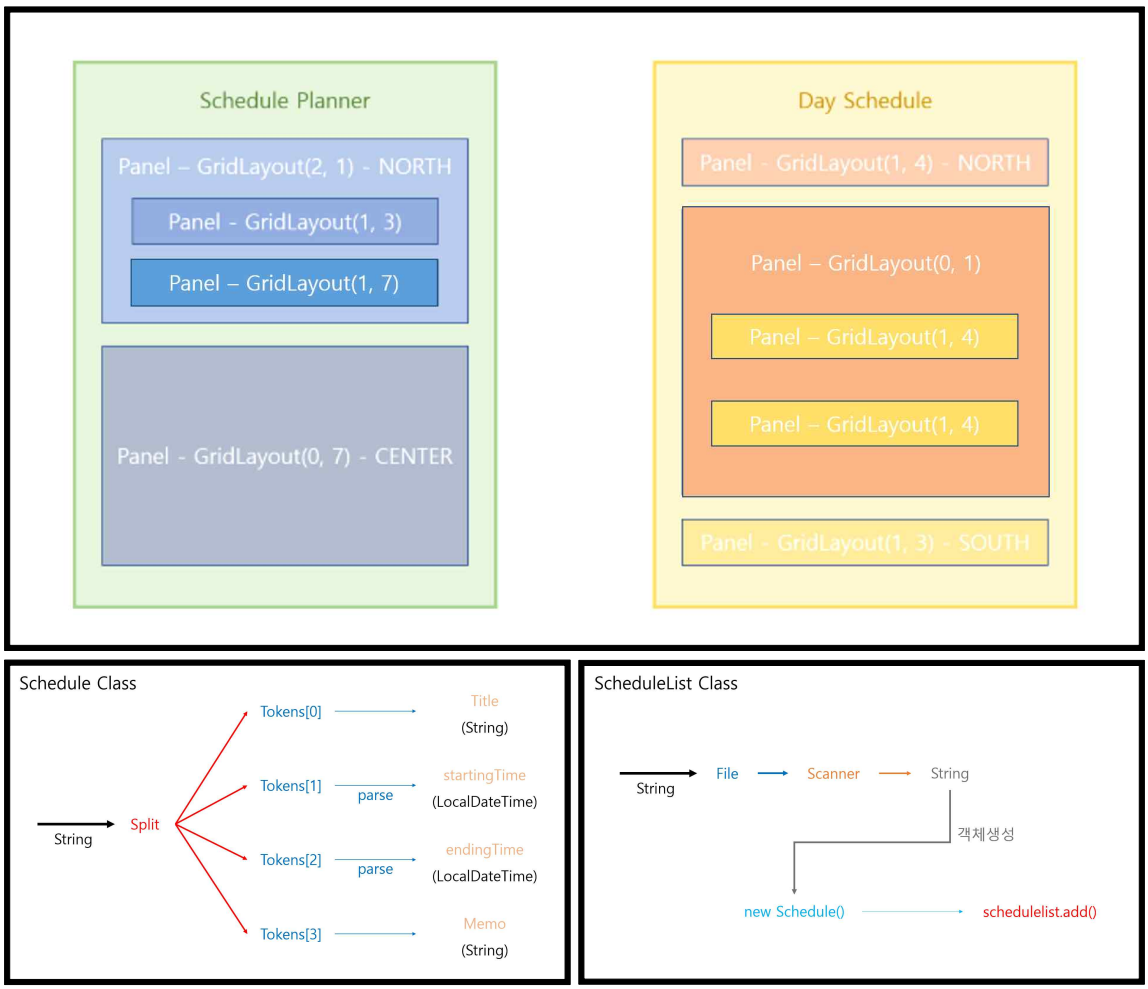
소프트웨어프로젝트

< 6. Schedule Planner 최종 설계 및 구현 >

분반 : 02분반
이름 : 김두홍
학번 : 20191127
제출일 : 2020.06.21

1. 설계 노트



"schedule-file.data" 파일에서 스케줄 정보를 한 줄씩 읽으며, 각각 Schedule 객체를 생성한다. Schedule 객체를 생성할 때, 인자로 넣은 String을 받아 split하여 Title, startingTime, endingTime, Memo 변수에 값을 저장한다. 그렇게 생성된 여러 개의 Schedule 객체를 ArrayList<Schedule>에 저장한다. 이 때, 잘못된 정보를 검출하는 것은 과제3에서 구현해본 적이 있으므로, 정상입력만 받는다고 가정한다.
Schedule Planner GUI는 MonthFrame 클래스로 만들고, Day Schedule GUI는 DayFrame 클래스로 만든다. MonthFrame에는 화살표 버튼 2개와 각각의 날짜 버튼에서 actionlistener를 사용해야 하고, DayFrame에는 cancel, add, save 3개의 버튼에서 actionlistener를 사용해야 한다.
처음 실행을 하면, 오늘이 속한 달의 달력과 오늘의 스케줄이 표시된다. 달력에서 왼쪽 화살표 버튼을 누르면 이전 달로 넘어가고, 오른쪽 화살표 버튼을 누르면 다음 달로 넘어간다. 날짜 버튼을 누르면 누른 날짜의 스케줄이 표시되고, cancel 버튼으로 스케줄을 지우거나, add, save 버튼으로 새로운 스케줄을 기록할 수 있다. 바뀐 모든 스케줄 정보는 메모장에 기록한다.

2. 코드 설명

(1) Schedule Class

```
public class Schedule {

    String title, memo;
    LocalDateTime startingTime, endingTime;

    private static DateTimeFormatter form = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");

    private Schedule() {

        System.out.println("Empty Schedule");

    }

    Schedule(String s) {

        String[] tokens = s.split("//");
        title = tokens[0];
        startingTime = LocalDateTime.parse(tokens[1], form);
        endingTime = LocalDateTime.parse(tokens[2], form);
        if(tokens.length == 4) memo = tokens[3];
        else memo = "";

    }

}
```

- String형 변수 title과 memo 선언
- LocalDateTime형 변수 startingTime과 endingTime 선언
- String형을 LocalDateTime형으로 변환할 때, 필요한 DateTimeFormatter 선언
- 객체를 생성할 때, String형 인자가 없으면 “Empty Schedule” 출력
- 객체를 생성할 때, String형 인자가 있으면 “//”로 split하여 각각의 변수에 저장

(2) ScheduleList Class

```
public class ScheduleList{

    ArrayList<Schedule> schedulelist = new ArrayList<>();

    public ScheduleList() {

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

    }

    ScheduleList(String fileName){

        try {
            File file = new File(fileName);
            Scanner scan = new Scanner(file);
            String line;
            while(scan.hasNext()) {
                line = scan.nextLine();
                if(line.isEmpty() || line.startsWith("/")) continue;
                else {
                    Schedule schedule = new Schedule(line);
                    schedulelist.add(schedule);
                }
            }
            scan.close();
        } catch (Exception e) {
            System.out.println("Unknown File");
        }

    }

}
```

- Schedule형으로 정보를 저장하는 ArrayList 생성
- 객체를 생성할 때, 파일이름 인자가 없으면 “Unknown File” 출력
- 객체를 생성할 때, 파일이름 인자가 있으면 File과 Scanner를 이용하여 파일에서 정보를 읽어와서 Schedule 객체를 생성
- 한 줄씩 읽으면서 Schedule 객체를 생성하여 ArrayList에 차례대로 저장
- 파일을 읽는 중 오류가 생기면 “Unknown File” 출력

(3) MonthFrame Class

```
public class MonthFrame extends JFrame {

    int year, month;
    int day, len, i, j;
    private LocalDate f_date;
    private JButton btn;
    private JButton lArrow = new JButton("<-");
    private JButton rArrow = new JButton("->");
    private JLabel date = new JLabel();
    private JLabel sun = new JLabel("SUN");
    private JLabel mon = new JLabel("MON");
    private JLabel tue = new JLabel("TUE");
    private JLabel wed = new JLabel("WED");
    private JLabel thu = new JLabel("THU");
    private JLabel fri = new JLabel("FRI");
    private JLabel sat = new JLabel("SAT");
    MyListener1 listener1 = new MyListener1();
    MyListener2 listener2 = new MyListener2();
    MyListener3 listener3 = new MyListener3();
```

=> MonthFrame에서 사용할 변수를 한 번에 미리 선언

```
    public MonthFrame() {

        this(LocalDate.now().getYear(), LocalDate.now().getMonthValue());

    }

    public MonthFrame(int y, int m) {

        super("Schedule Planner");
        setSize(500, 600);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        year = y; month = m;

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(2, 1));
        panel.add(new MonthPanel1());
        panel.add(new MonthPanel2());

        add(panel, BorderLayout.NORTH);
        add(new MonthPanel3(), BorderLayout.CENTER);

    }
```

- 객체를 생성할 때, 인자가 없으면 오늘이 포함된 달의 달력을 출력
- 객체를 생성할 때, 인자가 있으면 매개변수로 받은 값으로 그 달을 출력
- Frame 큰 틀을 만들고, Border Layout으로 지정
- 아래에서 만들 클래스들의 객체를 생성하여 panel을 불러 frame의 Border 각 위치에 삽입

```
    private class MonthPanel1 extends JPanel {
        public MonthPanel1() {

            setLayout(new GridLayout(1, 3));
            date.setText(Integer.toString(year)+"년 "+Integer.toString(month)+"월");
            date.setHorizontalAlignment(JLabel.CENTER);
            lArrow.addActionListener(listener1);
            rArrow.addActionListener(listener2);

            add(lArrow);
            add(date);
            add(rArrow);

        }
    }
```

- JButton X 2
- JLabel X 1
- 라벨 가운데 맞춤
- 버튼에 actionlistener 추가
- panel에 JButton과 JLabel을 추가

```
private class MonthPanel2 extends JPanel {
    public MonthPanel2() {

        setLayout(new GridLayout(1, 7));
        sun.setHorizontalAlignment(JLabel.CENTER);
        mon.setHorizontalAlignment(JLabel.CENTER);
        tue.setHorizontalAlignment(JLabel.CENTER);
        wed.setHorizontalAlignment(JLabel.CENTER);
        thu.setHorizontalAlignment(JLabel.CENTER);
        fri.setHorizontalAlignment(JLabel.CENTER);
        sat.setHorizontalAlignment(JLabel.CENTER);

        add(sun);
        add(mon);
        add(tue);
        add(wed);
        add(thu);
        add(fri);
        add(sat);

    }
}
```

- JLabel X 7
- 라벨 가운데 맞춤
- panel에 JLabel 추가

```
private class MonthPanel3 extends JPanel {
    public MonthPanel3() {

        f_date = LocalDate.of(year, month, 01);
        day = f_date.getDayOfWeek().getValue() % 7;
        len = f_date.lengthOfMonth();

        setLayout(new GridLayout(0, 7));
        for(i=0;i<day;i++) {
            add(new JLabel());
        }
        for(i=0;i<len;i++) {
            btn = new JButton(Integer.toString(i+1));
            btn.addActionListener(listener3);
            add(btn);
        }

    }
}
```

- f_date = 해당하는 달의 1일
- day = f_date의 요일
- len = 해당하는 달의 날짜 수
- 1일의 요일에 맞게 빈칸 띄우기
- len 수 만큼 JButton 생성하여 actionlistener 추가
- panel에 Jbutton 추가

```
private class MyListener1 implements ActionListener{
    public void actionPerformed(ActionEvent e) {
        if(month!=1) {
            month--;
        }
        else {
            year--;
            month = 12;
        }
        setVisible(false);
        new MonthFrame(year, month);
    }
}

private class MyListener2 implements ActionListener{
    public void actionPerformed(ActionEvent e) {
        if(month!=12) {
            month++;
        }
        else {
            year++;
            month = 1;
        }
        setVisible(false);
        new MonthFrame(year, month);
    }
}
```

- 왼쪽 화살표 버튼 -> MyListener1
- 이전 달의 year, month로 값 변경
- 새로운 MonthFrame 객체 생성

- 오른쪽 화살표 버튼 -> MyListener2
- 다음 달의 year, month로 값 변경
- 새로운 MonthFrame 객체 생성

```
private class MyListener3 implements ActionListener{
    public void actionPerformed(ActionEvent e) {
        j = Integer.parseInt(e.getActionCommand());
        new DayFrame(LocalDate.of(year, month, j), 0);
    }
}

}
```

- 달력의 날짜 버튼 -> MyListener3
- 버튼 값 반환받기
- 그 날의 스케줄을 나타내는 DayFrame 객체 생성

(4) DayFrame Class

```
public class DayFrame extends JFrame {

    String fName = "C:/Users/김두홍/Desktop/schedule-file.data";
    File file = new File(fName);
    ScheduleList list = new ScheduleList(fName);
    ArrayList <Schedule> info = new ArrayList<>();

    int y, m, d;
    private LocalDate l_date;
    private JButton cancel, add, save;
    private JTextField title, stime, etime, memo;
    private DateTimeFormatter form = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
    MyListener1 listener1 = new MyListener1();
    MyListener2 listener2 = new MyListener2();
    MyListener3 listener3 = new MyListener3();
```

=> DayFrame에서 사용할 변수를 한 번에 미리 선언

```
    public DayFrame() {

        this(LocalDate.now(), 0);

    }

    public DayFrame(LocalDate date, int adding) {

        super("Day Schedule" + "   " + date);
        setSize(500, 300);
        setVisible(true);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLayout(new BorderLayout());
        l_date = date;
        for(int i=0;i<list.schedulelist.size();i++) {
            y = list.schedulelist.get(i).startingTime.getYear();
            m = list.schedulelist.get(i).startingTime.getMonthValue();
            d = list.schedulelist.get(i).startingTime.getDayOfMonth();
            if(date.isEqual(LocalDate.of(y, m, d))) {
                s_info.add(list.schedulelist.get(i));
            }
        }

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(0, 1));
        for(int i=0;i<s_info.size();i++) panel.add(new DayPanel2(s_info.get(i)));
        if(adding==1) panel.add(new DayPanel2());

        add(new DayPanel1(), BorderLayout.NORTH);
        add(panel, BorderLayout.CENTER);
        add(new DayPanel3(), BorderLayout.SOUTH);

    }
```

- 객체를 생성할 때, 인자가 없으면 오늘의 스케줄을 출력
- 객체를 생성할 때, 인자가 있으면 매개변수로 받은 날의 스케줄을 출력
- Frame 큰 틀을 만들고, BorderLayout으로 지정
- 아래에서 만들 클래스들의 객체를 생성하여 panel을 불러 frame의 Border 각 위치에 삽입

```
    private class DayPanel1 extends JPanel {
        public DayPanel1() {

            JLabel Title = new JLabel("Title");
            JLabel Stime = new JLabel("Start Time");
            JLabel Etime = new JLabel("End Time");
            JLabel Memo = new JLabel("Memo");

            setLayout(new GridLayout(1, 4));
            Title.setHorizontalAlignment(JLabel.CENTER);
            Stime.setHorizontalAlignment(JLabel.CENTER);
            Etime.setHorizontalAlignment(JLabel.CENTER);
            Memo.setHorizontalAlignment(JLabel.CENTER);

            add(Title);
            add(Stime);
            add(Etime);
            add(Memo);

        }
    }
```

- JLabel X 4
- 라벨 가운데 맞춤
- 버튼에 actionlistener 추가
- panel에 JLabel을 추가

```
private class DayPanel2 extends JPanel {
    public DayPanel2() {

        title = new JTextField("");
        stime = new JTextField("");
        etime = new JTextField("");
        memo = new JTextField("");

        setLayout(new GridLayout(1, 4));
        add(title);
        add(stime);
        add(etime);
        add(memo);

    }

    public DayPanel2(Schedule s) {


        title = new JTextField(s.title);
        stime = new JTextField(s.startingTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")));
        etime = new JTextField(s.endingTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")));
        memo = new JTextField(s.memo);

        setLayout(new GridLayout(1, 4));
        title.setHorizontalAlignment(JTextField.CENTER);
        stime.setHorizontalAlignment(JTextField.CENTER);
        etime.setHorizontalAlignment(JTextField.CENTER);
        memo.setHorizontalAlignment(JTextField.CENTER);

        add(title);
        add(stime);
        add(etime);
        add(memo);

    }
}
```

- 객체를 생성할 때, 인자가 없으면 비어있는 TextField 4개 생성하여 추가
- 객체를 생성할 때, Schedule형 인자가 있으면 Schedule의 정보를 TextField 4개 안에 넣고 panel에 추가
- 라벨 가운데 맞춤

```
private class DayPanel3 extends JPanel {
    public DayPanel3() {

        cancel = new JButton("Cancel");
        add = new JButton("Add");
        save = new JButton("Save");

        setLayout(new GridLayout(1, 3));

        cancel.addActionListener(listener1);
        add.addActionListener(listener2);
        save.addActionListener(listener3);

        add(cancel);
        add(add);
        add(save);

    }

}
```

- JButton X 3
- 버튼에 actionlistener 추가
- panel에 JButton 추가

```
private class MyListener1 implements ActionListener{
    public void actionPerformed(ActionEvent e) {
        setVisible(false);
        list.schedulelist.removeAll(info);
        try {
            BufferedWriter f = new BufferedWriter(new FileWriter(fName));
        }catch (Exception e1) {
            System.out.println("Delete Error");
        }
        try (FileWriter fw = new FileWriter(fName, true);
            BufferedWriter bw = new BufferedWriter(fw);)
        {
            for(int i=0;i<list.schedulelist.size();i++) {
                bw.write(list.schedulelist.get(i).title + "//");
                bw.write(list.schedulelist.get(i).startingTime.format(form) + "//");
                bw.write(list.schedulelist.get(i).endingTime.format(form) + "//");
                bw.write(list.schedulelist.get(i).memo);
                bw.newLine();
            }
        }catch (Exception e2) {
            System.out.println("Write Error");
        }
        new DayFrame(l_date, 0);
    }
}
```

- cancel 버튼 -> MyListener1
- 그 날의 schedule을 list에서 삭제
- 원래의 파일에 schedulelist의 Schedule들로 업데이트
- 파일 쓰는 중에 오류가 나면 “Write Error” 출력
- 새로운 DayFrame 객체 생성

```
private class MyListener2 implements ActionListener{
    public void actionPerformed(ActionEvent e) {
        setVisible(false);
        new DayFrame(l_date, 1);
    }
}

private class MyListener3 implements ActionListener{
    public void actionPerformed(ActionEvent e) {
        setVisible(false);
        try (FileWriter fw = new FileWriter(fName, true);
                BufferedWriter bw = new BufferedWriter(fw);)
        {
            bw.newLine();
            bw.write(title.getText() + "//");
            bw.write(stime.getText() + "//");
            bw.write(etime.getText() + "//");
            bw.write(memo.getText());
        }catch (Exception e2) {
            System.out.println("Write Error");
        }
        new DayFrame(l_date, 0);
    }
}
```

- add 버튼 -> MyListener2
- 새로운 DayFrame 객체 생성
- 빈 TextField 추가 생성
- save 버튼 -> MyListener3
- TextField의 정보를 받아옴
- 원래의 파일에 schedulelist의 Schedule들로 업데이트
- 파일 쓰는 중에 오류가 나면 “Write Error” 출력
- 새로운 DayFrame 객체 생성

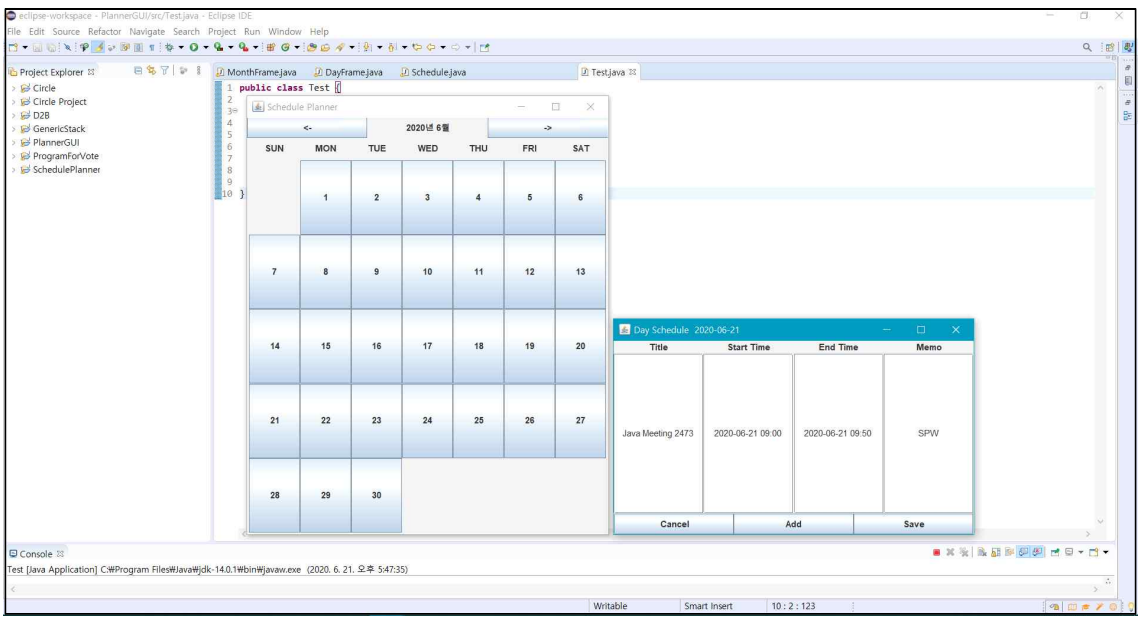
(5) Test Class (메인 메소드)

```
public class Test {

    public static void main(String[] args) {

        new MonthFrame();
        new DayFrame();

    }

}
```

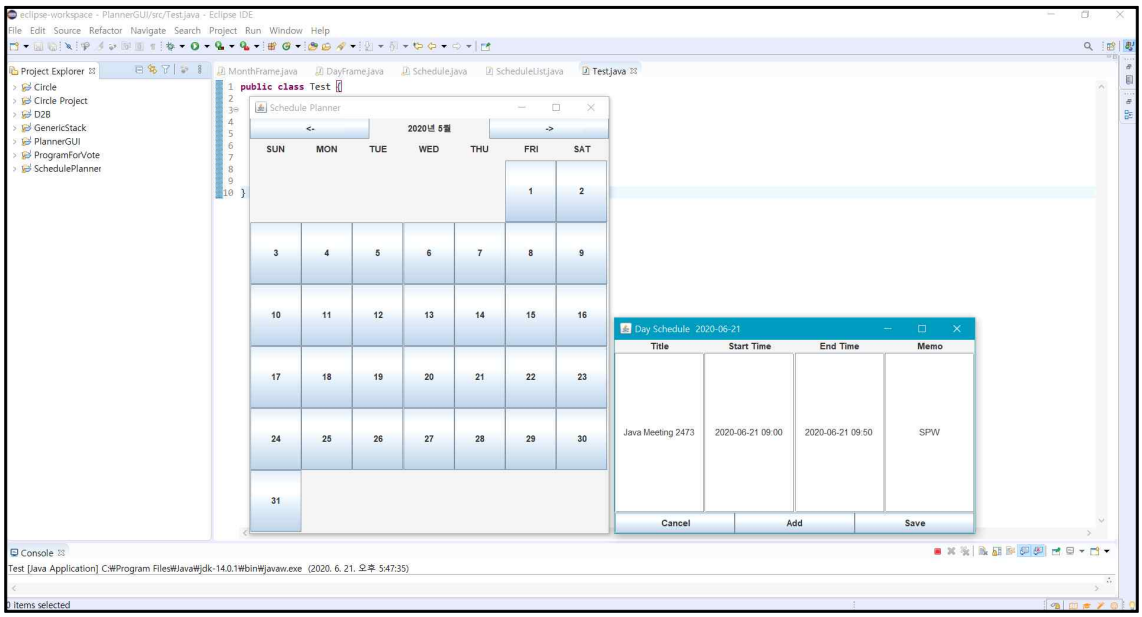
- MonthFrame 객체 생성
- DayFrame 객체 생성

3. 결과 캡처

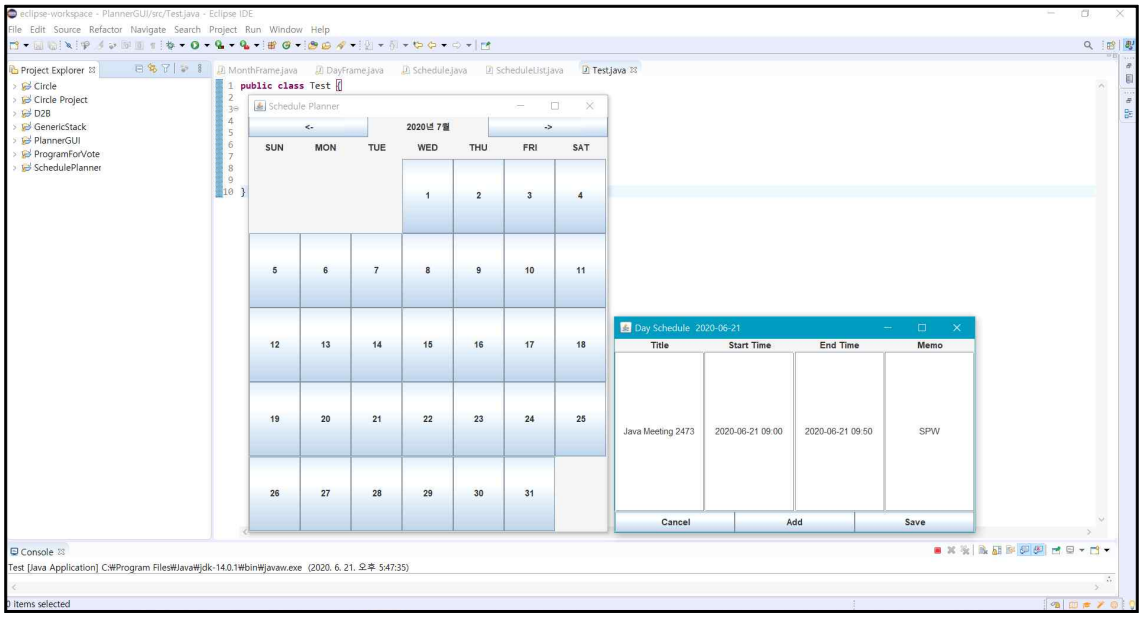
(1) 기본 화면



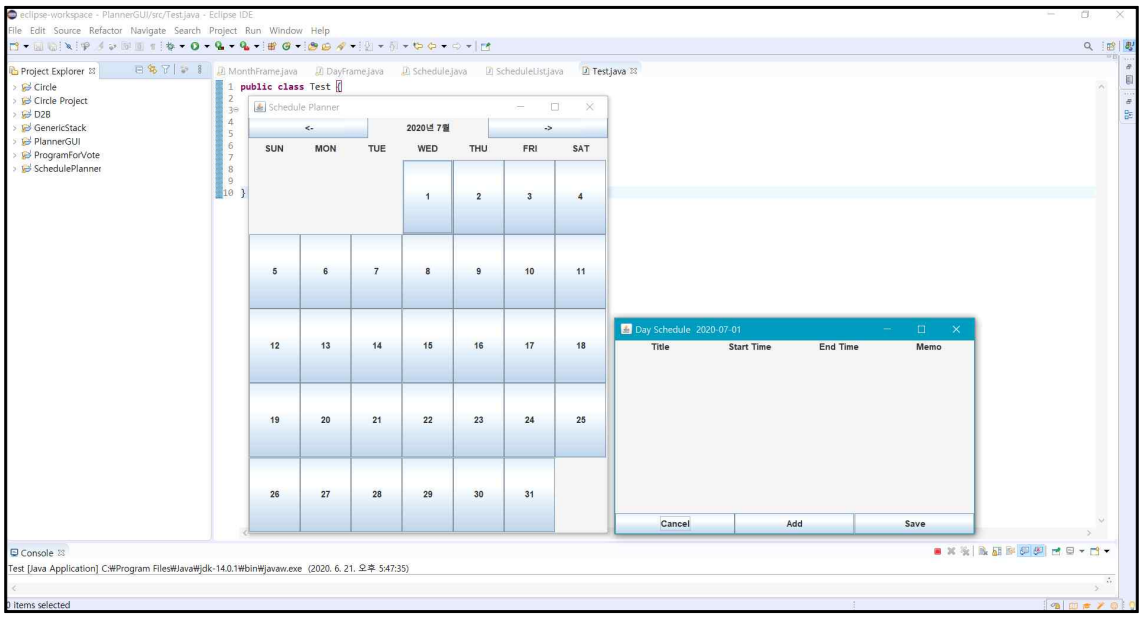
(2-1) 왼쪽 화살표 버튼 클릭



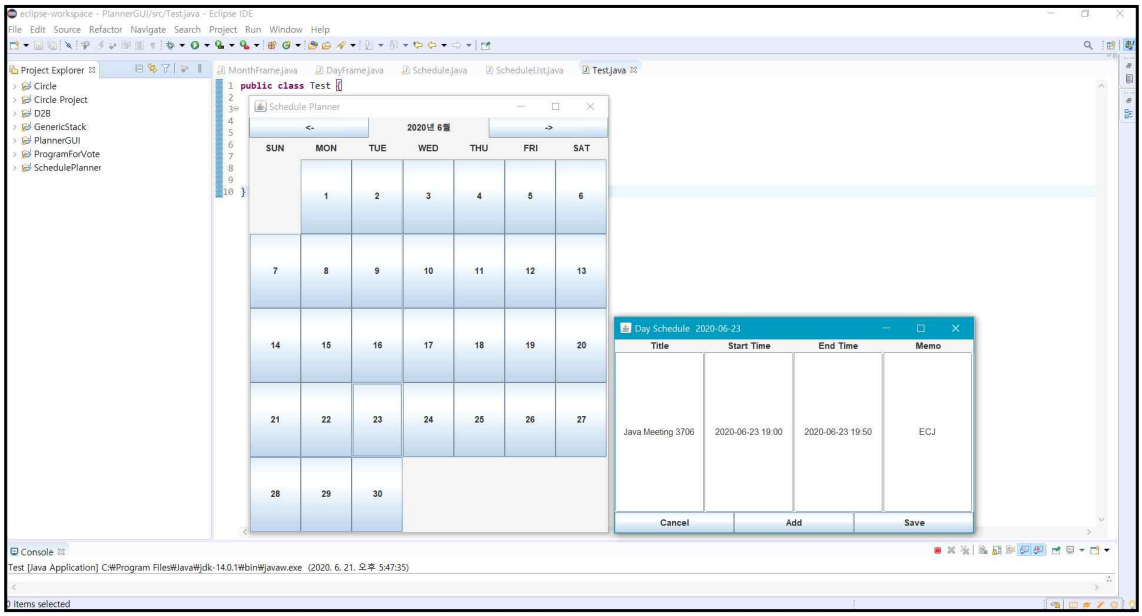
(2-2) 오른쪽 화살표 버튼 클릭



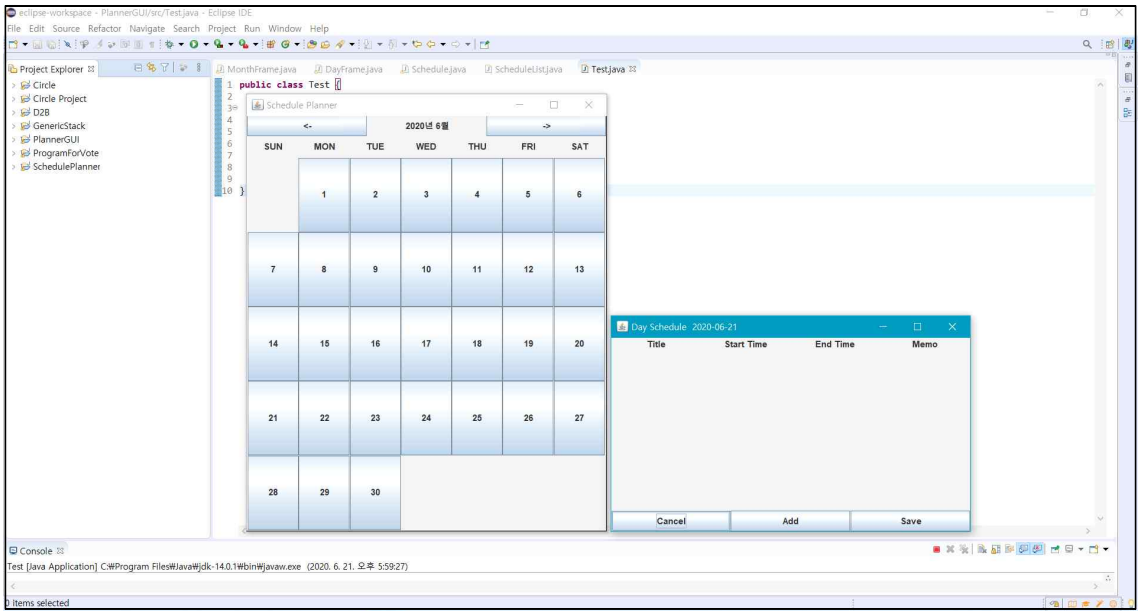
(2-3) 날짜 버튼 클릭 (스케줄 없음)



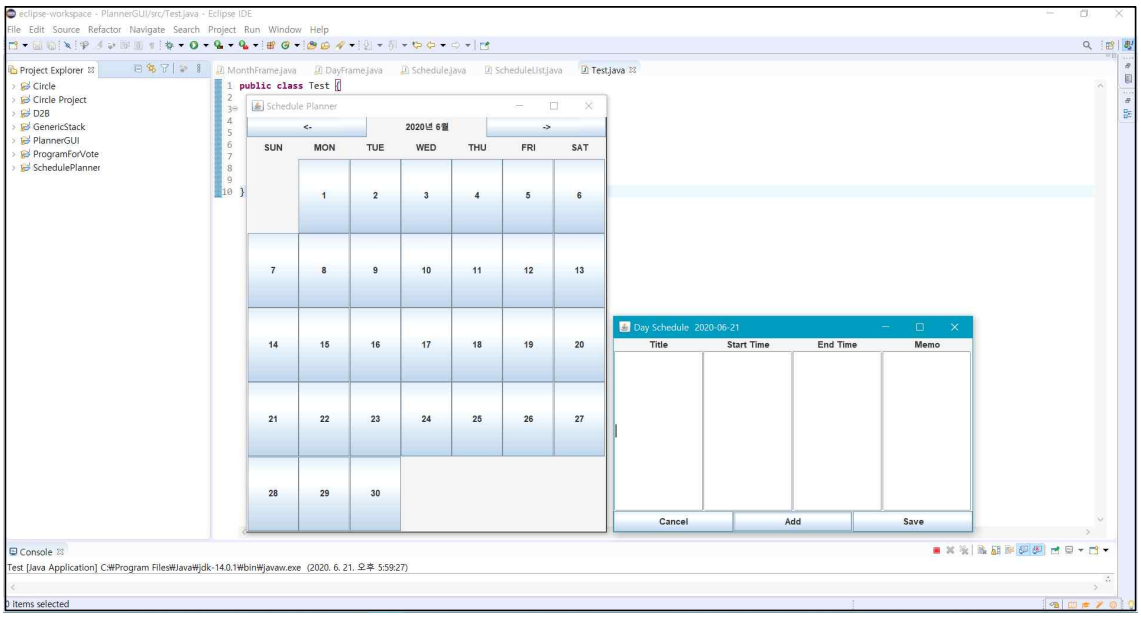
(2-4) 날짜 버튼 클릭 (스케줄 있음)



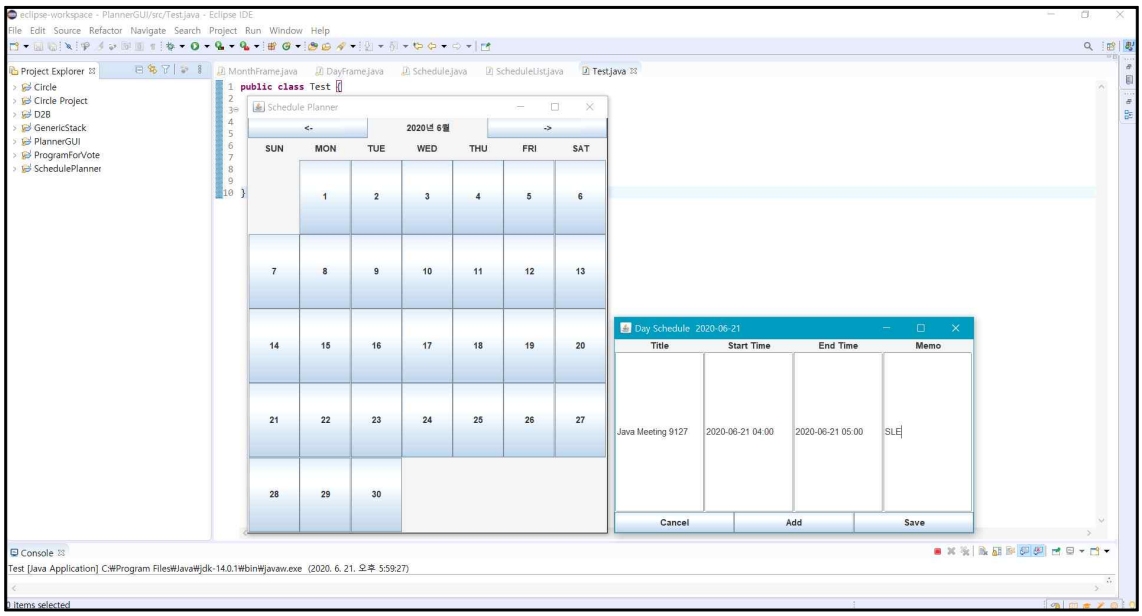
(3) Cancel 버튼 클릭



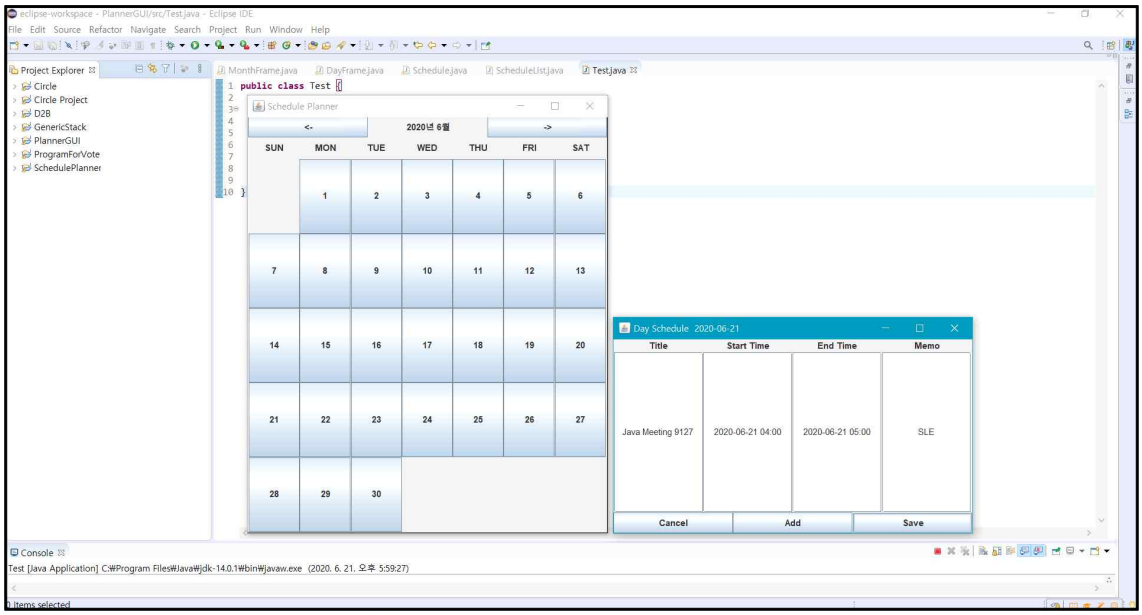
(4) Add 버튼 클릭



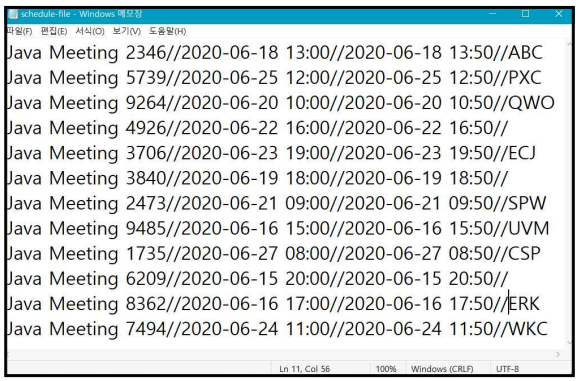
(5-1) 저장할 스케줄 입력하기



(5-2) Save 버튼 클릭하기



(6) 사용한 schedule-file.data



4. 소스 코드

```
public class Test {
	public static void main(String[] args) {
		new MonthFrame();
		new DayFrame();
	}
}
```

```
public class Schedule {
	String title, memo;
	LocalDateTime startingTime, endingTime;
	private static DateTimeFormatter form = DateTimeFormatter.ofPattern("y
yyy-MM-dd HH:mm");
	private Schedule() {
		System.out.println("Empty Schedule");
	}
	Schedule(String s) {
		String[] tokens = s.split("//");
		title = tokens[0];
		startingTime = LocalDateTime.parse(tokens[1], form);
		endingTime = LocalDateTime.parse(tokens[2], form);
		if(tokens.length == 4) memo = tokens[3];
		else memo = "";
	}
}
```

```
public class ScheduleList {
	ArrayList<Schedule> schedulelist = new ArrayList<>();
	public ScheduleList() {
		System.out.println("Unknown File");
	}
	ScheduleList(String fileName){
		try {
			File file = new File(fileName);
			Scanner scan = new Scanner(file);
			String line;
			while(scan.hasNext()) {
				line = scan.nextLine();
				if(line.isEmpty() || line.startsWith("/")) continue;
				else {
					Schedule schedule = new Schedule(line);
					schedulelist.add(schedule);
				}
			}
			scan.close();
		} catch (Exception e) {
			System.out.println("Unknown File");
		}
	}
}
```

```
public class MonthFrame extends JFrame {
	int year, month;
	int day, len, i, j;
	private LocalDate f_date;
	private JButton btn;
	private JButton lArrow = new JButton("<-");
	private JButton rArrow = new JButton("->");
	private JLabel date = new JLabel();
	private JLabel sun = new JLabel("SUN");
	private JLabel mon = new JLabel("MON");
	private JLabel tue = new JLabel("TUE");
	private JLabel wed = new JLabel("WED");
```

```
	private JLabel thu = new JLabel("THU");
	private JLabel fri = new JLabel("FRI");
	private JLabel sat = new JLabel("SAT");
	MyListener1 listener1 = new MyListener1();
	MyListener2 listener2 = new MyListener2();
	MyListener3 listener3 = new MyListener3();
	public MonthFrame() {
		this(LocalDate.now().getYear(),	LocalDate.now().getMonthValue
());
	}
	public MonthFrame(int y, int m) {
		super("Schedule Planner");
		setSize(500, 600);
		setVisible(true);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setLayout(new BorderLayout());
		year = y; month = m;
		JPanel panel = new JPanel();
		panel.setLayout(new GridLayout(2, 1));
		panel.add(new MonthPanel1());
		panel.add(new MonthPanel2());
		add(panel, BorderLayout.NORTH);
		add(new MonthPanel3(), BorderLayout.CENTER);
	}
	private class MonthPanel1 extends JPanel {
		public MonthPanel1() {
			setLayout(new GridLayout(1, 3));
			date.setText(Integer.toString(year)+"년  "+Integer.toString(m
onth)+"월");
			date.setHorizontalAlignment(JLabel.CENTER);
			lArrow.addActionListener(listener1);
			rArrow.addActionListener(listener2);
			add(lArrow);
			add(date);
			add(rArrow);
		}
	}
	private class MonthPanel2 extends JPanel {
```

```
	public MonthPanel2() {
		setLayout(new GridLayout(1, 7));
		sun.setHorizontalAlignment(JLabel.CENTER);
		mon.setHorizontalAlignment(JLabel.CENTER);
		tue.setHorizontalAlignment(JLabel.CENTER);
		wed.setHorizontalAlignment(JLabel.CENTER);
		thu.setHorizontalAlignment(JLabel.CENTER);
		fri.setHorizontalAlignment(JLabel.CENTER);
		sat.setHorizontalAlignment(JLabel.CENTER);
		add(sun);
		add(mon);
		add(tue);
		add(wed);
		add(thu);
		add(fri);
		add(sat);
	}
}
private class MonthPanel3 extends JPanel {
	public MonthPanel3() {
		f_date = LocalDate.of(year, month, 01);
		day = f_date.getDayOfWeek().getValue() % 7;
		len = f_date.lengthOfMonth();
		setLayout(new GridLayout(0, 7));
		for(i=0;i<day;i++) {
			add(new JLabel());
		}
		for(i=0;i<len;i++) {
			btn = new JButton(Integer.toString(i+1));
			btn.addActionListener(listener3);
			add(btn);
		}
	}
}
private class MyListener1 implements ActionListener{
	public void actionPerformed(ActionEvent e) {
		if(month!=1) {
			month--;
```

```
                }
                else {
                        year--;
                        month = 12;
                }
                setVisible(false);
                new MonthFrame(year, month);
        }
    }
    private class MyListener2 implements ActionListener{
        public void actionPerformed(ActionEvent e) {
                if(month!=12) {
                        month++;
                }
                else {
                        year++;
                        month = 1;
                }
                setVisible(false);
                new MonthFrame(year, month);
        }
    }
    private class MyListener3 implements ActionListener{
        public void actionPerformed(ActionEvent e) {
                j = Integer.parseInt(e.getActionCommand());
                new DayFrame(LocalDate.of(year, month, j), 0);
        }
    }
}
```

```
public class DayFrame extends JFrame {
    String fName = "C:/Users/김두홍/Desktop/schedule-file.data";
    File file = new File(fName);
    ScheduleList list = new ScheduleList(fName);
    ArrayList <Schedule> info = new ArrayList<>();
    int y, m, d;
    private LocalDate l_date;
    private JButton cancel, add, save;
```

```
	private JTextField title, stime, etime, memo;
	private DateTimeFormatter form = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
	MyListener1 listener1 = new MyListener1();
	MyListener2 listener2 = new MyListener2();
	MyListener3 listener3 = new MyListener3();
	public DayFrame() {
		this(LocalDate.now(), 0);
	}
	public DayFrame(LocalDate date, int adding) {
		super("Day Schedule" + "   " + date);
		setSize(500, 300);
		setVisible(true);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setLayout(new BorderLayout());
		l_date = date;
		for(int i=0;i<list.schedulelist.size();i++) {
			y = list.schedulelist.get(i).startingTime.getYear();
			m = list.schedulelist.get(i).startingTime.getMonthValue();
			d = list.schedulelist.get(i).startingTime.getDayOfMonth();
			if(date.isEqual(LocalDate.of(y, m, d))) {
				info.add(list.schedulelist.get(i));
			}
		}
		JPanel panel = new JPanel();
		panel.setLayout(new GridLayout(0, 1));
		for(int i=0;i<info.size();i++) panel.add(new DayPanel2(info.get(i)));
		if(adding==1) panel.add(new DayPanel2());
		add(new DayPanel1(), BorderLayout.NORTH);
		add(panel, BorderLayout.CENTER);
		add(new DayPanel3(), BorderLayout.SOUTH);
	}
	private class DayPanel1 extends JPanel {
		public DayPanel1() {
			JLabel Title = new JLabel("Title");
			JLabel Stime = new JLabel("Start Time");
			JLabel Etime = new JLabel("End Time");
			JLabel Memo = new JLabel("Memo");
```

```
                setLayout(new GridLayout(1, 4));
                Title.setHorizontalAlignment(JLabel.CENTER);
                Stime.setHorizontalAlignment(JLabel.CENTER);
                Etime.setHorizontalAlignment(JLabel.CENTER);
                Memo.setHorizontalAlignment(JLabel.CENTER);
                add(Title);
                add(Stime);
                add(Etime);
                add(Memo);
            }
        }
        private class DayPanel2 extends JPanel {
            public DayPanel2() {
                title = new JTextField("");
                stime = new JTextField("");
                etime = new JTextField("");
                memo = new JTextField("");
                setLayout(new GridLayout(1, 4));
                add(title);
                add(stime);
                add(etime);
                add(memo);
            }
            public DayPanel2(Schedule s) {
                title = new JTextField(s.title);
                stime = new JTextField(s.startingTime.format(DateTimeFo
rmatter.ofPattern("yyyy-MM-dd HH:mm")));
                etime = new JTextField(s.endingTime.format(DateTimeFor
matter.ofPattern("yyyy-MM-dd HH:mm")));
                memo = new JTextField(s.memo);
                setLayout(new GridLayout(1, 4));
                title.setHorizontalAlignment(JTextField.CENTER);
                stime.setHorizontalAlignment(JTextField.CENTER);
                etime.setHorizontalAlignment(JTextField.CENTER);
                memo.setHorizontalAlignment(JTextField.CENTER);
                add(title);
                add(stime);
                add(etime);
```

```
                add(memo);
            }
        }
        private class DayPanel3 extends JPanel {
            public DayPanel3() {
                cancel = new JButton("Cancel");
                add = new JButton("Add");
                save = new JButton("Save");
                setLayout(new GridLayout(1, 3));
                cancel.addActionListener(listener1);
                add.addActionListener(listener2);
                save.addActionListener(listener3);
                add(cancel);
                add(add);
                add(save);
            }
        }
        private class MyListener1 implements ActionListener{
            public void actionPerformed(ActionEvent e) {
                setVisible(false);
                list.schedulelist.removeAll(info);
                try {
                    BufferedWriter f = new BufferedWriter(new FileWrit
er(fName));
                }catch (Exception e1) {
                    System.out.println("Delete Error");
                }
                try (FileWriter fw = new FileWriter(fName, true);
                    BufferedWriter bw = new BufferedWriter(fw);)
                {
                    for(int i=0;i<list.schedulelist.size();i++) {
                        bw.write(list.schedulelist.get(i).title + "//");
                        bw.write(list.schedulelist.get(i).startingTim
e.format(form) + "//");
                        bw.write(list.schedulelist.get(i).endingTime.
format(form) + "//");
                        bw.write(list.schedulelist.get(i).memo);
                        bw.newLine();
```

```
                        }
                    }catch (Exception e2) {
                        System.out.println("Write Error");
                    }
                    new DayFrame(l_date, 0);
                }
            }
            private class MyListener2 implements ActionListener{
                public void actionPerformed(ActionEvent e) {
                    setVisible(false);
                    new DayFrame(l_date, 1);
                }
            }
            private class MyListener3 implements ActionListener{
                public void actionPerformed(ActionEvent e) {
                    setVisible(false);
                    try (FileWriter fw = new FileWriter(fName, true);
                            BufferedWriter bw = new BufferedWriter(f
w);)
                    {
                        bw.newLine();
                        bw.write(title.getText() + "//");
                        bw.write(stime.getText() + "//");
                        bw.write(etime.getText() + "//");
                        bw.write(memo.getText());
                    }catch (Exception e2) {
                        System.out.println("Write Error");
                    }
                    new DayFrame(l_date, 0);
                }
            }
}
```

5. 평가표

평가 항목	학생 자체 평가	평가	점수
완성도 (동작여부) - 초기 동작 - Add 동작 - Cancel 동작 - Save 동작 - 기타 비정상 동작 실험	초기 동작, Add 동작, Cancel 동작, Save 동작 모두 잘 실행되었다. 하지만 Add 후, Save 할 때는 제대로 잘 실행되었지만, 기존에 있는 TextField를 수정하여 Save 하는 경우에는 오류가 발생하기도 했다.		
설계 노트 - 주요 결정사항 및 근거 - 한계/문제점 - 해결 방안	설계 노트를 작성하였다. GUI의 구성부터 클래스 간의 구조에 대해 그림을 그렸고, 어떻게 만들지 구상을 마친 후에 프로그램을 만들 수 있었다.		
리포트 - 평가자 시각으로 리포트 검토 - 위의 평가 요소 명확한 기술	설계노트와 코드설명, 결과화면 모두 캡처하여 시각자료를 업로드 하였고, 위의 평가 요소들에 대해 자세한 설명을 덧붙였다.		
총평 / 계	강의시간에 배운 것들을 활용하여 GUI로 프로그램을 만들어보았다. 단순 암기가 아니라 활용하는 것이 많아서 처음 배우는 나에게 너무 어려웠지만, 최선을 다해서 결과물을 거의 완성시켰다.		