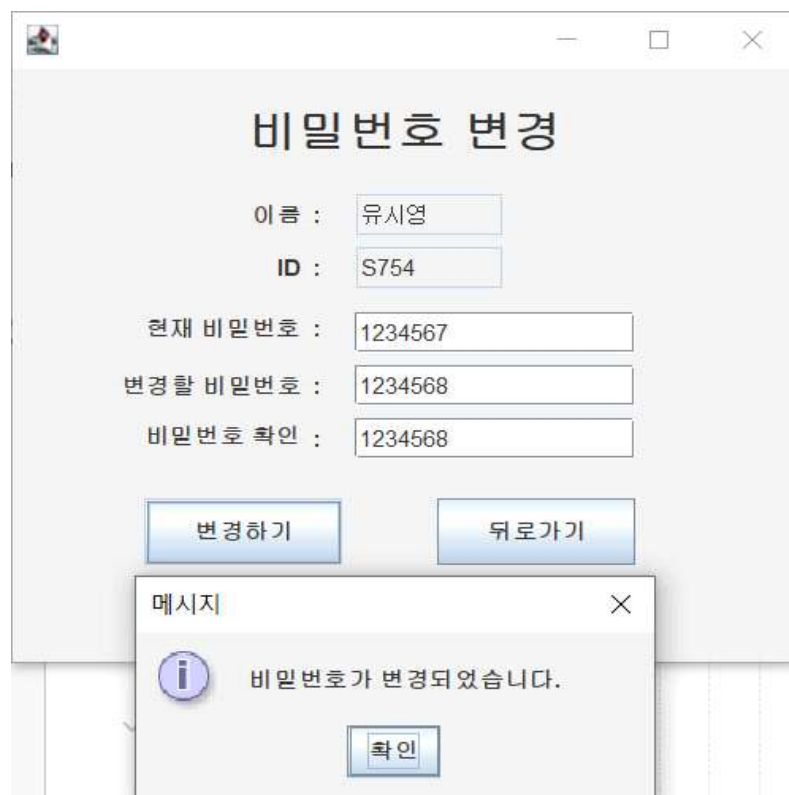
        showMessageDialog(null, "비밀번호 확인이 일치하지 않습니다.");
    }
} else {
    showMessageDialog(null, "현재 비밀번호가 일치하지 않습니다.");
}
}
exit(); // 비밀번호 변경 후 종료
} catch (IOException ex) {
    Logger.getLogger(Exchange_Pw.class.getName()).log(Level.SEVERE, null, ex);
}
}
}

```

위 코드는 현재 비밀번호, 새로운 비밀번호, 확인 비밀번호를 입력받아 비밀번호 변경을 시도합니다. 현재 사용자의 인덱스를 찾기 위해 사용자 목록을 순회합니다. 현재 비밀번호가 일치하는지 확인하고, 일치할 경우 새로운 비밀번호와 확인 비밀번호가 일치하는지 확인합니다. 비밀번호 길이가 7자리로 제한되어 있으며, 이를 확인합니다. 변경된 비밀번호를 사용자 목록에 반영하고, 파일에 저장합니다. 변경이 완료되면 비밀번호가 변경되었음을 알리는 메시지를 표시하고, 프로그램을 종료합니다.

#### 구현 결과

##### ▶ 비밀번호 변경

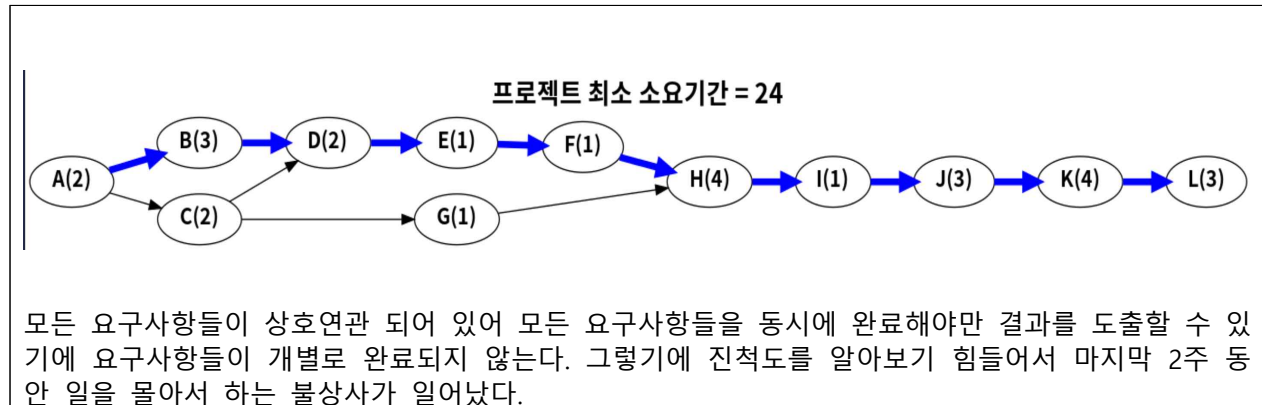


4. 프로젝트 평가

4.1 프로젝트 완성도

추적 가능성 표		완성도	우수성
요구사항	F-001	100%	상
	F-002	100%	상
	F-003	100%	상
	F-004	100%	상
	F-005	100%	상
	F-006	100%	상
	F-007	100%	상
	F-008	100%	상
	F-009	100%	상
	F-010	100%	상
	F-011	100%	상
	F-012	100%	상
	F-013	100%	상
자체 완성도 평가		100%	상

## 4.2 일정 계획 평가



## 4.3 역할 수행 평가

이름	담당 역할	비고
변정빈	- 사용자 추가와 제거하는 사용자 관리를 구현함.	.
신홍재	- 학사 담당자가 학생과 교수의 정보를 검색하고 수정 및 삭제를 구현함.	.
이민우	- 수업담당자가 개설된 강의의 학생 수 MAX, MIN 지정을 구현함.	.
장준혁	- 학생, 교수의 수강 신청, 강좌기능 관리를 구현함.	.
하태형	- 학생, 교수, 직원의 로그인 관리와 사용자 정보를 관리를 구현함.	.

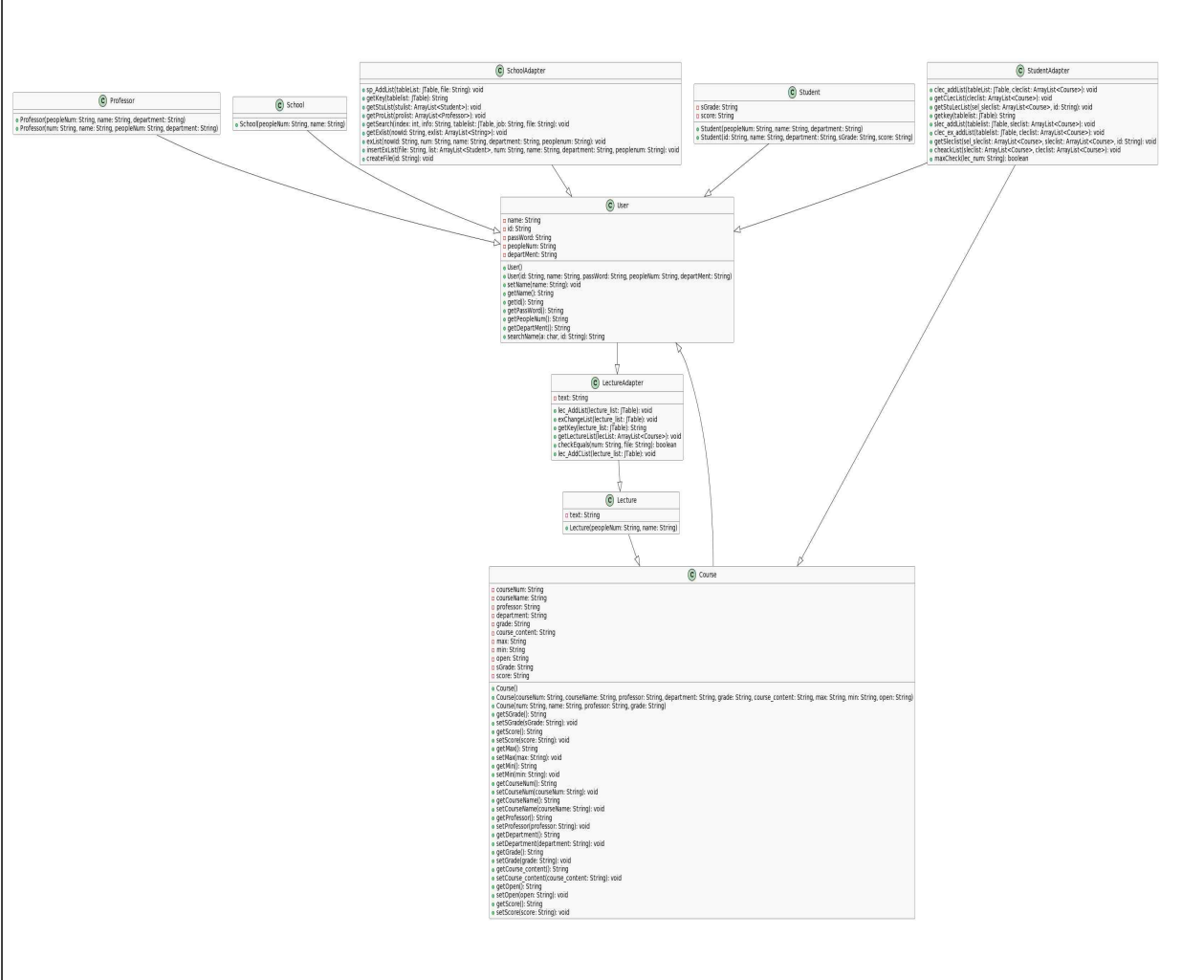
## 4.4 소스 코드 버전 제어 도구 사용

사용 내역 없음
----------

## 4.5 설계 구성요소

항목	내 용
분석	문제 정의와 작업 분해를 통하여 프로젝트 주제에 대한 목적 이해 및 해야 할 일을 계획할 수 있다.
	대학 정보 시스템 문제기술서 파일을 보고 요구사항 들을 리스트화 하였다
시스템 설계	SW의 유연성, 유지보수성과 재사용성을 높이기 위하여 SW의 전체 구조를 클라이언트-서버 구조로 설계할 수 있다.
	코드가 늘어날 경우 수정이 어려워질 가능성이 높고 재사용성이 낮다.
구현	분석과 설계 산출물을 사용하여 정해진 일정에 맞게 소프트웨어를 구현할 수 있

	다.사용자 권한별 화면을 나누고 그 안에 들어가는 기능을 나눈 후 프로젝트 생성 시 패키지를 나누어 그 안에 권한별로 프로그램을 구현하였다. 클래스 다이어그램은 다음과 같다. (목차 2의 시스템 부분의 시스템 소프트웨어 구조 부분과 같음)
--	---



## 4.6 현실적 제한조건

항목	내 용
생산성	개발 과정에 생산성을 높일 수 있는 도구를 적용하였는지, 그리고 산출물을 활용하여 기존 시스템에 비해 생산성을 높일 수 있는지를 기술한다. 웹 응용 프로그램을 작성하기 위하여 모든 구성요소를 직접 프로그램하지 않고 기존의 잘 알려진 시스템 또는 라이브러리를 활용하여 코딩의 효율성 및 테스트가 보다 간편해짐을 이해하고 활용할 수 있다.
	NetBeans IDE를 오류 개선의 용이성과 작업 효율성을 높였다. PlantUML의 클래스 다이어그램 생성을 사용하여 클래스 다이어그램을 더 편리하게 만들어 클래스 간의 관계도 작성이 용이했다. JFrame폼 클래스 파일을 사용하여 직관적인 GUI 생성에 도움이 되었다. 개발 시 구조를 비슷하게 맞추으로써 유지보수가 용이했다.
산업표준	개발 프로세스나 개발 산출물에 아래와 같이 산업표준/국가표준/국제표준을 고려하였는지, 고려하였다면 어떻게 했는지를 기술한다. 1. 개발 환경은 Eclipse/NetBeans IDE를 사용하여야 한다. 2. 형상 관리, 테스트, 문서화를 적용하여야 한다. 3. 클래스 다이어그램을 사용하여 프로그램의 아키텍처를 표현해야 한다. 4. 서버 프로그램과 클라이언트 프로그램은 TCP 또는 UDP 소켓 프로그래밍을 사용하여 구현한다. 5. 서버/클라이언트 프로그램은 강의 시간에 배운 Java SE를 활용한다.
	NetBeans IDE를 사용하여 개발하였다. 깃을 이용하여 형상 관리를 하였고, 테스트 드라이브를 만들어 테스트하였다. 클래스 다이어그램을 사용하여 프로그램의 전체 구조를 표현했다.

## 5. 소감

자바 스윙을 이용하여 GUI를 만드는 것에 대해 알게 되어서 좋았다.

요구사항들을 객체별로 나누었는데 생각하던 대로 개발이 이루어지지 않았던 경험을 통하여 요구사항을 만드는 것이 개발하는 단계에 영향을 크게 미친다는 것을 깨달았다.

QA 요구사항들을 단계별로 완성할 수 있도록 구성하지 않아 프로젝트 진행에 어려움을 느꼈지만, 프로젝트를 만들며 자바 언어를 수월하게 알게 되었다