Adventure
Design

출석부 정리 프로그램 2021

LEVEL 02

이름 김다은
학번 20200123
학과 컴퓨터 공학과

문제

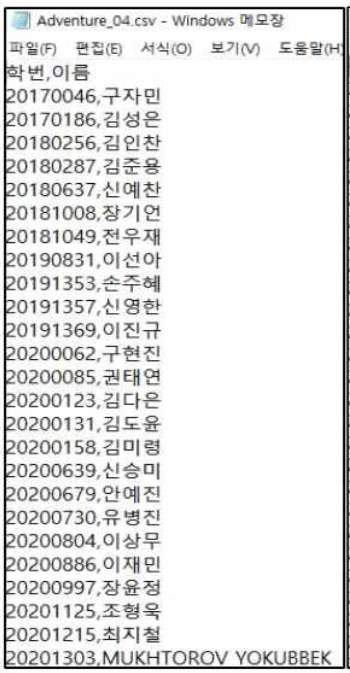
1. 문제 소개

가. 문제

온라인 회의 플랫폼에서 제공하는 접속 기록을 사용해 출석부 정리를 한다. 접속 기록을 처리하여 각 학생별 출석상황(지각, 결석, 정상 출석)을 쉽게 알 수 있도록 csv 파일로 저장하고 GUI를 통해 출결 상황 확인, 수업 시작 시간 설정 등을 가능하도록 한다.

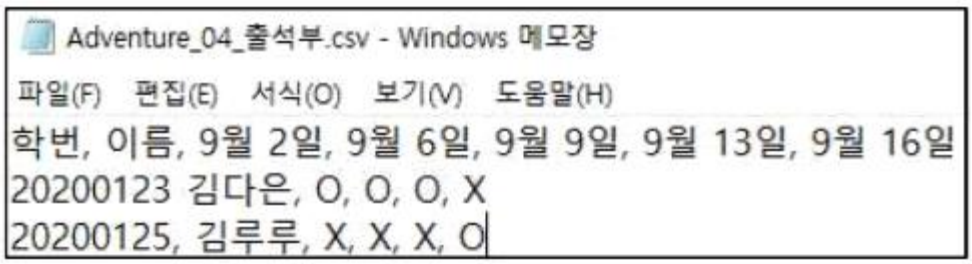
나. 입력

- 수업_분반.csv : 수강 학생 전원의 학번과 이름이 담김
- 수업_분반_날짜.csv : 미팅 참여자의 정보와 들어온 시간 나간 시간이 담김



다. 출력

- 수업_분반_출석부.csv : 학번 이름 그리고 날짜별 출석 현황을 알려줌



2. 주의 사항 및 고려할 점

1) 동명이인의 처리

가) 관리자가 입력한 학생 정보가 담긴 CSV 파일(수업_분반.csv 형식)에서 이름이 동일하고 학번이 다른 한명 이상의 학생이 존재할 경우 이를 동명 이인으로 처리한다. 동명이인을 구분할 때에는 이름이 아닌 학번을 기준으로 구분한다. 학번은 부가적인 정보이므로 반드시 이름을 우선으로 구분하고 동명이인일 경우에 학번을 비교하도록 한다.

나) 동명이인에 대한 것을 구분할 방법에 대해서 고려해야 한다. 전체 학생을 읽은 후 같은 이름을 가진 사람이 몇 명인지 알아내야 한다. 만약 특정 문자열을 1개 이상 가지고 있는 배열 일 경우 해당 문자열은 동명이인 이라고 판단할 수 있다. unique 하지 않은 이름 문자열을 찾아 해당 학생이 동명이인이라는 정보를 알 수 있도록 한다.

2) 지각에 대한 처리

가) 참여자가 회의에 참여한 시간 그 이후 정각을 기준으로 출석을 판단한다. 만약 회의에 참여한 시간 그 이전에 참여하였을 경우 정상 출석, 정각 기준 출석 시간보다는 늦었으나 참여한 시간으로부터 십분이 지나기 전 참여하였을 경우 지각, 회의에 참여하지 않았거나 회의 전에 참여하였을 경우 결석으로 처리한다. 지각 네 번은 결석 한 번으로 처리한다.

3) 학생별 출석률과 날짜별 출석률의 계산과 파일 기록

가) 학생별로 전체 수업에 대해 출석률이 얼마인지 전체 입력된 출석부 수업에서 수업별 출석률이 얼마인지 확인할 수 있어야 한다. 전체 학생의 처음 입장 시간 마지막 퇴장 시간 그리고 회의 참석 시간을 학생별로 저장할 수 있도록 하고 입장 시간을 기준으로 출석률을 계산한다. 수업 파일 마다 학생의 출석 여부가 저장되어야 하고 “(각각 다른 날의 수업에 대해 한 학생의 총 출석 합) / 총 입력된 수업의 수” 가 학생별 출석률이 된다. “(한 수업에 참여한 전체 학생의 출석의 합) / 총 입력된 학생의 수”가 날짜별 출석률이 된다.

4) 잘못된 입력에 대한 처리

(가) 영어 이름 혹은 오타 등의 잘못된 입력이 들어올 경우 이를 사용자에게 알릴 방법에 대해 고려하여야 한다. 어떻게 하면 사용자가 해당 수업에 어떤 잘못 표기된 사용자가 있으며 그 사용자에 대한 처리가 어떻게 이루어져야 출석 판단에 효과적일지 고려하여야 한다.

기능 및 요구 사항 분석

1. 입출력 정의

1) CSV 파일

CSV(comma separated values) 파일이란 ‘,’로 분리되어 기록된 파일이다. 엑셀을 통해 파일을 읽거나 쓸 수 있다.

2) 입력 정의

가) 강의의 출석 학생 명단과 학번이 입력된 파일.

파일 이름 형식 : 강의명_분반.csv

입력 정보 : 학번과 이름이 쉼표로 구분되어 있는 파일

나) 해당 날짜의 zoom 수업에 출석한 학생들의 출석 기록 파일

파일 이름 형식 : 강의명_분반_mmss.csv

입력 정보 : 이름, 사용자 이메일, 참가 시간, 나간 시간, 기간, 게스트 정보가 쉼표를 기준으로 구분되어 있는 파일

3) 출력 정의

가) 해당 강의에 입력된 수업 기록 각 학생의 출석 여부, 파일별 출석률과 학생별 출석률이 담긴 파일

파일 이름 형식 : 강의명_분반.csv

출력 정보 : 학번과 이름 그리고 입력된 출석 파일의 날짜, 전체 학생 출석률 그리고날짜별 출석률이 입력된 파일

나) 수업별로 잘못된 출석 입력을 기록해둔 파일

파일 이름 형식 : 강의명_분반_mmss_Error.csv

출력 정보 : 잘못된 입력의 출석기록의 이름, 사용자 이메일, 참가 시간, 나간 시간, 기간 그리고 게스트가 입력된 파일

2. 핵심 기능과 그들간의 관계

1) 강의 참여자 CSV 파일 처리

가) 학번과 이름이 담긴 CSV 파일을 처리하여 프로그램은 해당 수업의 수강생 정보를 가지고 있게 된다.

나) 강의 출석 기록 처리, 동명이인 처리 등 수강생의 정보가 필요한 모든 기능은 강의 참여자 CSV 파일 처리가 수행된 뒤에 작동하게 된다.

2) 강의 출석 기록 CSV 파일 처리

가) 특정 요일의 수업에 출석 기록이 담긴 CSV 파일이 처리를 처리하여 학생의 처음 입장 시간, 퇴장 시간, 회의 참여 총 시간, Guest 여부, 이름, 학번 등의 정보를 프로그램이 저장하게 된다.

나) 실질적인 프로그램 구동에서 핵심적인 입력에 대한 처리를 맡았다. 다른 기능들이 실행되기 위해서는 이 기능이 선행되어야 한다. 강의 참가자의 CSV 파일 작업 기능이 선행되어야 이 기능이 정상적으로 작동한다

3) 동명이인 처리

가) 학번이 다른데 이름이 같은 참여자가 두 명 이상 존재할 경우 중복된 대상임을 알 수 있도록 하고 동명이인 참가자일 경우 학번을 기준으로 판별할 수 있도록 한다.

나) 모든 출석부 처리 관련 기능들은 동명이인 처리 기능이 선행되어야 한다.

4) 잘못된 입력에 대한 처리

가) 강의의 수강생에 해당하지 않은 학생의 입력이 들어왔을 경우 이에 대한 처리가 필요하다.

5) 출석 구분 (정상 출석, 지각, 결석)

가) 학생의 입장 퇴장 시간을 기준으로 출석을 구분한다. 정상 출석, 퇴장, 지각을 판단할 수 있다.

6) 날짜별 출석률, 학생별 출석률 처리 후 입력

가) 학생별로 전체 수업에 대해 출석률이 얼마인지 전체 입력된 출석부 수업에서 수업 별 출석률이 얼마인지 확인할 수 있어야 한다. 전체 학생의 처음 입장 시간 마지막 퇴장 시간 그리고 회의 참석 시간을 학생별로 저장할 수 있도록 하고 입장 시간을 기준으로 출석률을 계산한다. 수업 파일 마다 학생의 출석 여부가 저장되어야 하고 “(각각 다른 날의 수업에 대해 한 학생의 총 출석 합) / 총 입력된 수업의 수”가 학생별 출석률이 된다. “(한 수업에 참여한 전체 학생의 출석의 합) / 총 입력된 학생의 수”가 날짜별 출석률이 된다.

3. 요구 사항

1) 문제 해결 범위

가) 동명 이인일 경우 줌 기록은 이름이 우선이기 때문에 이름으로 먼저 구분한 뒤에 이를 처리해준다.

나) 이름을 인식할 수 없는 경우 적절한 처리를 통해 사용자에게 고지해준다.

다) 주어지는 과목 및 접속 기록 파일의 수는 정해져 있지 않다. 프로그램 내부에서 파악하여 처리해 주어야 한다.

라) 수업 시작 시간에서 +10분 까지는 지각, 그 이후는 접속 기록이 있는 경우에도 결석으로 처리한다.

마) 학생별 출석률과 날짜별 출석률을 계산하여 파일에 같이 기록한다.

사) 파일 선택, 수업 시작 시간 설정 출결 상황 확인 등을 GUI를 통해 제공한다.

2) 서비스 개요

자동 출결 처리 서비스는 GUI를 통해 제공되며 입력 받은 CSV파일의 정보를 통해 출결 처리 서비스를 자동으로 제공하는 서비스이다. 출결 처리는 이름을 기반으로 하고 동명이인, 잘못된 형식의 입력을 고려해 주어야 한다. 또한 회의 주최자와 참여자를 구분할 필요가 있다.

3) 시스템 문맥도

이 서비스는 사용자와의 역할을 고려하여 6가지의 유스 케이스로 정의할 수 있다. 파일 입력 - 저장과 같은 파일 입출력을 위한 기능적인 부분과 특정 학생, 날짜별 출석률, 과목 선택과 같은 사용자가 더 편리하게 정보를 확인할 수 있도록 도와주는 view를 바꾸어주는 부분으로 나눌 수 있다.

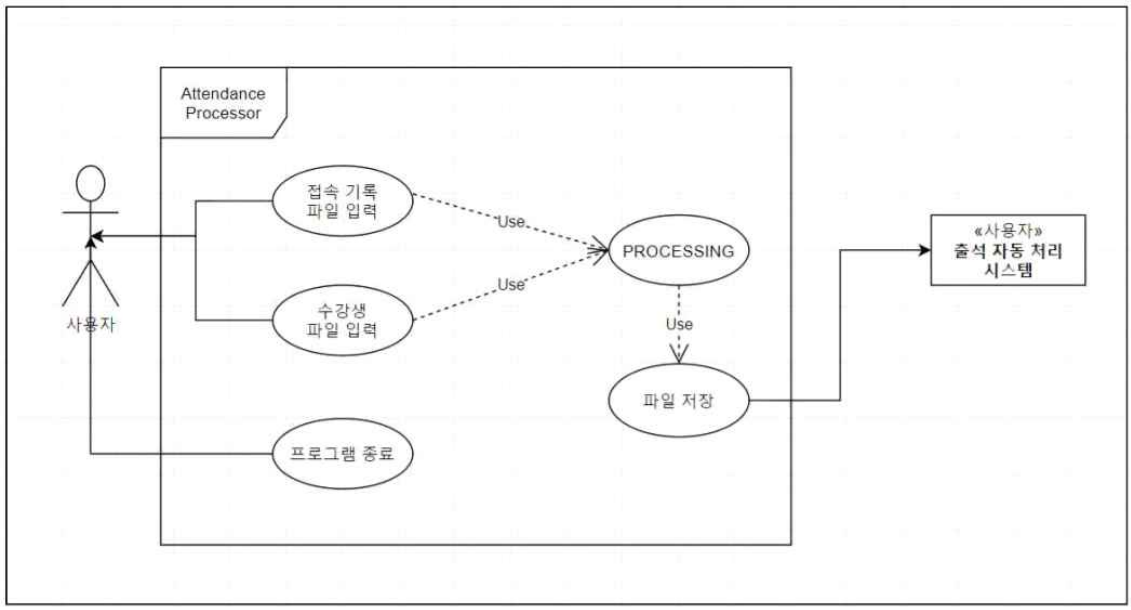


그림 4 Use case diagram

가) 파일 입력 유스 케이스

접속 기록 파일과 수강생 파일을 받는다. 접속 기록을 처리해주고 수업 관리를 위한 해시 테이블을 구성한다.

나) 수업 시작 시간 설정 유스 케이스

사용자가 수업 시작 시간을 설정할 수 있다. 설정된 수업 시작 시간을 기준으로 10분 안 까지는 지각 그 이상은 결석처리 된다.

다) 어떤 수업의 출석 상황 확인

수강생 파일을 Combo box 에 추가해 해당 날짜의 수업을 선택하면 학생들의 출석 상황이 보인다.

바) 파일 저장하기

학생별 출석에 대한 정보와 출석률과 지각률을 입력한 파일을 저장한다.

설계

1. 클래스 다이어그램

AttendanceProcessing
private HashMap<String, List<Lecture>> LectureDB
public Boolean AppendLecture(String dirPath, String dirName) public List<Student> SetStudentAttd(Lecture l) public Boolean AppendStudent(String dirPath, String dirName) public void ProcessingLectureRate(String lecture) public String ConvertShape(int n) public void ReturnToCSV()

Lecture
private String LectureName; private String Date; private Integer LectureSTime; private Integer LectureStdNum; private List<Student> std;
/*getter setter*/ public Lecture GetCopy()

Student
/*이름, 이메일, 게스트 여부, 학번, 시작 시간, 끝시간, 총 접속 시간*/ private Boolean IsDuplicated; private float TotalAttd; private float AttdLate;
/*getter setter*/ public Lecture GetCopy()

FileProcessing
private String FileName; private String FilePath; private List<String> trashList;
FileProcessing(String FilePath, String FileName) private List<List<String>> ReadCSVFile() private Student ProcessingDuplicated(List stdList, Student s1) public Lecture ReadStudentFile() private Integer ParseTime(String s) private Student ProcessTotalAtdTime(Student s, List<String> Attd) private Student ProcessSetEndTime(Student s, List<String> Attd) private Student ProcessSetStartTime(Student s, List<String> Attd) private Integer ParseStartTime(Integer i) private Lecture ProcessLectureStartTime(Lecture l, List<String> Attd) private void SetStudentInfo(Student s, List<String> SList, Lecture l, int i) public Lecture fSetAttdLate(Lecture l) public String dateParse(String dirName) public void SendErrorStudent(String dirName, List<List<String>> list) public Lecture ReadAttendanceFile(String dirName ,Lecture ll)

2. 프로그램 구조

가) Student

Student
private String Name; private String Email; private Boolean isGuest; private Integer Code; private Integer StartTime; private Integer EndTime; private Integer TotalAtdTime; private Boolean IsDuplicated; private float TotalAttd; private float AttdLate;
/*getter setter*/ public Lecture GetCopy()

표 8 학생의 정보를 담은 학생 클래스

1) 학생의 이름, 이메일, 게스트 여부, 학번, 처음 입장 시간, 마지막 퇴장 시간, 총 회의 참석 시간, 동명이인인지 여부, 이 학생의 모든 수업에서 출석률을 더한 값 그리고 현재 해당하는 수업에서의 출석률을 나타낸다.

2) Student를 복사하여 사용할 일이 잦으므로 GetCopy 복사를 위한 함수를 만든다.

나) Lecture

Lecture
private String LectureName; private String Date; private Integer LectureSTime; private Integer LectureStdNum; private List<Student> std;
/*getter setter*/ public Lecture GetCopy()

표 9 수업의 정보를 담은 수업 클래스

1) 현재 입력된 수업들의 정보가 담긴 클래스이다. 입력된 수업의 이름과 수업의 시작 날짜 수업의 시작 시간 그리고 수업에 참여한 학생들의 클래스가 리스트로 저장되게 된다.

다)File Processing

FileProcessing
private String FileName; private String FilePath; private List<String> trashList;
FileProcessing(String FilePath, String FileName) private List<List<String>> ReadCSVFile() private Student ProcessingDuplicated(List stdList, Student s1) public Lecture ReadStudentFile() private Integer ParseTime(String s) private Student ProcessTotalAtdTime(Student s, List<String> Attd) private Student ProcessSetEndTime(Student s, List<String> Attd) private Student ProcessSetStartTime(Student s, List<String> Attd) private Integer ParseStartTime(Integer i) private Lecture ProcessLectureStartTime(Lecture l, List<String> Attd) private void SetStudentInfo(Student s, List<String> SList, Lecture l, int i) public Lecture fSetAttdLate(Lecture l) public String dateParse(String dirName) public void SendErrorStudent(String dirName, List<List<String>> list) public Lecture ReadAttendanceFile(String dirName ,Lecture ll)

표 10 CSV 파일을 읽고 출석 파일 파싱 및 클래스 맴버변수 값 대입

1) 파일이 입력되면 해당 파일의 정보를 읽고 수업 참여 학생들의 정보를 저장, 처리하고 수업에 대한 정보를 저장, 처리한다. 최종적으로 Attendance Processing 에서는 출석률만을 계산하여 저장할 수 있도록 미리 값들을 다 저장해둔다.

라)AttendanceProcessing

AttendanceProcessing
private HashMap<String, List<Lecture>> LectureDB
public Boolean AppendLecture(String dirPath, String dirName) public List<Student> SetStudentAttd(Lecture l) public Boolean AppendStudent(String dirPath, String dirName) public void ProcessingLectureRate(String lecture) public String ConvertShape(int n) public void ReturnToCSV()

표 11 출석을 기록하고 최종 출석률을 처리하는 클래스

1) 날짜별 학생별 출석률을 계산하고 CSV 파일에 저장한다. 각각의 출석에 대한

것은 문자의 형태 (O, X, /) 로 바꾸어주는 함수를 적용한다.

3. 알고리즘 설계

가) 강의 참여자 CSV 파일 처리

```
public Lecture ReadStudentFile()
        중복을 처리 하기 위한 학생 배열 stdList 선언 및 초기화
        문자열 이차원 배열 Attd 선언 및 초기화
        Attd = ReadCSVFile(); // 문자열 그대로를 입력 받음
        새로운 Lecture 객체 생성
        Lecture 클래스에 입력을 위한 학생 배열 std 생성
        for (i = 0; i < 문자 줄 수 만큼; i++) {
            학생 객체 생성
            문자열 리스트의 0번 인덱스를 학생 객체의 학번에 입력
            문자열 리스트의 1번 인덱스를 학생 객체의 이름에 입력
            중복 처리를 위한 배열에 학생 추가
            학생 배열에 학생 객체 추가
        }
        for (j = 0; j < 문자 줄 수 만큼; j++) {
            새로 만든 학생 객체에 학생 인덱스 j 번을 복사하여 초기화
            학생이 중복일 경우를 처리해주는 함수 호출
            중복인지 아닌지 setIsDuplicated를 통해 설정
        }
        수업의 이름은 .csv를 제거한 수업_분반으로 설정
        l 이 가지게 될 학생 배열은 std 로 설정
        return l;
```

나) 동명 이인 처리

```
private Student ProcessingDuplicated(List stdList, Student s1)
        학생 객체 s 에 s1을 복사하여 초기화
        int a = 리스트에 중복되는 이름이 얼마나 많은지 세어 줌
        if (만약 a 가 1 이상이라면) {
            학생 객체의 Duplicated를 true 로 설정
        } else {
            학생 객체의 Duplicated를 false 로 설정
        }
        return s;
```

다) 강의 출석 기록 CSV파일 처리

```
public Lecture ReadAttendanceFile(String dirName ,Lecture ll)
        강의 l 객체에 ll을 복사하여 초기화
        문자열 이차원 배열 Attd 생성
        Attd에 ReadCSVFile 의 결과 대입
        dateParse를 위한 함수를 사용하여 dirName에 있는 문자를 변형
        학생 배열 std 생성
        잘못된 입력을 처리하기 위한 배열 생성
        for (int i =0; i < 문자열 리스트 수 만큼; i++) {
            s 배열에 l의 학생 배열 대입
            string 배열에 string I 번째 인덱스 대입
            for (int j =0; j < 해당 수업에 존재하는 학생 수; j++) {
                s j 번째 인덱스의 학생 t 에 복사
                if(만약 이름이 존재하고 && 중복이 아니라면){
                    학생 정보 입력
                    break;
                else if(만약 학번이 존재한다면){
                     학생 정보 입력
                     break;
                }
                }else if (게스트가 아니라면){
                    수업 시작 시간 설정
                    break;
                }
                if(만약 전체 문자열 배열을 돌 때까지 값을 대입하지 못
했다면)
                    에러리스트에 추가
                    에러리스트와 학생 정보 전송
                    break;
                }
        String LectureName = FileName.replace(".csv", "");
        l.SetLectureName(LectureName);
        총 학생들 전체의 출석률 처리
        return l;
    }
```

마) 출석 구분 (정상 출석, 지각, 결석)

```
public Lecture fSetAttdLate(Lecture l)
        학생 배열에 l 의 학생 배열을 넣는다.
        for (int i = 0; i < 학생 수만큼; i++) {
            if (학생의 입장 시간이 null 이 아니라면) {
                if (수업 시작 후 십분 초과) {
                    s.get(i).SetAttdLate(0);
                } else if (수업 시작보다 초과했으나 십분 안에 도착) {
                    s.get(i).SetAttdLate(100 * 3 / 4);
                } else {
                    s.get(i).SetAttdLate(100);
                }
            }
        }
        return l;
    }
```

바)날짜별 출석률, 학생별 출석률 처리 후 입력

```
public void ReturnToCSV()
float a = 0;
float a1 = 0;
int Lecture;
keys 에 현재 LectureDB의 키 값을 가진다.
while (키값이 없어질 때 까지) {
학번 입력
, 입력
이름 입력
, 입력
수업 배열 l 에 수업을 키 값으로 찾은 배열 등록
for (int i = 1; i < 수업 크기 만큼; i++) {
날짜 입력
, 입력
}
줄띄움 입력
```

```
수강생 정보 받기
for(int Student = 0; Student < 수강생 수만큼; Student++){
학번 입력
쉼표 입력
이름 입력
쉼표 입력
for(Lecture = 1; Lecture < 전체 수업 크기; Lecture++){
한 학생의 출석 점수 합하기
한 학생의 출석 점수 문자 형태로 바꾸어 입력
쉼표 입력
}
전체 점수 합에서 수업 크기로 나누어 평균 구하기
줄 띄움
}
, 입력
, 입력
for(int Lec = 1; Lec < l.size(); Lec++){
for(int Student = 0; Student < standard.size(); Student++){
해당 수업의 출석 점수 합
}
학생별 출석 전체 합의 평균 입력
, 입력
}
sum = sum / (l.size() - 1); // 전체 합의 평균 입력
합 입력
줄 띄움 입력
CSV 파일 쓰기
}
```

구현

1. 클래스 코드
   가) Student

```
public class Student {
    private String Name;
    private String Email;
    private Boolean isGuest;
    private Integer Code;
    private Integer StartTime;
    private Integer EndTime;
    private Integer TotalAtdTime;
    private Boolean IsDuplicated;
    private float TotalAttd;
    private float AttdLate;
    public void SetAttdLate(float AttdLate) {
        this.AttdLate = AttdLate;
    }

    public float GetAttdLate() {
        return this.AttdLate;
    }
    /*getter setter GetAttdLate SetAttdLate와 같은 형식으로 */

    public Student GetCopy() {
        Student s = new Student();
        s.SetName(Name);
        s.SetEmail(Email);
        s.SetIsGuest(isGuest);
        s.SetCode(Code);
        s.SetStartTime(StartTime);
        s.SetEndTime(EndTime);
        s.SetTotalAtdTime(TotalAtdTime);
        s.SetIsDuplicated(IsDuplicated);
        s.SetTotalAttd(TotalAttd);
        return s;
    }
}
```

나) Lecture

```
public class Lecture{
    private String LectureName;
    private String Date;
    private Integer LectureSTime;
    private Integer LectureStdNum;
    private List<Student> std;
    public void SetDate(String Date){
        this.Date = Date;
    }
    public String GetDate(){
        return this.Date;
    }
    public void SetLectureName(String LectureName){
        this.LectureName = LectureName;
    }
    /*getter setter */
    public Lecture GetCopy() {
        Lecture lecture = new Lecture();
        lecture.SetLectureName(LectureName);
        lecture.SetLectureSTime(LectureSTime);
        lecture.SetLectureStdNum(LectureStdNum);
        lecture.SetStudents(new ArrayList<Student>());
        for (Student student : std)
            lecture.GetStudents().add(student.GetCopy());
        return lecture;
    }

}
```

다) FileProcessing

```
public class FileProcessing {
    private String FileName;
    private String FilePath;
    private List<String> trashList;

    FileProcessing(String FilePath, String FileName);
    private List<List<String>> ReadCSVFile();
    private Student ProcessingDuplicated(List stdList, Student s1);
    public Lecture ReadStudentFile();
    private Integer ParseTime(String s);
    private Student ProcessTotalAtdTime(Student s, List<String> Attd);
    private Student ProcessSetEndTime(Student s, List<String> Attd);
    private Student ProcessSetStartTime(Student s, List<String> Attd);
    private Integer ParseStartTime(Integer i);
    private Lecture ProcessLectureStartTime(Lecture l, List<String> Attd);
    private void SetStudentInfo(Student s, List<String> SList, Lecture l, int i);
    public Lecture fSetAttdLate(Lecture l);
    public String dateParse(String dirName);
    public void SendErrorStudent(String dirName, List<List<String>> list);
    public Lecture ReadAttendanceFile(String dirName ,Lecture ll);
```

라) AttendanceProcessing

```
public class AttendanceProcessing {
    private HashMap<String, List<Lecture>> LectureDB =
    new HashMap<String, List<Lecture>>();

    public List<Lecture> GetLecture(String Lecture);
    public void SetStartTime(String className,String time)
    public Boolean AppendLecture(String dirPath, String dirName);
    public Boolean AppendStudent(String dirPath, String dirName);
    public void ProcessingLectureRate(String lecture);
    public String ConvertShape(int n);
    public void ReturnToCSV();
```

2. 핵심 알고리즘

가) 결과 CSV파일 입력

```
 public void ReturnToCSV() {
        /*초기화*/
        while (keys.hasNext()) {
            String key = keys.next();
            try (PrintWriter writer = new PrintWriter(new
File(key+".csv"))) {
                StringBuilder sb = new StringBuilder();
                sb.append("학번");
                sb.append(",");
                sb.append("이름");
                sb.append(",");
                List<Lecture> l = LectureDB.get(key);
                for (int i = 1; i < l.size(); i++) {
                    sb.append(l.get(i).GetDate());
                    sb.append(",");
                }
                sb.append("\n");
                List<Student> standard = l.get(0).GetStudents();
                for(int Student = 0; Student < standard.size();
Student++){
                    a = 0;
                    Lecture = 1;

sb.append(l.get(Lecture).GetStudents().get(Student).GetCode());
                    sb.append(",");

sb.append(l.get(Lecture).GetStudents().get(Student).GetName());
                    sb.append(",");
                    for(Lecture = 1; Lecture < l.size(); Lecture++){
                        a =
 a + (l.get(Lecture).GetStudent(Student).GetAttdLate());

sb.append(ConvertShape((int)l.get(Lecture).GetStudent(Student).GetAttd
Late()));
                        sb.append(",");
                    }
                    sb.append(a/(l.size()-1)); // 한 학생의 수업별 출석
점수의                          평균을 입력한다.
```

```
                    sb.append("\n");
                }
                sb.append(",");
                sb.append(",");
                float sum = 0;
                for(int Lec = 1; Lec < l.size(); Lec++){
                    a1 = 0;
                    for(int Student = 0; Student < standard.size();
 Student++){
                        a1 =
 a1 + (l.get(Lec).GetStudent(Student).GetAttdLate()); //학생들의 출석
점수를 더한다
                    }
                    a1 = a1 / (standard.size()); // 전체 학생 수로 나눈
다
                    sum += a1;
                    sb.append(a1);
                    sb.append(",");
                }
                sum = sum / (l.size() - 1);
                sb.append(sum); // 전체 학생들의 수업별 출석의 평균
                sb.append("\n");
                writer.write(sb.toString());
                writer.close();
            }catch(FileNotFoundException e){
                System.out.println(e.getMessage());
            }
        }
    }
```

LectureDB 에 들어있는 값을 전부 다 한번에 처리해주는 클래스이다. LectureDB 에 포함 되어있는 key 들 마다 수업 배열을 전부다 처리하여 출석률과 지각률이 반영된 csv 파일이 최종적으로 출력되게 된다.

나) 동명이인 처리

```
public Lecture ReadAttendanceFile(String dirName ,Lecture ll) {
        /*변수에 값들 대입한 부분*/
        for (int i =0; i < Attd.size(); i++) {
            List<Student> s = l.GetStudents();
            List<String> SList = Attd.get(i);
            for (int j =0; j < s.size(); j++) {
```

```
                    Student t = s.get(j).GetCopy();
                    if(SList.get(0).contains(t.GetName()) &&
 !t.GetIsDuplicated()){ // 이름을 포함 한다면
                        SetStudentInfo(t, SList, l, j);
                        break;
                    }else if(SList.get(0).contains(t.GetIsDuplicated() &&
 Integer.toString(t.GetCode()))){
                        SetStudentInfo(t, SList, l, j); //동명이인임으로 학번
구분
                        break;
                    }else if (SList.get(5).contains("아니요")){// Guest 가
아니면
                        l = ProcessLectureStartTime(l, SList);// 시작 시간
설정
                        break;
                    }
                    if(j == s.size() - 1){
                        errorList.add(Attd.get(i)); // 잘못된 입력 처리
                        SendErrorStudent(dirName, errorList);
                        break;
                    }
                }
            }
            String LectureName = FileName.replace(".csv", "");
            l.SetLectureName(LectureName);
            l = fSetAttdLate(l);
            return l;
        }
```

참여자 파일에서 이름이 같은 사람이 한명이상 존재한다면 그 이름을 가진 전원을 동명이인으로 처리하여 SetIsDuplicated(true)를 해주었다. 따라서 종명이인이 아니고 이름이 같다면 학생 정보 입력 동명이인이고 학번이 같다면 정보 입력 절차를 거쳐 처리해 주었다. 이 방법을 쓰면 많은 인원의 동명이인이 입력되어도 처리할 수 있다.

다) 출석 처리

```
public List<Student> SetStudentAttd(Lecture l){
        Lecture lc = l.GetCopy();
        List<Student> s = lc.GetStudents();
        for (int i = 0; i < s.size(); i++) {
            if (s.get(i).GetStartTime() != null) {
```

```
                if (s.get(i).GetStartTime() > l.GetLectureSTime() +
10 || s.get(i).GetName() == null) { // 수업시작 후 10분 초과
                    s.get(i).SetAttdLate(0);
                } else if (s.get(i).GetStartTime() <=
l.GetLectureSTime() + 10 && s.get(i).GetStartTime() >
l.GetLectureSTime()) { // 지각에 대한 처리
                    s.get(i).SetAttdLate(100 * 3 / 4);
                } else {
                    s.get(i).SetAttdLate(100); // 정상 출석
                }
            }
        }
        return s;
    }
```

지각 4번에 결석 1번으로 처리됨으로 점수를 정상 출석 100점 지각 75점 결석 0점을 주었다. 이를 기반으로 처리하여 만약 정상 회의 주최자가 들어온 시간에서 그 이후 정각 보다 이른 시간 참여하였다면 출석 그 정각보다 늦었으나 10분 안에 들어왔다면 지각 그 외에는 결석으로 처리한다.

라) 기준 시간 설정

```
private Integer ParseStartTime(Integer i) {
        if (i % 100 != 0) {
            i = i + 100;
            i = i / 100;
            i = i * 100;
        }
        return i; // 정각을 기준으로 바꾸어줌
    }

    private Lecture ProcessLectureStartTime(Lecture l, List<String>
Attd) {
        Integer i = ParseStartTime(ParseTime(Attd.get(2)));
        if (l.GetLectureSTime() == null) {
            l.SetLectureSTime(i);
        } else if (l.GetLectureSTime() > ParseTime(Attd.get(2))) {
            l.SetLectureSTime(i);
        }
        return l;
    }
```

수업마다 기준 시간이라는 멤버 변수가 있다 만약 값이 입력이 되었다면 기준 시

간을 변경하여 반영하여 준다.

결과 분석 및 검토

1. 시도할 의미가 있는 기술 요소

가) 이진 탐색

이진 탐색이란 정렬되어 있는 배열에서 특정한 값을 찾아내는 알고리즘이다. 배열의 중간에 있는 임의의 값을 선택하여 찾고자 하는 값과 비교 하고 찾고자 하는 값보다 작으면 중간 값들을 기준으로 좌측 데이터들을 다시 탐색하는 식이다.

이진 탐색은 순차적으로 정렬되어있는 데이터에서 O(logN) 의 시간 복잡도가 된다. 출석부 처리 문제와 같이 순차적인 데이터가 입력된다면 이진 탐색을 사용하여 탐색하는 것이 훨씬 효율적일 것이다.

나) 학생의 추가와 삭제

데이터 베이스같은 경우 요소의 삭제와 추가가 잘 고려되어있다 현재 프로그램같은 경우에도 사용자가 학생의 정보를 수정하고 추가하고 삭제할 수 있는 기능이 지원된다면 훨씬 더 편리한 프로그램이 될 것 같다.

2. 코드 평가

과한 자료구조의 사용으로 인해 코드의 구조가 전반적으로 복잡해 보였다. 그리고 클래스 하나 하나당 의존도가 너무 높아 하나의 함수를 수정하게 된다면 다른 클래스에 속하는 함수까지 수정이 필요하였다. 다음부터는 이를 매우 신경 써야겠다. 또한 지금은 모델이 굉장히 간단하게 이루어져 뷰 부분에 f1을 모델로 사용하는 식이었다. 모델 클래스도 분리하여 사용하는 편이 훨씬 후를 생각해서라도 좋을 것이다. 새로운 기능을 추가하고 기능을 삭제할 수도 있을 것이라는 것을 항상 염두해두고 설계를 하는 것이 좋을 것 같다. 아직은 많이 부족한 코드라는 생각이 든다.

개발 후기

1. 후기

탐색에 있어서 상수시간을 갖는 해시 함수를 사용한다면 훨씬 더 빠른 시간 복잡도로 접근할 수 있을거라 생각했지만 현재 가지고 있는 데이터는 순차적인 데이터이며 해시 테이블과 현재 데이터는 맞지 않는다는 것을 조금 뒤늦게 깨달아 버렸다. 그래서 구조적으로 조금 복잡해진 면, 과한 자료구조를 사용한 점이 아쉬웠다. 하지만 알아보던 중 해시 충돌에 대해 조금 더 깊이 알게되었다. DB가 구성되는 것도 당연히 해시 테이블로 구성이 되어있을거라 생각했지만 그렇지 않았다. 충돌 문제로 인하여 이진 완전 트리로 구성되어 탐색하는 식이었다.

다음에 이 자료구조를 사용할만한 적합한 문제를 또 만나게 된다면 적극적으로 사용해볼 생각이다. 현재 데이터는 시간 복잡도의 차이를 느낄 만큼 많지 않아 코드 내부의 자료구조보다는 다른 측면에서 효율성을 높이는 방법이 더 효과적일 수도 있겠다는 생각이 들었다. 예를 들면 불필요한 반복문들을 최대한 없애는 방법등을 항상 염두해 두어야 겠다.

이번 문제는 전반적인 설계와 문제 해결이 중요했던 것 같다. 설계를 할 때 항상 검증을 하는 것이 중요하다는 것을 깨달았다. 작은 데이터를 임의로 만들어 알고리즘을 검증해 나갔다면 코드를 짜는 속도가 훨씬 더 빨라질 것 같다.