

소프트웨어 프로젝트 5

20131790

권소현

목차

1. 프로젝트 목표 (p.3)
2. 프로젝트 내용 (p.3)
3. 설계 노트 (p.4~10)
4. Test 결과 (p.11~14)
5. 소스 프로그램 (p.15~36)
6. 자체 평가표 (p.37)

1. 프로젝트 목표

Project4의 GUI에 event-handler를 구현

2. 프로젝트 내용

- 프로그램이 시작하면, Project2에서 구현했던 내용대로 예약 파일을 읽어 와서 내용을 예약 리스트에 표시
- 빈칸에 예약 내용을 직접 입력해서 "예약" 버튼을 클릭하면 예약 추가 (예약 리스트 추가, 예약 파일 갱신)
- 예약 리스트에서 하나를 선택하고 "취소" 버튼을 클릭하면 예약 취소 (예약 리스트 제거, 예약 파일 갱신)

3. 설계 노트

1) 추가된 ReservationGUI 클래스

- import

```
import java.io.*;
```

```
import java.util.ArrayList;
```

- 변수

타입	변수명	역할
JComboBox<String>	roomNum	강의실의 JComboBox
JComboBox<String>	reservDay	예약 요일의 JComboBox
JComboBox<String>	reservTime	예약 시간의 JComboBox
JComboBox<String>	situation	예약 상황의 JComboBox
String[]	room	강의실은 Room514, Room515, Room516만 존재한다고 가정
String[]	day	reservDay의 내용
String[]	time	reservTime의 내용
TextField	reservName	예약자 이름
TextField	reservMemo	예약 메모
RoomReservation	dataFile	파일에 저장된 예약 내용을 불러온다.
ArrayList<ReservationRecord>	reservList	예약 리스트를 저장

- ReservationGUI 추가 내용

- 1) reservList에 저장한 데이터파일의 예약 내용을 JTable에 표시하기 위해 String배열을 생성하여 저장후 .addRow(String)을 이용하여 JTable에 추가

```
// RoomReservation(datafile)에 저장되어있는 예약사항을
JTable에 표시
for(int i = 0; i<reservList.size(); i++){
    String[] input = new String[5];
    for(int j=0; j<input.length; j++){
        input[j] = null;
    }
    input[0] = reservList.get(i).room_num;
    input[1] = reservList.get(i).day;
    input[2] = String.valueOf(reservList.get(i).time +
1);

    input[3] = reservList.get(i).name;
    input[4] = reservList.get(i).memo;
    model.addRow(input);
}
```

- ReservListener Class 변경

- 1) 새로운 예약 사항을 저장하기위해 ReservListener 생성자를 다음과 같이 변경

```
ReservListener(JTable table, JComboBox roomNum,
JComboBox reservDay,
JComboBox reservTime, JTextField
reservName, JTextField reservMemo){
    this.table = table;
    this.roomNum = roomNum;
    this.reservDay = reservDay;
    this.reservTime = reservTime;
    this.reservName = reservName;
    this.reservMemo = reservMemo;
}
```

- 2) actionPerformed 메소드

1. 예약사항을 String배열에 저장, ReservationRecord로 생성

```
// 예약사항을 String 배열에 저장
String[] input = new String [5];
```

```

        input[0] = (String)roomNum.getSelectedItemAt();
        input[1] =
        (String)reservDay.getSelectedItemAt();
        input[2] =
        (String)reservTime.getSelectedItemAt();
        input[3] = reservName.getText();
        input[4] = reservMemo.getText();

        // ReservationRecord에 저장
        ReservationRecord newReservation = new
        ReservationRecord(input);

```

2. 이미 예약 리스트에 중복이 있는지 확인

```

// 예약 리스트에 추가 , 같은 것이 있을 경우 추가 x
        boolean conflict = false;
        for(int i = 0; i<table.getRowCount(); i++){
            if(table.getValueAt(i,
0).equals(input[0]) && table.getValueAt(i,
1).equals(input[1]) && table.getValueAt(i,
2).equals(input[2])){

                System.out.println("Your new reservation conflicted");
                conflict = true;
            }
        }

```

3. 중복이 없을 경우 jTable에 추가

```

// 같은것이 없으면 jTable 예약 리스트에 추가 & 예약파일 갱신
        if(!conflict){

            // jTable 예약 리스트에 추가
            DefaultTableModel model =
            (DefaultTableModel) table.getModel();
            model.addRow(input);
        }

```

4. 예약 파일을 갱신

```

BufferedWriter bw;

        try {
            bw = new BufferedWriter(new
            FileWriter("roomreserve-norm.data",true));
            PrintWriter pw = new

```

```

PrintWriter(bw,true);
        String line =
input[0]+":"+input[1]+":"+input[2]+":"+input[3]+":"+input[4]
;
        pw.println();
        pw.write(line);
        pw.flush();
        pw.close();
    }
    catch (IOException e1) {
        // TODO Auto-generated catch
        e1.printStackTrace();
    }
}

```

- CancelListener Class 변경

1) JTable에서 제거 해야하므로 생성자를 다음과 같이 변경

```

CancelListener(JTable table){
    this.table = table;
}

```

2) actionPerformed 메소드

1. datafile에서 선택한 예약 사항의 line을 지우기 위해 제거할 line의 위치를 찾는다.

```

int row = table.getSelectedRow();
File file = new File("roomreserve-
norm.data");
String dummy = "";

try{
    BufferedReader search = new
BufferedReader(new InputStreamReader(new
FileInputStream(file)));

    String searchLine = search.readLine();
    int position = -1;

    // 제거할 line의 위치 찾기
    for(int i = 0; searchLine != null;
i++){

```

```

        if
(!searchLine.startsWith("//") && searchLine.length()!=0){
            String[] tokens =
searchLine.split(":");
            String[] data = new
String[5];
            if(tokens.length == 5){
                data[0] =
tokens[0].trim();
                data[1] =
tokens[1].trim().toLowerCase();
                data[2] =
tokens[2].trim();
                data[3] =
tokens[3].trim();
                data[4] =
tokens[4].trim();
            }
            else if(tokens.length
== 4){
                data[0] =
tokens[0].trim();
                data[1] =
tokens[1].trim().toLowerCase();
                data[2] =
tokens[2].trim();
                data[3] =
tokens[3].trim();
            }

            if((data[0].equals((String)(table.getValueAt(row, 0))))
&&
(data[1].equals((String)(table.getValueAt(row,
1).toString().toLowerCase()))
&&
(data[2].equals((String)(table.getValueAt(row, 2))))){
                position = i;
            }
        }
        searchLine = search.readLine();
    }
    search.close();

```

2. 일치하는 예약의 line을 찾지 못했을 경우 오류를 표시하고 제거할 예약 사항을 찾았을 경우 그 line을 제외한 내용을 저장한다.

```

// 일치하는 예약 사항의 line을 찾지 못했을 경우
        if(position == -1){
            System.out.println("There is no

```



```

statement");
    }
    else{
        // 제거할 예약사항을 제외하고
        내용을 저장한다.
        String line;
        BufferedReader br = new
        BufferedReader(new InputStreamReader(new
        FileInputStream(file)));

        for(int i=0; i<position; i++){
            line = br.readLine();
            dummy += (line+

"\r\n");
        }
        String delData = br.readLine();
        while((line = br.readLine()) !=
        null){
            dummy += (line+

"\r\n");
        }

        // 원래 datafile에 내용을
        덮어씌운다.
        FileWriter fw = new
        FileWriter("roomreserve-norm.data");
        fw.write(dummy);

        fw.close();
        br.close();
    }
}

```

3. JTable에서 선택한 예약을 제거

```

if(row == -1){
    return;
}
DefaultTableModel model = (DefaultTableModel)
table.getModel();
model.removeRow(row);

```

- 시행착오, 내용 및 해결 방안

1. RoomReservation에서의 예약 내용을 불러오기 위해서는 예약 사항만 저장한 ArrayList가 필요했고, 그것을 얻을 수 있는 메소드도 필요했다.

따라서 RoomReservation.java에 ArrayList<ReservationRecord> realRecords를 생성하고 Get_Real_Records()라는 메소드도 만들었다.

2. 프로젝트 리포트에 나온 것처럼 예약을 추가할 경우 리스트를 갱신 시키기 위해 새로운 file을 만든 후 대체하려고 했다.

하지만 PrintWriter를 생성 할 경우 new PrintWriter(buff, true);를 할 경우 원래 파일에 덮어쓰지 않고 이어 쓸 수 있어서 이 방법으로 더 편하게 했다.

3. 예약을 제거 할 경우 JTable에서 예약을 제거한 후에 datafile을 갱신 시키려고 했다. 하지만 자꾸 선택한 것의 밑의 것이 삭제되었다.

한 줄 씩 내용과 위치를 표시해보니 이미 JTable에서 지워졌기 때문에 다음 것의 내용이 JTable에서 얻어졌던 것이었다. 따라서 순서를 바꿔 datafile의 내용을 갱신 시킨 후 JTable에서 예약을 제거했다.

4. Test 결과

- 처음 실행 시 GUI 화면

The screenshot shows a GUI window titled "RoomReservation_20131790_권소현". It features a table with columns: 호실 (Room), 요일 (Day), 시간 (Time), 예약자 (Reserver), and 메모 (Memo). The table contains reservation data for Room514 and Room515. To the right of the table is a vertical bar with buttons labeled "예약" (Reserve) and "취소" (Cancel). At the bottom, there is a section for "예약가능시간" (Available Time) and a dropdown menu currently showing "Room514", with a button labeled "방별예약상황" (Room Reservation Status).

호실	요일	시간	예약자	메모
Room514	sun	1		
예약 상황				
Room514	tue	1	Park	Test
Room514	wed	2	Kim	
Room514	wed	3	Kim	
Room514	thr	7	Java	
Room515	sat	2	Kim	New Room
Room515	fri	2	Lee	
Room515	sat	1	Kim	

- 처음 datafile의 내용

The screenshot shows a text editor window titled "roomreserve-norm - 메모장". The content of the file is as follows:

```
// Course Schedule
// Format :: CourseName:Credit:StartDay:StartHour:Hour

Room514:Tue:1:Park:Test
//
Room514:Wed:2:Kim:
Room514:Wed:3:Kim:
//
Room514:Thr:7: Java
//
Room515:Sat:2:Kim: New Room
Room515:Fri:2:Lee:
Room515:Sat:1:Kim:
```

- 호실: Room516, 요일: tue, 시간: 3, 예약자: sohyun, 메모: test1으로
예약 화면 & 데이터파일 화면

RoomReservation_20131790_권소현

호실	요일	시간	예약자	메모
Room516	tue	3	sohyun	test1
예약 상황				
Room514	tue	1	Park	Test
Room514	wed	2	Kim	
Room514	wed	3	Kim	
Room514	thr	7	Java	
Room515	sat	2	Kim	New Room
Room515	fri	2	Lee	
Room515	sat	1	Kim	
Room516	tue	3	sohyun	test1

예약

취소

예약가능시간 Room514 방별예약상황

roomreserve-norm - 메모장

```
// Course Schedule
// Format:: CourseName:Credit:StartDay:StartHour:Hour
Room514:Tue:1:Park:Test
//
Room514:Wed:2:Kim:
Room514:Wed:3:Kim:
//
Room514:Thr:7: Java
//
Room515:Sat:2:Kim: New Room
Room515:Fri:2:Lee:
Room515:Sat:1:Kim:
Room516:tue:3:sohyun:test1
```

- 호실: Room515, 요일: fri, 시간: 2, 예약자: sohyun, 메모: test2로 예약 시 화면 & 오류 표시 콘솔창

RoomReservation_20131790_권소현

호실	요일	시간	예약자	메모
Room515	fri	2	sohyun	test2

예약 상황

Room514	tue	1	Park	Test
Room514	wed	2	Kim	
Room514	wed	3	Kim	
Room514	thr	7	Java	
Room515	sat	2	Kim	New Room
Room515	fri	2	Lee	
Room515	sat	1	Kim	
Room516	tue	3	sohyun	test1

예약

취소

예약가능시간

Room514 방별예약상황

```

Test (1) [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (2016. 5.
Room Name :: Room514
  Sun    Mon    Tue    Wed    Thr    Fri    Sat
-----
1                Park
2                Kim
3                Kim
4
5
6
7                Java
8
Room Name :: Room515
  Sun    Mon    Tue    Wed    Thr    Fri    Sat
-----
1                Kim
2                Lee    Kim
3
4
5
6
7
8
Your new reservation conflicted
  
```

- 두번째 줄의 호실: Room514, 요일: wed, 시간: 2, 예약자: Kim을 선택 후 취소를 눌렀을 경우 GUI화면과 datafile 화면

The GUI window displays a reservation table with the following data:

호실	요일	시간	예약자	메모
Room515	fri	2	sohyun	test2
예약 상황				
Room514	tue	1	Park	Test
Room514	wed	3	Kim	
Room514	thr	7	Java	
Room515	sat	2	Kim	New Room
Room515	fri	2	Lee	
Room515	sat	1	Kim	
Room516	tue	3	sohyun	test1

On the right side of the table, there is a vertical bar with a button labeled '예약' (Reservation) and a button labeled '취소' (Cancel). Below the table, there is a section with a button labeled '예약가능시간' (Reservation Available Time), a dropdown menu currently showing 'Room514', and a button labeled '방별예약상황' (Room-by-room Reservation Status).

```
// Course Schedule
// Format:: CourseName:Credit:StartDay:StartHour:Hour
Room514:Tue:1:Park:Test
//
Room514:Wed:3:Kim:
//
Room514:Thr:7: Java
//
Room515:Sat:2:Kim: New Room
Room515:Fri:2:Lee:
Room515:Sat:1:Kim:
Room516:tue:3:sohyun:test1
```

5. 소스 프로그램

- Test.java

```
package report6;

import javax.swing.*;

public class Test {

    public static void main(String[] args) {
        RoomReservation res = new RoomReservation();
        res.showReservation("roomreserve-norm.data");
        ReservationGUI gui = new ReservationGUI();
        gui.setTitle("RoomReservation_20131790_권소현");
        gui.setSize(700, 500);
        gui.setLocationRelativeTo(null);
        gui.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        gui.setVisible(true);
    }
}
```

- RoomReservation.java

```
package report6;

import java.io.File;

import java.util.Scanner;

import java.util.ArrayList;

public class RoomReservation {

    String[][][] rooms = new String[3][8][7]; // 최대 3개의 강의실과 입력받을 8x7칸의
    강의실 내용을 저장할 String rooms를 만든다.

    String[] room_name = new String[3]; // 최대 3개의 강의실이름을 입력받을 String
    room_name을 만든다.

    int number_of_rooms = 0; // 현재 강의실의 갯수를 알려줄 int형
    number_of_rooms를 만든다.

    ArrayList <ReservationRecord> records = new ArrayList<ReservationRecord>();
```

```

        public static ArrayList <ReservationRecord> realRecords = new
        ArrayList<ReservationRecord>();

```

```

        public void showReservation(String reservationFileName){ // 예약 사항을 보여줄
        메소드를 만든다.

```

```

            // 파일의 내용을 불러온다.

```

```

            Scanner input = null;

```

```

            File file = new File(reservationFileName);

```

```

            // 파싱된 예약사항(ReservationRecord)을 저장해줄 ArrayList를 만든다.

```

```

            String[] data = new String[5]; // 파싱할 정보를 저장해줄 String array

```

```

            try{

```

```

                input = new Scanner(file);

```

```

            }

```

```

            catch(Exception e){

```

```

                // 파일을 읽지 못했을 경우

```

```

                System.out.println("Unknwon File");

```

```

            }

```

```

            while(input.hasNext()){

```

```

                String line = input.nextLine();

```

```

                // 예약 사항 파싱

```

```

                if(!line.startsWith("/") && line.length()!=0){ // 주석처리로 된 문
                장이나 빈줄을 제외시킨다.

```

```

                    String[] tokens = line.split(":"); // 구분자 ':'를 기준으로
                문장을 나눈다.

```

```

                    if(tokens[0] != null && tokens[1] != null &&
                tokens[2] != null && tokens[3] != null ){

```

```

                        // 내용이 비어있는 곳이 있는 경우를 제외시
                킨다. (입력 형식의 오류)

```


	if(tokens.length==5){
	// 강의실 이름, 요일, 시간, 예약자명,
메모가 모두 있는 경우	
	data[0] = tokens[0].trim();
	data[1] = tokens[1].trim();
	data[2] = tokens[2].trim();
	data[3] = tokens[3].trim();
	data[4] = tokens[4].trim();
	// record 생성 후 arraylist인 records
에 저장	
	ReservationRecord record = new
ReservationRecord(data);	
	records.add(record);
	}
	else if(tokens.length == 4){
	// 강의실 이름, 요일, 시간, 예약자명
이 있는 경우 (메모 입력 X)	
	data[0] = tokens[0].trim();
	data[1] = tokens[1].trim();
	data[2] = tokens[2].trim();
	data[3] = tokens[3].trim();
	data[4] = null;
	// record 생성 후 arraylist인 records
에 저장	
	ReservationRecord record = new
ReservationRecord(data);	
	records.add(record);
	}

```

else{
    //비어있는 내용이 있을 경우 오류
    표시 (입력 형식이 잘못된 경우)

    System.out.println("Irregular
reservation line");

}

}

}

}

for(int i = 0; i < 3; i++){
    for(int j = 0; j < 8; j++){
        for(int k = 0; k < 7; k++){
            this.rooms[i][j][k]=null;
        }
    }
}

for(int i = 0; i < 3; i++){
    this.room_name[i] = null;
}

this.number_of_rooms = 0;

// records에 있는 record가 요일check, 시간check가 정상일때 , 시간 중
복X, 강의실 3개 초과X일때 저장

for(int i = 0; i < records.size(); i++){
    // 요일의 위치를 찾아 저장
    int day_index = records.get(i).get_Day_Index();

    // 강의실이 저장되어있는 위치를 찾아 저장
    if(records.get(i).day_Check()){ // 요일이 정상적으로 입력된 경우

```

```

        if(records.get(i).time_Check()){ // 시간이 정상적으로 입력된 경우

            int room_index =
get_Room_Index(records.get(i));

            if(room_index != -1){ // room이 존재하거나 존재하지 않을 경우 3개를 넘지 않을 때

                if(rooms[room_index][records.get(i).time][day_index] == null){ // 예약이 비어있으면 내용을 저장한다.

                    rooms[room_index][records.get(i).time][day_index] = records.get(i).name;

                    // 실제 저장되는 예약 사항을 저장

                    realRecords.add(records.get(i));

                }

                else{

                    // 예약이 이미 되어있는 경우 충돌 요일 & 시간을 표시한다.

                    System.out.println("Conflict
hour -- "+records.get(i).day + " " + (records.get(i).time+1));

                }

            }

        }

    }

}

// 강의실 시간표를 출력해준다.

```

```

        Display(rooms, room_name, number_of_rooms);
    }

    // 강의실의 위치를 구하는 메소드

    public int get_Room_Index(ReservationRecord record){

        if( number_of_rooms == 0){ // 저장된 강의실이 아예 없을 때

            room_name[number_of_rooms] = record.room_num;

            number_of_rooms++; // 방의 갯수가 1개 증가

            return (number_of_rooms-1); // 이 강의실은 room_name[0]에
저장된 강의실

        }

        else{ // 저장된 강의실이 있을 경우

            for(int i = 0; i<number_of_rooms; i++){

                // 저장된 강의실이 있고, 자신의 강의실이 있다면 저장되어있
는 강의실의 위치를 반환

                if(record.room_num.equals(room_name[i])){

                    return i;

                }

            }

            // 나의 강의실이 저장되어있지 않고, 강의실의 갯수가 2개이하
라면

            if(number_of_rooms<3){

                room_name[number_of_rooms] = record.room_num;

                number_of_rooms++; // 방의 갯수가 증가

                return (number_of_rooms-1);

            }

            // 나의 강의실이 저장되어있지 않고, 강의실의 갯수가
3개라면 강의실이 초과되었다고 표시

```

```

else if( number_of_rooms>=3){

    System.out.println("The number of rooms is exceeded");

    return -1;

}

return -1;

}

}

// 강의실 시간표를 출력해주는 메소드

public void Display(String [][] rooms, String[] room_name, int
number_of_rooms ){

    String[] days = {"Sun","Mon","Tue","Wed","Thr","Fri","Sat"};

    for( int i = 0; i<number_of_rooms ; i++){

        // 강의실 이름을 출력

        System.out.println("Room Name :: "+ room_name[i]);

        // 요일 출력

        for( int j = 0; j< days.length; j++){

            System.out.print("\t" + days[j]);

        }

        System.out.println();

        System.out.println("-----");

        -----");

        // 입력된 내용이 없을경우 '\t'을 입력, 있을 경우 이름 +'\t'

        입력

        for(int j = 0; j< 8; j++){

            System.out.print(j+1+"\t");

            for(int k = 0; k<7; k++){

```

```

        if(rooms[i][j][k]==null){

            System.out.print("Wt");

        }

        else{

            System.out.print(rooms[i][j][k]+"Wt");

        }

    }

    // 한시간을 출력하면 다음시간으로 띄어준다.

    System.out.println("");

}

}

}

// 예약사항을 저장한 ArrayList를 전달

public ArrayList<ReservationRecord> Get_Real_Records(){

    return realRecords;

}

}

```

- **ReservationRecord.java**

```
package report6;

public class ReservationRecord {

    // 저장할 강의실 이름, 요일, 시간, 예약자, 메모를 만들어준다.

    String room_num, day, name, memo;

    int time;

    public ReservationRecord(String[] data){

        // 입력 받은 데이터를 저장

        this.room_num = data[0];

        this.day = data[1].toLowerCase();

        this.time = Integer.parseInt(data[2])-1;

        this.name = data[3];

        this.memo = data[4];

    }

    // 요일이 정상적으로 입력되었는지 확인하는 메소드

    public boolean day_Check(){

        String[] days = {"sun","mon","tue","wed","thr","fri","sat"};

        boolean ret = false;

        for (int i = 0; i<days.length; i++){

            if(day.equals(days[i])){

                ret = true;

            }

        }

    }

}
```

```

    }

    if(ret == false){

        System.out.println("Input Day is wrong");

    }

    return ret;

}

// 시간이 정상적으로 입력 (1~8교시)되었는지 확인하는 메소드

public boolean time_Check(){

    boolean ret = false;

    if(0<=time && time<=7){

        ret = true;

    }

    if(ret == false){

        System.out.println("Input Time is wrong");

    }

    return ret;

}

// 요일의 위치를 반환해주는 메소드

public int get_Day_Index(){

    int index = -1;

    String[] days = {"sun","mon","tue","wed","thr","fri","sat"};

    for (int i = 0; i<days.length; i++){

        if(day.equals(days[i])){

            index = i;


```



```

        }
    }
    return index;
}
}

```

- ReservationGUI.java

```
package report6;
```

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
import javax.swing.*;
```

```
import javax.swing.table.*;
```

```
import java.util.ArrayList;
```

```
import java.io.*;
```

```
public class ReservationGUI extends JFrame {
```

```
    // JButton 변수 생성
```

```
    private JButton validTime = new JButton("예약가능시간");
```

```
    private JButton reservSituation = new JButton("방별예약상황");
```

```
    private JButton reserv = new JButton("예약");
```

```
    private JButton cancel = new JButton("취소");
```

```
    // JComboBox와 JTextField 생성
```

```
    // 방은 Room514, Room515, Room516 3개가 있다고 가정

```

```

private String[] room = {"Room514","Room515", "Room516"};

private JComboBox<String> roomNum = new JComboBox<String>(room);

private String[] day = {"sun","mon","tue","wed","thr","fri","sat"};

private JComboBox<String> reservDay = new JComboBox<String>(day);

private String[] time = {"1", "2", "3", "4", "5", "6", "7", "8"};

private JComboBox<String> reservTime = new JComboBox<String>(time);

private JComboBox<String> situation = new JComboBox<String>(room);

private JTextField reservName = new JTextField();

private JTextField reservMemo = new JTextField();


// RoomReservation에서 예약 파일에 저장되어있던 내용을 불러온다.
RoomReservation dataFile = new RoomReservation();

ArrayList <ReservationRecord> reservList = dataFile.Get_Real_Records();


public ReservationGUI() {

    // 호실, 요일, 시간, 예약자, 메모에 대한 Label과 JComboBox, JTextField 추가
    // GridLayout(2,5) 2행 5열
    JPanel p1 = new JPanel();
    p1.setLayout(new GridLayout(2, 5));

    p1.add(new JLabel("호실"));
    p1.add(new JLabel("요일"));
    p1.add(new JLabel("시간"));
    p1.add(new JLabel("예약자"));
    p1.add(new JLabel("메모"));

```

```

p1.add(roomNum);

p1.add(reservDay);

p1.add(reservTime);

p1.add(reservName);

p1.add(reservMemo);


// "예약 상황"이라고 JTextField 생성

JPanel p2 = new JPanel(new BorderLayout());

p2.add(new JTextField("예약 상황"));


// JTable 생성

DefaultTableModel model = new DefaultTableModel();

model.setColumnCount(5);

JTable p3 = new JTable(model);


// RoomReservation(datafile)에 저장되어있는 예약사항을 JTable에 표시

for(int i = 0; i<reservList.size(); i++){

    String[] input = new String[5];

    for(int j=0; j<input.length; j++){

        input[j] = null;

    }

    input[0] = reservList.get(i).room_num;

    input[1] = reservList.get(i).day;

    input[2] = String.valueOf(reservList.get(i).time + 1);

    input[3] = reservList.get(i).name;

    input[4] = reservList.get(i).memo;

```

```

        model.addRow(input);

    }

    // 예약가능시간 버튼 생성

    JPanel p4 = new JPanel(new BorderLayout());
    p4.add(validTime);

    // 방을 선택할 JComboBox, 예약상황버튼 JButton
    // FlowLayout으로 글자간격 5, 줄간격 10
    JPanel p5 = new JPanel();
    p5.setLayout(new FlowLayout(FlowLayout.LEFT, 5, 10));
    p5.add(situation);
    p5.add(reservSituation);

    // 예약, 취소버튼 생성
    // GridLayout(2,1) 2행 1열
    JPanel p6 = new JPanel();
    p6.setLayout(new GridLayout(2,1));
    p6.add(reserv);
    p6.add(cancel);

    // reserv, cancel를 클릭하면 ReservListener, Cancellistener가 각각 실행
    reserv.addActionListener(new ReservListener(p3, roomNum, reservDay, reservTime,
reservName, reservMemo));

    cancel.addActionListener(new Cancellistener(p3));

```

```
// p1을 NORTH, p2를 CENTER로 배치 & Grouping
```

```
JPanel p7 = new JPanel(new BorderLayout());
```

```
p7.add(p1, BorderLayout.NORTH);
```

```
p7.add(p2, BorderLayout.CENTER);
```

```
add(p7);
```

```
// p7을 NORTH, p3를 CENTER로 배치 & Grouping
```

```
JPanel p8 = new JPanel(new BorderLayout());
```

```
p8.add(p7, BorderLayout.NORTH);
```

```
p8.add(p3, BorderLayout.CENTER);
```

```
add(p8);
```

```
// p4를 CENTER, p5를 EAST로 배치 & Grouping
```

```
JPanel p9 = new JPanel(new BorderLayout());
```

```
p9.add(p4, BorderLayout.CENTER);
```

```
p9.add(p5, BorderLayout.EAST);
```

```
add(p9);
```

```
// p8를 CENTER, p6를 EAST로 배치 & Grouping
```

```
JPanel p10 = new JPanel(new BorderLayout());
```

```
p10.add(p8, BorderLayout.CENTER);
```

```
p10.add(p6, BorderLayout.EAST);
```

```
add(p10);
```

```
// p10을 CENTER, p9를 SOUTH로 배치 & Grouping
```

```
JPanel p = new JPanel(new BorderLayout());
```

```

p.add(p10, BorderLayout.CENTER);

p.add(p9, BorderLayout.SOUTH);

add(p);
}

```

// 예약 버튼을 누르면 새로운 예약사항을 JTable에 표시하고 예약사항 파일을 갱신

```

class ReservListener implements ActionListener{

    JTable table;

    JComboBox roomNum, reservDay, reservTime;

    JTextField reservName, reservMemo;

    // 새로운 예약 사항을 저장

    ReservListener(JTable table, JComboBox roomNum, JComboBox reservDay,
                    JComboBox reservTime, JTextField reservName, JTextField
reservMemo){

        this.table = table;

        this.roomNum = roomNum;

        this.reservDay = reservDay;

        this.reservTime = reservTime;

        this.reservName = reservName;

        this.reservMemo = reservMemo;

    }

    public void actionPerformed(ActionEvent e){

```

```

// 예약사항을 String 배열에 저장

String[] input = new String [5];

input[0] = (String)roomNum.getSelectedItemAt();

input[1] = (String)reservDay.getSelectedItemAt();

input[2] = (String)reservTime.getSelectedItemAt();

input[3] = reservName.getText();

input[4] = reservMemo.getText();


// ReservationRecord에 저장

ReservationRecord newReservation = new ReservationRecord(input);


// 예약 리스트에 추가 , 같은 것이 있을 경우 추가 X

boolean conflict = false;

for(int i = 0; i<table.getRowCount(); i++){

    if(table.getValueAt(i, 0).equals(input[0]) && table.getValueAt(i,
1).equals(input[1]) && table.getValueAt(i, 2).equals(input[2])){

        System.out.println("Your          new
reservation conflicted");

        conflict = true;

    }

}

// 같은것이 없으면 jTable 예약 리스트에 추가 & 예약파일 갱신

if(!conflict){

    // jTable 예약 리스트에 추가

    DefaultTableModel      model      =      (DefaultTableModel)
table.getModel();

```

```

        model.addRow(input);

        // 예약 파일 갱신
        BufferedWriter bw;
        try {
            bw = new BufferedWriter(new FileWriter("roomreserve-
norm.data",true));

            PrintWriter pw = new PrintWriter(bw,true);

            String line =
input[0]+":"+input[1]+":"+input[2]+":"+input[3]+":"+input[4];

            pw.println();
            pw.write(line);
            pw.flush();
            pw.close();
        }
        catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }
    }

}
}

```


// 예약 버튼을 누르면 새로운 예약사항을 JTable에 표시하고 예약사항 파일을 갱신

```
class CancelListener implements ActionListener{
```

```
    JTable table;
```

```
    CancelListener(JTable table){
```

```
        this.table = table;
```

```
    }
```

```
    public void actionPerformed(ActionEvent e){
```

```
        // datafile에서 선택한 예약사항의 line을 지운다
```

```
        int row = table.getSelectedRow();
```

```
        File file = new File("roomreserve-norm.data");
```

```
        String dummy = "";
```

```
        try{
```

```
            BufferedReader search = new BufferedReader(new  
InputStreamReader(new FileInputStream(file)));
```

```
            String searchLine = search.readLine();
```

```
            int position = -1;
```

```
            // 제거할 line의 위치 찾기
```

```
            for(int i = 0; searchLine != null; i++){
```

```
                if (!searchLine.startsWith("//") &&  
searchLine.length() != 0){
```

```

String[] tokens = searchLine.split(":");

String[] data = new String[5];

if(tokens.length == 5){

    data[0] = tokens[0].trim();

    data[1] = tokens[1].trim().toLowerCase();

    data[2] = tokens[2].trim();

    data[3] = tokens[3].trim();

    data[4] = tokens[4].trim();

}

else if(tokens.length == 4){

    data[0] = tokens[0].trim();

    data[1] = tokens[1].trim().toLowerCase();

    data[2] = tokens[2].trim();

    data[3] = tokens[3].trim();

}

if((data[0].equals((String)(table.getValueAt(row,
0))))

    &&

(data[1].equals((String)(table.getValueAt(row, 1).toString().toLowerCase())))

    &&

(data[2].equals((String)(table.getValueAt(row, 2))))){

    position = i;

}

}

```

```

        searchLine = search.readLine();
    }

    search.close();

    // 일치하는 예약 사항의 line을 찾지 못했을 경우
    if(position == -1){
        System.out.println("There is no statement");
    }
    else{

        // 제거할 예약사항을 제외하고 내용을 저장한다.

        String line;

        BufferedReader br = new BufferedReader(new
InputStreamReader(new FileInputStream(file)));

        for(int i=0; i<position; i++){
            line = br.readLine();
            dummy += (line+ "WrWn");
        }

        String delData = br.readLine();
        while((line = br.readLine()) != null){
            dummy += (line+ "WrWn");
        }

        // 원래 datafile에 내용을 덮어씌운다.

        FileWriter fw = new FileWriter("roomreserve-

```

```

norm.data");

                                fw.write(dummy);

                                fw.close();
                                br.close();
                                }
        }
        catch(Exception ex){
            ex.printStackTrace();
        }

        // JTable에서 선택한 예약을 제거
        if(row == -1){
            return;
        }

        DefaultTableModel model = (DefaultTableModel) table.getModel();
        model.removeRow(row);

    }

}

```

6. 자체 평가표

평가 항목	학생 자체 평가 (리포트 해당 부분 표시 및 간단한 의견)	평가 (빈칸)	점수 (빈칸)
<p>완성도 (동작 여부)</p> <ul style="list-style-type: none"> - 직접 입력 후, 예약 동작 - 리스트에서 과목 선택하면? - "제거" 동작 - 파일 내용 또는 종료/재시작으로 예약/제거 동작 확인 - 기타 비정상 동작 실험 	<p>예약 목록을 직접 입력 후, 예약 버튼을 누르면 정상적으로 GUI에 추가되고, datafile도 갱신된다.</p> <p>리스트에서 과목을 선택할 수 있다.</p> <p>리스트에서 과목을 선택 한 후, 취소를 누르면 GUI에서 제거되고 datafile에서도 그 문장이 지워진다.</p> <p>파일의 내용을 예약, 제거 버튼을 누를 때마다 재시작하여 제대로 동작했는지 확인했다.</p> <p>이미 있는 시간에 예약이 추가될 경우 오류 문장을 출력 하도록 했다.</p> <p>(리포트 p.11~14 Test 결과)</p>		
<p>설계 노트</p> <ul style="list-style-type: none"> - 주요 결정사항 및 근거 - 한계점/문제점 - 해결 방안 - 필수내용: 프로그램 구성(Class 구조) member visibility 	<p>project 2와 4의 내용 외에 추가적으로 구현한 class와 기능들의 구조를 나타내었고 구현 할 때의 시행착오와 그 해결방안을 작성하였다.</p> <p>(리포트 p.4~10 설계 노트)</p>		

<p>리포트</p> <p>-평가자 시각으로 리포트 검토</p> <p>-위의 평가 요소들이 명확하게 기술되었는가?</p>	<p>목차를 작성하였고, 페이지 번호를 이용하여 순서대로 레포트를 작성하였다.</p> <p>위의 평가 요소들을 명확하게 작성하였다.</p>		
<p>총평/계</p>			