소프트웨어 프로젝트 6

20131790

권소현

목차

1. 프로젝트 목표

Project5의 GUI를 확장하여 호실 예약 프로그램을 설계한다.

2. 프로젝트 내용

- 예약가능시간 버튼을 누르면, 새로운 창이 생기면서 예약 가능 시간을 보여준다.
- 호실을 선택한 후, 방별예약상황 버튼을 누르면, 해당 호실의 현재 예약 상황을 보여준다.
- 방별예약상황 창에서 예약을 원하는 칸을 클릭하면, 해당하는 강의실, 요일, 시간이 입력 창에 자동으로 입력된다.
- 방별예약상황 창에서 취소를 원하는 칸을 클릭하면, 해당하는 내용이 있는 예약상황리스트에서 자동으로 선택된다.
- 예약 또는 취소를 수행하면 표가 자동으로 갱신

3. 설계 노트

1) 추가된 ReservationGUI 클래스

- **변수의 접근제한자 변경**

타입	변수명	접근제한자 변경
JComboBox<String>	roomNum	public static
JComboBox<String>	reservDay	public static
JComboBox<String>	reservTime	public static
JComboBox<String>	situation	public static
String[]	room	private static
String[]	day	private static
String[]	time	private static
RoomReservation	dataFile	private static
ArrayList <ReservationRecord>	reservList	private static

- **ReservationGUI 추가 내용**

예약가능시간과 방별예약상황을 눌렀을 때 ActionListener가 실행되도록 한다.

```
                // validTime, reservSituation을 클릭하면
ValidTimeListener와 ReservationStatus가 각각 실행
        validTime.addActionListener(new ValidTimeListener(reservList));
        reservSituation.addActionListener(new ReservationStatus(reservList,
situation));
```

- **ValidTimeListener class 추가 (ReservationGUI의 inner class)**

  예약가능시간을 눌렀을 때 새로운 JFrame창이 뜨도록 한다.

```
class ValidTimeListener implements ActionListener{
	AvailableTime valid;
	ArrayList<ReservationRecord> reservList;
	ValidTimeListener(ArrayList<ReservationRecord> reservList){
		this.reservList = reservList;
	}
	@Override
	public void actionPerformed(ActionEvent e){
		valid = new AvailableTime(reservList);
		valid.setTitle("AvailableTime_20131790_권소현");
		valid.setSize(700, 500);
		valid.setLocationRelativeTo(null);
		valid.setVisible(true);
	}
}
```

- **ReservationStatus class 추가 (ReservationGUI의 inner class)**

  방별예약상황을 눌렀을 때 새로운 JFrame창이 뜨도록 한다.

```
class ReservationStatus implements ActionListener{
	RoomStatus status;
	ArrayList<ReservationRecord> reservList;
	JComboBox situation;
	ReservationStatus(ArrayList<ReservationRecord> reservList,
JComboBox situation){
		this.reservList = reservList;
		this.situation = situation;
	}
	@Override
	public void actionPerformed(ActionEvent e){
		status = new RoomStatus(reservList,
this.situation.getSelectedItem().toString());
		String name = situation.getSelectedItem().toString()+"
Reservation Status";
		status.setTitle(name);
		status.setSize(700, 500);
		status.setLocationRelativeTo(null);
		status.setVisible(true);
	}
}
```

2) ReservationChart class 추가

- **import**

  import java.util.*;

- **변수**

타입	변수명	접근제한자	역할
String[]	roomName	private	강의실 이름 저장
String[][]	roomSituation	private	강의실 예약 상황 저장

- **생성자**

  roomSituation을 초기화 한 후 예약 내역을 저장한다.

```
ReservationChart(ArrayList<ReservationRecord> reservList){

		// roomSituation 초기화
		for(int i = 0; i<roomSituation.length; i++){
			for(int j = 0; j<roomSituation[0].length; j++){
				for(int k = 0; k<roomSituation[0][0].length;
k++){
					roomSituation[i][j][k] = null;
				}
			}
		}

		// 예약 내역 저장
		for(int i = 0; i<reservList.size(); i++){
			int roomIndex =
GetRoomIndex(reservList.get(i).room_num);
			int dayIndex = reservList.get(i).get_Day_Index();

	if(roomSituation[roomIndex][reservList.get(i).time][dayIndex] ==
null){

	roomSituation[roomIndex][reservList.get(i).time][dayIndex] =
reservList.get(i).name;
			}
		}
	}
```

- **메소드**

메소드명	접근제한자	반환 타입	인자	역할
GetChart	public	String[][][]		roomSituation 전달
GetRoomIndex	public	int	String room	강의실의 index를 찾아준다.

3) AvailableTime class 추가

- **import**

  import java.awt.*;

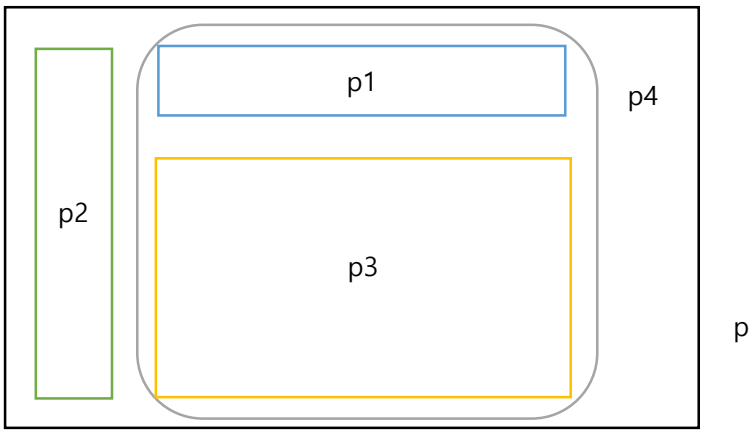
  import java.util.ArrayList;

  import javax.swing.*;

- **생성자**

변수명	타입	역할	Layout
getChart	ReservationChart	예약 내역을 불러온다	
chart	String[][][]	예약 내역 저장	
p1	JPanel	요일 JLabel	GridLayout(1,7)
p2	JPanel	시간 JLabel	GridLayout(9,1)
p3	JPanel	예약가능 표시 버튼	GridLayout(8,7)
p4	JPanel	p1, p3 Grouping	BorderLayout()
p	JPanel	p2, p4 Grouping	BorderLayout()

- **Panel 구조**



4) RoomStatus class 추가

- **import**

```
import java.util.ArrayList;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
```

- **변수**

타입	변수명	접근제한자	역할
JButton[][]	buttons	private	2차원 배열의 예약 상황 버튼
Object	SelRoomName	private	현재 강의실 저장
Object	selDay	private	클릭한 요일 저장
Object	selTime	private	클릭한 시간 저장

- **생성자**

변수명	타입	역할	Layout
getChart	ReservationChart	예약 내역을 불러온다	
chart	String[][][]	예약 내역 저장	
p1	JPanel	요일 JLabel	GridLayout(1,7)
p2	JPanel	시간 JLabel	GridLayout(9,1)
p3	JPanel	예약 상황 표시 버튼	GridLayout(8,7)
p4	JPanel	p1, p3 Grouping	BorderLayout()
p	JPanel	p2, p4 Grouping	BorderLayout()

- **ButtonListener class 추가 (RoomStatus의 inner class)**

  예약 상황의 버튼을 누르면 빈 칸의 내용을 상황에 예약 내용을 바꾸거나 표시해준다.

  버튼이 예약이 비어있는 경우 :
  ReservationGUI의 예약 입력 JComboBox에서 예약 내용으로 바꾸기

  버튼이 예약이 되어있는 경우 :
  예약 취소할 내용을 table에서 자동으로 선택

```
class ButtonListener implements ActionListener{
		Object[] day = {"sun","mon","tue","wed","thr","fri","sat"};
		Object[] time = {"1", "2", "3", "4", "5", "6", "7", "8"};

		public void actionPerformed(ActionEvent e){
			for(int i = 0; i<8; i++){
				for(int j = 0; j<7; j++){
					if(e.getSource() == buttons[i][j]){

						// 누른 버튼의 요일, 시간 내용 저장
						selTime = time[i];
						selDay = day[j];

						// 버튼이 비어있는경우 -> 예약 내용으로
바꾸기
						if(buttons[i][j].getText() == ""){

	ReservationGUI.roomNum.setSelectedItem(selRoomName);

	ReservationGUI.reservDay.setSelectedItem(selDay);

	ReservationGUI.reservTime.setSelectedItem(selTime);
						}

						// 버튼이 예약되어있는 경우 -> 예약
취소할 table 선택
						else{
							for(int k = 0;
k<ReservationGUI.p3.getRowCount(); k++){

	if(ReservationGUI.p3.getValueAt(k,0).equals(selRoomName)
&&
ReservationGUI.p3.getValueAt(k, 1).equals(selDay)
&&
ReservationGUI.p3.getValueAt(k, 2).equals(selTime)){

	ReservationGUI.p3.setRowSelectionInterval(k, k);
}
```

```
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
}
```

- **시행 착오 및 해결**

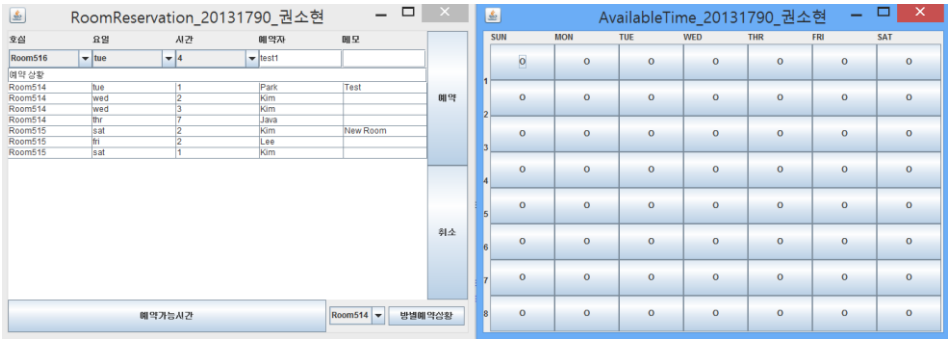
  1) 예약 상황을 저장하는 배열을 클래스를 만들지 않고 ReservationGUI에서 직접 나타냈었는데 예약가능시간과 방별예약 상황 GUI를 만들면서 표를 가져올 방법이 필요했다.
     따라서 해결방안으로 ReservationChart라는 새로운 클래스를 만들어 ReservationGUI에 있는 arrayList를 사용하여 배열을 만들었다.
     첫 과제 때 class를 만들어서 했다면 더 편하지 않았을까 싶었다.

  2) 여태 ActionListener를 사용 할 때 하나의 button에 하나의 inner class를 만들었었는데 RoomStatus에서 총 56개의 버튼의 ActionListener를 사용하려고 하니 새로운 방법을 찾아야했다.
     따라서 해결방안으로 JButton을 2차원 배열로 만들고 actionPerformed에서 getSource() 메소드를 사용하여 만들었다.

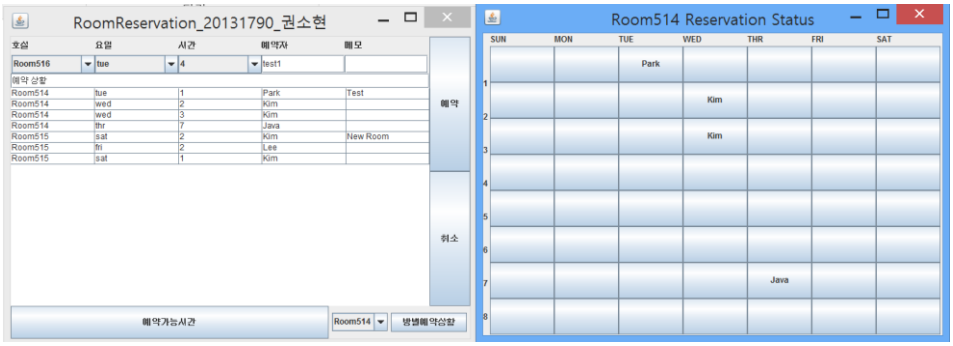
4. Test 결과

- 처음 화면 (GUI가 정상 작동하는가?)

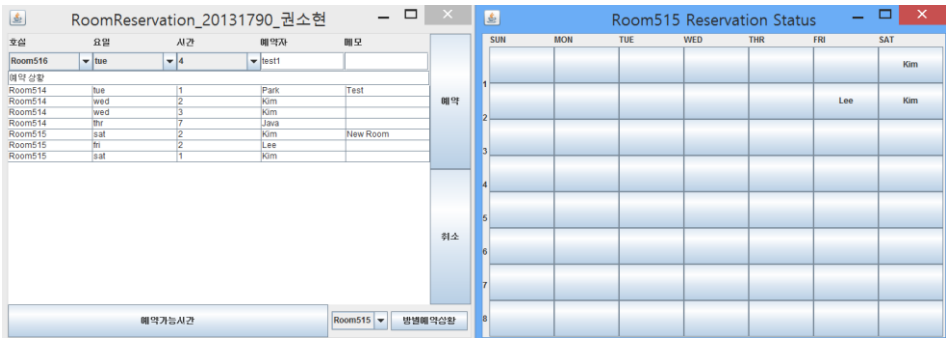
1) 예약가능 시간



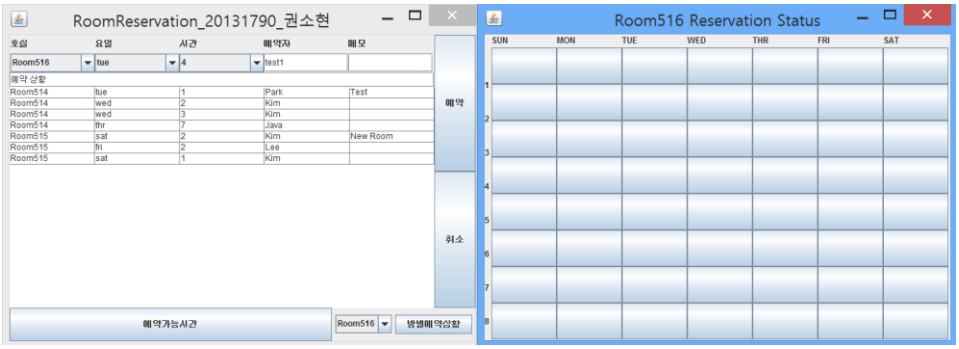
2) Room514 예약 상황



3) Room515 예약 상황

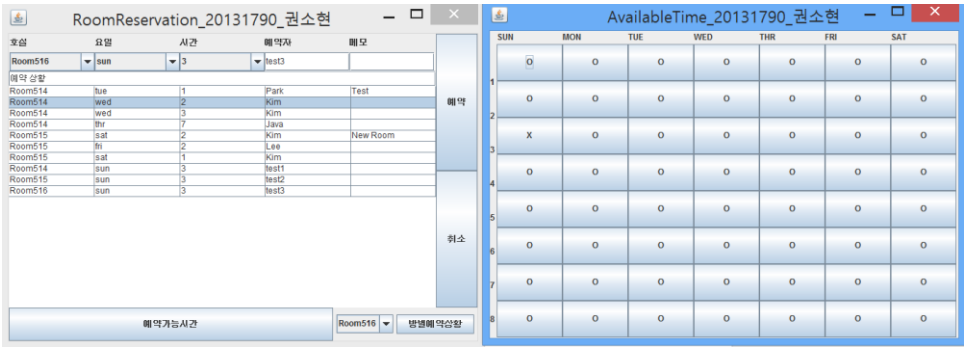


4) Room516 예약 상황

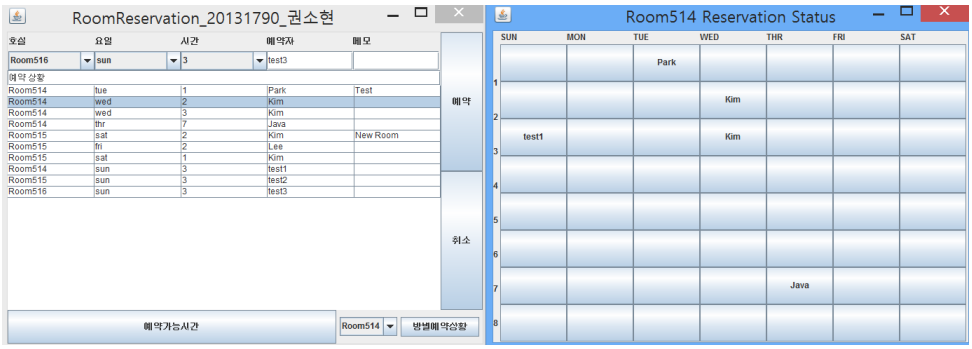


- 일요일 3교시 모든 강의실에 예약을 추가했을 경우
(예약 버튼이 작동하는가? 예약 버튼을 눌렀을 때 예약가능시간, 예약 상황이 갱신되는가?)

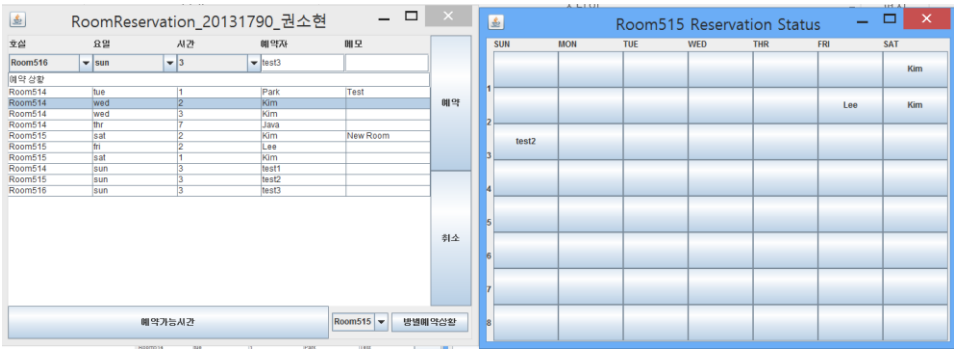
1) 예약 가능 시간



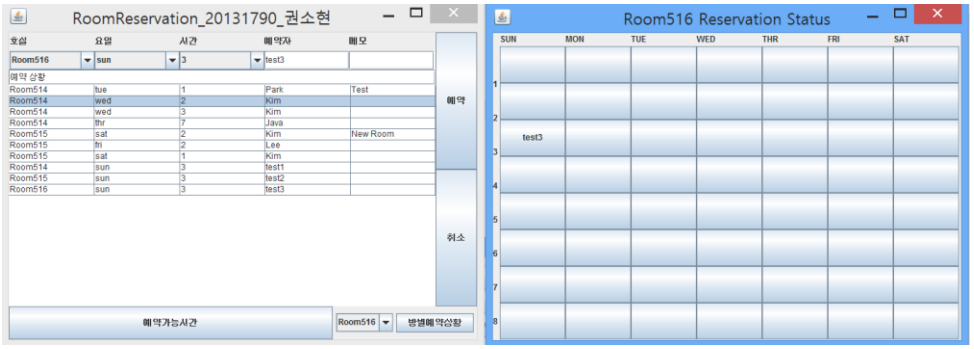
2) Room514 예약 상황



3) Room515 예약 상황

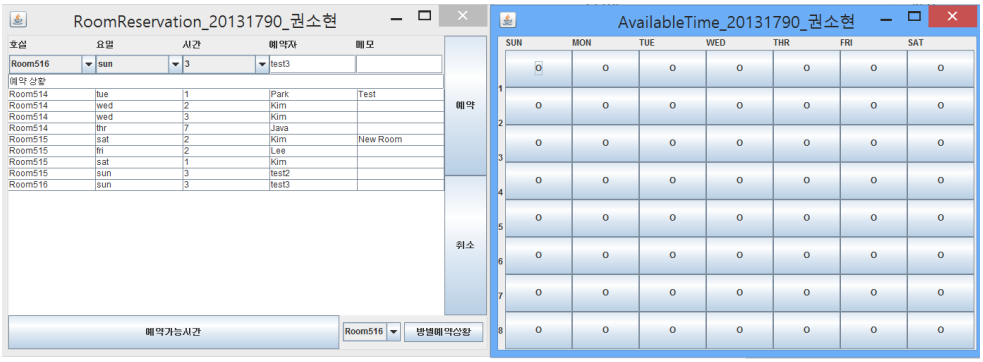


4) Room516 예약 상황

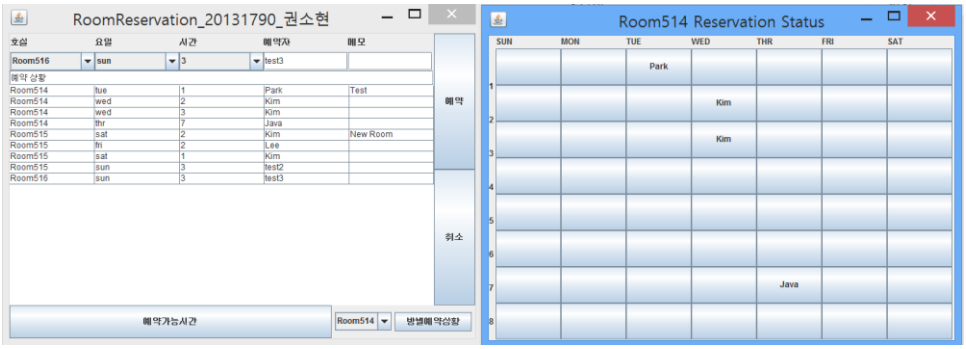


- 일요일 Room514 3교시의 예약을 지웠을 때, 예약 가능 시간에서 X가 사라지고, Room514의 3교시 예약 상황이 지워지는가? (취소 버튼이 작동하는가? 취소를 눌렀을 때 예약가능시간, 예약 상황이 갱신되는가?)

1) 예약 가능 시간

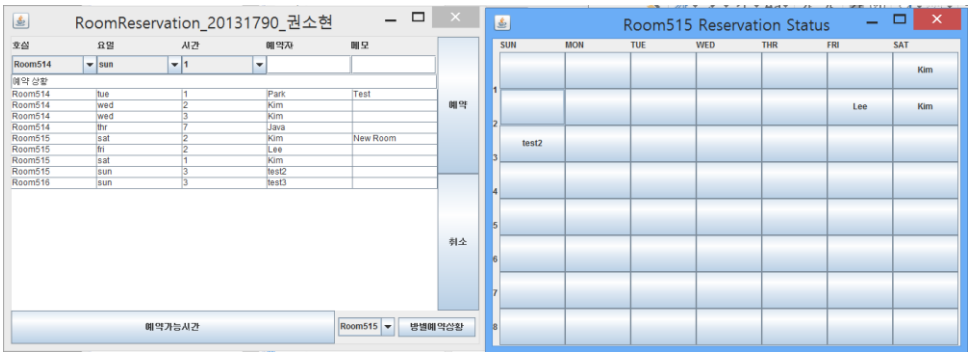


2) Room514 예약 상황

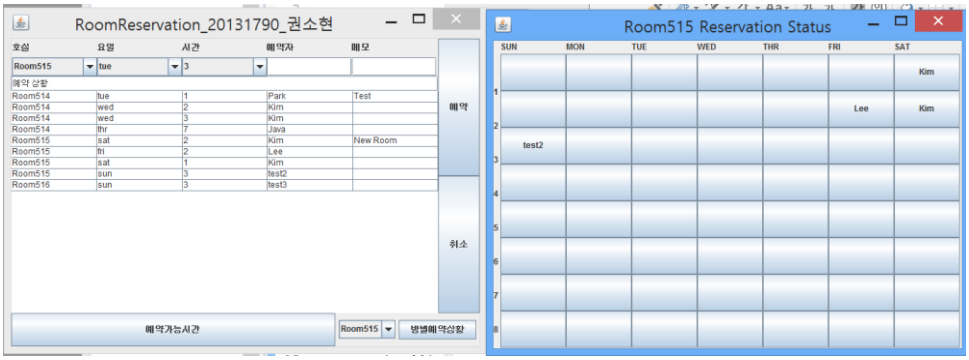


- 방별예약상황에서 예약이 없는 곳의 button을 눌렀을 경우

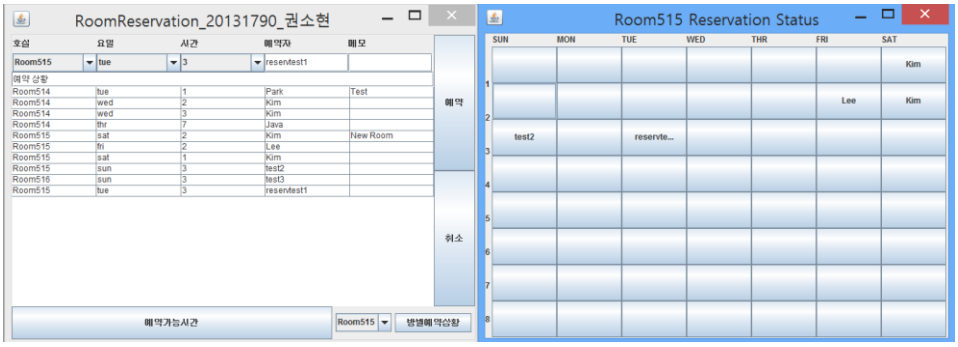
1) 호실이 Room514, 요일이 sun, 시간이 1로 되어있을 때



2) 호실이 Room515, 요일이 Tue, 시간이 3을 누르면 JComboBox에서 내용이 바뀐다.



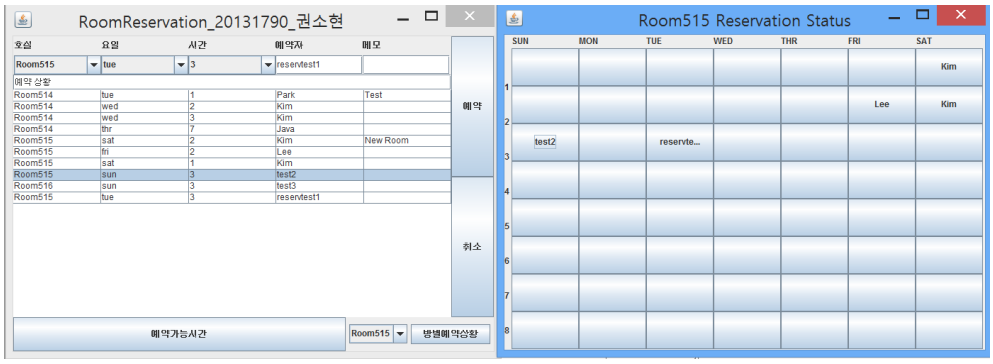
3) 예약자를 reservtest1으로 하면 table에 추가가 되고 Room515의 방별예약상황도 바뀐다.



- 방별예약상황에서 예약이 있는 곳의 button을 눌렀을 경우

위의 3번 상황에서 test2를 눌렀을 경우

강의실 Room515, 요일 Sun, 시간 3의 내용을 table에서 찾아 선택한다.



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 {

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

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

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

	private static ArrayList <ReservationRecord> records = new ArrayList<ReservationRecord>();

	private 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 문장이나 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 제외시킨다. (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 중복X, 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 저장되어있는 강의실의 위치를 반환
            if(record.room_num.equals(room_name[i])){
                return i;
            }
        }
        // 나의 강의실이 저장되어있지 않고, 강의실의 갯수가 2개이하라면
        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("₩t");
                                }
                                else{
                                        System.out.print(rooms[i][j][k]+"₩t");
                                }
                        }
                        // 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 가정9
    public static JTable p3;
    private static String[] room = {"Room514","Room515", "Room516"};
    public static JComboBox<String> roomNum = new JComboBox<String>(room);
```

```
        private static String[] day = {"sun","mon","tue","wed","thr","fri","sat"};
        public static JComboBox<String> reservDay = new JComboBox<String>(day);
        private static String[] time = {"1", "2", "3", "4", "5", "6", "7", "8"};
        public static JComboBox<String> reservTime = new JComboBox<String>(time);
        private JComboBox<String> situation = new JComboBox<String>(room);
        public JTextField reservName = new JTextField();
        public JTextField reservMemo = new JTextField();

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

// validTime, reservSituation을 클릭하면 ValidTimeListener와 ReservationStatus가 각각 실행

validTime.addActionListener(**new** ValidTimeListener(*reservList*));

reservSituation.addActionListener(**new** ReservationStatus(*reservList*, situation));

// 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);
    }
        public ArrayList<ReservationRecord> Get_Chart_List(){
                return reservList;
        }

        //예약 버튼을 누르면 새로운 예약사항을 JTable에 표시하고 예약사항 파일을 갱
신
        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.getSelectedItem();
            input[1] = (String)reservDay.getSelectedItem();
            input[2] = (String)reservTime.getSelectedItem();
            input[3] = reservName.getText();
            input[4] = reservMemo.getText();
```

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

                    // 에약 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 갱
신
                    if(!conflict){

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

                        // reservList에 추가
                        reservList.add(newReservation);
```

```
                    // 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) {
                        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 = "";

        // 지울 내용을 저장한다 (비교해서 지울때 필요)
        // 강의실 번호
        String delRoom = table.getValueAt(row, 0).toString();
        // 요일
        String delDay = table.getValueAt(row, 1).toString().toLowerCase();
        // 시간
        int delTime = (Integer.parseInt(table.getValueAt(row,
2).toString())-1);

        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(delRoom))
                            &&
(data[1].equals(delDay))
                            &&
(Integer.parseInt(data[2])-1 == delTime)){
                        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 저장
한다.
                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();
                }
            }
            catch(Exception ex){
                ex.printStackTrace();
            }

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

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



                    // reservList의 어디에 있는지 찾아낸다.
                    int where = -1;
                    for(int i = 0; i<reservList.size(); i++){
                        if((reservList.get(i).room_num.equals(delRoom))

        &&(reservList.get(i).day.equals(delDay))
                                        &&  (reservList.get(i).time  ==
delTime )){
                            where = i;
                        }
                    }


                    // reservList에서 지우기
                    if(where == -1){
                        System.out.println("Can't find reservList");
                    }
                    else{
                        reservList.remove(where);
                    }
```

```
		}
	}

	class ValidTimeListener implements ActionListener{
		AvailableTime valid;
		ArrayList<ReservationRecord> reservList;
		ValidTimeListener(ArrayList<ReservationRecord> reservList){
			this.reservList = reservList;
		}
		@Override
		public void actionPerformed(ActionEvent e){
			valid = new AvailableTime(reservList);
			valid.setTitle("AvailableTime_20131790_권소현");
			valid.setSize(700, 500);
			valid.setLocationRelativeTo(null);
			valid.setVisible(true);
		}
	}

	class ReservationStatus implements ActionListener{
		RoomStatus status;
		ArrayList<ReservationRecord> reservList;
		JComboBox situation;
		ReservationStatus(ArrayList<ReservationRecord> reservList, JComboBox
situation){
```

```
                this.reservList = reservList;
                this.situation = situation;
            }
            @Override
            public void actionPerformed(ActionEvent e){
                status = new RoomStatus(reservList, this.situation.getSelectedItem().toString());
                String name = situation.getSelectedItem().toString()+" Reservation Status";
                status.setTitle(name);
                status.setSize(700, 500);
                status.setLocationRelativeTo(null);
                status.setVisible(true);
            }
        }
}
```

- **ReservationChart.java**

```
package report6;

import java.util.*;

public class ReservationChart{

	// 방 Room514, Room515, Room516을 String[] 배열에 순서대로 지정
	private String[] roomName = {"Room514", "Room515", "Room516"};
	// 3개의 강의실의 예약을 저장할 3차원 배열을 생성
	private String[][][] roomSituation = new String[3][8][7];

	ReservationChart(ArrayList<ReservationRecord> reservList){

		// roomSituation 초기화
		for(int i = 0; i<roomSituation.length; i++){
			for(int j = 0; j<roomSituation[0].length; j++){
				for(int k = 0; k<roomSituation[0][0].length; k++){
					roomSituation[i][j][k] = null;
				}
			}
		}
```

```
                // 예약 내역 저장
                for(int i = 0; i<reservList.size(); i++){
                        int roomIndex = GetRoomIndex(reservList.get(i).room_num);
                        int dayIndex = reservList.get(i).get_Day_Index();
                        if(roomSituation[roomIndex][reservList.get(i).time][dayIndex] ==
null){

        roomSituation[roomIndex][reservList.get(i).time][dayIndex] = reservList.get(i).name;
                        }
                }
        }

        public String[][][] GetChart(){
                return roomSituation;
        }

        public int GetRoomIndex(String room){
                int index = -1;
                for(int i = 0; i<roomName.length; i++){
                        if(roomName[i].equals(room)){
                                index = i;
                        }
                }
                if(index == -1){
                        System.out.println("can't get index");
```

```
                }

                return index;

            }

        }
```

- **AvailableTime.java**

```
package report6;

import java.awt.*;
import java.util.ArrayList;
import javax.swing.*;

public class AvailableTime extends JFrame {

	public AvailableTime(ArrayList<ReservationRecord> reservList){
		ReservationChart getChart = new ReservationChart(reservList);
		String[][][] chart = getChart.GetChart();
		// 요일에 대한 Label
		// GridLayout(1,7)

		JPanel p1 = new JPanel();
		p1.setLayout(new GridLayout(1,7));
		p1.add(new JLabel("SUN"));
		p1.add(new JLabel("MON"));
		p1.add(new JLabel("TUE"));
		p1.add(new JLabel("WED"));
		p1.add(new JLabel("THR"));
		p1.add(new JLabel("FRI"));
```

```
p1.add(new JLabel("SAT"));



// 시간에 대한 Label

// GridLayout(9,1)


JPanel p2 = new JPanel();

p2.setLayout(new GridLayout(9,1));



p2.add(new JLabel(""));

p2.add(new JLabel("1"));

p2.add(new JLabel("2"));

p2.add(new JLabel("3"));

p2.add(new JLabel("4"));

p2.add(new JLabel("5"));

p2.add(new JLabel("6"));

p2.add(new JLabel("7"));

p2.add(new JLabel("8"));


// 예약 가능 표시를 위한 버튼 생성

JPanel p3 = new JPanel();

p3.setLayout(new GridLayout(8,7));

for(int i = 0; i<chart[0].length; i++){
```

```
                    for(int j = 0; j<chart[0][0].length; j++){
                        if(chart[0][i][j] == null
                                || chart[1][i][j] == null
                                || chart[2][i][j] == null){
                            p3.add(new JButton("O"));
                        }
                        else{
                            p3.add(new JButton("X"));
                        }
                    }
                }
                // p1와 p3를 Grouping
                JPanel p4 = new JPanel(new BorderLayout());
                p4.add(p1, BorderLayout.NORTH);
                p4.add(p3, BorderLayout.CENTER);
                add(p4);

                // p2과 p4를 Grouping
                JPanel p = new JPanel(new BorderLayout());
                p.add(p2, BorderLayout.WEST);
                p.add(p4, BorderLayout.CENTER);
                add(p);
        }
}
```

- **RoomStatus.java**

```
package report6;



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

public class RoomStatus extends JFrame{

	private JButton[][] buttons = new JButton[8][7];
	private Object selRoomName;
	private Object selDay;
	private Object selTime;

	public RoomStatus(ArrayList<ReservationRecord> reservList, String situation) {
		ReservationChart getChart = new ReservationChart(reservList);
		String[][][] chart = getChart.GetChart();
		selRoomName = (Object)situation;
		// 요일에 대한 Label
		// GridLayout(1,7)
```

```
JPanel p1 = new JPanel();

p1.setLayout(new GridLayout(1,7));

p1.add(new JLabel("SUN"));

p1.add(new JLabel("MON"));

p1.add(new JLabel("TUE"));

p1.add(new JLabel("WED"));

p1.add(new JLabel("THR"));

p1.add(new JLabel("FRI"));

p1.add(new JLabel("SAT"));


// 시간에 대한 Label

// GridLayout(9,1)


JPanel p2 = new JPanel();

p2.setLayout(new GridLayout(9,1));


p2.add(new JLabel(""));

p2.add(new JLabel("1"));

p2.add(new JLabel("2"));

p2.add(new JLabel("3"));

p2.add(new JLabel("4"));

p2.add(new JLabel("5"));

p2.add(new JLabel("6"));

p2.add(new JLabel("7"));
```

```
p2.add(new JLabel("8"));


// 예약 내역 표시를 위한 버튼 생성
JPanel p3 = new JPanel();
p3.setLayout(new GridLayout(8,7));
int reservRoomIndex = getChart.GetRoomIndex(situation);
for(int i = 0; i<chart[reservRoomIndex].length; i++){
        for(int j = 0; j<chart[reservRoomIndex][0].length; j++){
            if(chart[reservRoomIndex][i][j] == null){
                buttons[i][j] = new JButton("");
                p3.add(buttons[i][j]);

            }
            else{

                buttons[i][j] = new
JButton(chart[reservRoomIndex][i][j]);
                p3.add(buttons[i][j]);
            }
        }
}

for(int i = 0; i<8; i++){
        for(int j = 0; j<7; j++){
            buttons[i][j].addActionListener(new ButtonListener());
```

```
                }
            }

            // p1와 p3를 Grouping
            JPanel p4 = new JPanel(new BorderLayout());
            p4.add(p1, BorderLayout.NORTH);
            p4.add(p3, BorderLayout.CENTER);
            add(p4);

            // p2과 p4를 Grouping
            JPanel p = new JPanel(new BorderLayout());
            p.add(p2, BorderLayout.WEST);
            p.add(p4, BorderLayout.CENTER);
            add(p);

}

// 예약 상황의 버튼을 누르면 빈 칸을 예약 내용의 JComboBox로 변경

class ButtonListener implements ActionListener{
            Object[] day = {"sun","mon","tue","wed","thr","fri","sat"};
            Object[] time = {"1", "2", "3", "4", "5", "6", "7", "8"};

            public void actionPerformed(ActionEvent e){
```

```
for(int i = 0; i<8; i++){
    for(int j = 0; j<7; j++){
        if(e.getSource() == buttons[i][j]){

            // 누른 버튼의 요일, 시간 내용 저장
            selTime = time[i];
            selDay = day[j];

            // 버튼이 비어있는경우 -> 예약 내용으로 바꾸기
            if(buttons[i][j].getText() == ""){
ReservationGUI.roomNum.setSelectedItem(selRoomName);
ReservationGUI.reservDay.setSelectedItem(selDay);
ReservationGUI.reservTime.setSelectedItem(selTime);
            }

            // 버튼이 예약되어있는 경우 -> 예약 취소할 table 선택
            else{
                for(int k = 0; k<ReservationGUI.p3.getRowCount(); k++){
if(ReservationGUI.p3.getValueAt(k,0).equals(selRoomName)
```

```
            && ReservationGUI.p3.getValueAt(k, 1).equals(selDay)

            && ReservationGUI.p3.getValueAt(k, 2).equals(selTime)){

            ReservationGUI.p3.setRowSelectionInterval(k, k);
                                                                    }
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

6. 자체 평가표

평가 항목	학생 자체 평가 (리포트 해당 부분 표시 및 간단한 의견)	평가 (빈칸)	점수 (빈칸)
완성도 (동작 여부) - "예약가능" 동작 - "방별상황" 동작 - 상황표에서 선택을 통한 예약/취소 동작 - 예약/취소 동작 시 가능/상황표 자동 갱신 - 기타 비정상 동작 실험	예약 가능, 방별상황, 상황표에서 선택을 통한 예약/ 취소 동작, 예약/ 취소 동작시 가능/상황표 자동 갱신 비정상적인 동작에 대해 실험을 마치고 모두 제대로 작동하였다. (리포트 p.11~15 Test 결과)		
설계 노트 - 주요 결정사항 및 근거 - 한계점/문제점 - 해결 방안 - 필수내용: 프로그램 구성(Class 구조) member visibility	project 5 내용 외에 추가적으로 구현한 class와 기능들의 구조를 나타내었고 구현 할 때의 시행착오와 그 해결방안을 작성하였다. (리포트 p.4~10 설계 노트)		
리포트 -평가자 시각으로 리포트 검토 -위의 평가 요소들이 명확하게 기술되었는가?	목차를 작성하였고, 페이지 번호를 이용하여 순서대로 레포트를 작성하였다. 위의 평가 요소들을 명확하게 작성하였다.		
총평/계			