

소프트웨어프로젝트

프로젝트6 레포트

일정 관리 프로그램 최종 설계 및 구현

소프트웨어프로젝트 02분반
월5 수 5, 6교시 박창윤 교수님
서동혁 조교님

소프트웨어학부
20190323 배인경

<1> 완성도(동작 여부 ; 기능)

* 파란 글씨는 조건 만족을 위해 고려해 준 부분 or 코드를 짜며 생각해 본 것들

< 1-1 > 초기 동작 ① 실행 시 첫 GUI화면 + txt파일 + 콘솔창 출력 내용

The screenshot shows an IDE with a project named 'SoftwareProject'. The main class is 'Main', which contains a 'main' method. The 'main' method creates a 'MonthSchedule' object and sets its visibility to true. The 'MonthSchedule' GUI window is displayed, showing a calendar for June 2020. The console window shows the following output:

```

C:\eclipse-java-2020-03-R-win32-x86_64\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2020.
No Schedule Title ; Skip the input line : //2020-06-01 18:00//2020-06-01 18:00
Start Time is later than End Time ; Skip the input line : 6/1일정3//2020-06-01 20:00//2020-06-01 19:00
  
```

- 1) 바탕화면에 미리 저장 되어 있는 파일은 '202005', '202006'파일임
 그 중 초기 모습을 보이기 위해 '202006' txt파일의 초기 정보를 보임
 * 주요 결정 사항) Schedule의 해당 년, 월에 맞게 초기 정보를 해당 파일에 input해 넣는다고 가정하고 '년+월'로 txt파일을 각각 따로 만들어 줌.
 * 근거) 처음에는 모든 정보가 한 파일에 들어가 있다고 가정하고 코드를 짰더니 save과정에서 생성되어 있는 년+월+일자들에 대한 모든 daySchedule 객체의 값을 비교해서 찾아 들어가야해서 성능적으로 굉장히 좋지 않은 코드를 짤 수 밖에 없다는 것을 알게됨.
 * 해결 과정) '년+월'로 txt파일을 각각 따로 만들면 일에 대해서만 비교하면 되어서 최대 31번만 비교하게 되어 훨씬 복잡도가 줄어들게 코드를 짤 수 있음
- 2) main함수에서 처음에 보이는 MonthSchedule GUI는 2020년 6월에 대해 나타낸다고 가정
- 3) '202006' 초기 txt파일 속 정보 중 문법에 오류가 있는 줄 색출
 - ① 4번째 줄 -> title이 없음 -> 콘솔창에 'No Schedule Title' 출력
 - ② 10번째 줄 -> Start Time이 End Time보다 늦음 -> 콘솔창에 'Start Time is later than End Time' 출력

< 1-1 > 초기 동작 ② Day버튼을 클릭했을 때 DaySchedule GUI 구현 입증

Title	StartTime	EndTime	Memos
6/1일정	2020-06-01T18:00	2020-06-01T19:00	null
6/1일정2	2020-06-01T19:00	2020-06-01T20:00	null

▲ 2020 6/1 버튼을 클릭했을 때

Title	StartTime	EndTime	Memos
6/20일정	2020-06-20T19:00	2020-06-20T20:00	null

▲ 2020 6/20 버튼을 클릭했을 때

202006 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
6/1일정//2020-06-01 18:00//2020-06-01 19:00
6/1일정2//2020-06-01 19:00//2020-06-01 20:00
;
//2020-06-01 18:00//2020-06-01 18:00
;
;
6/3일정//2020-06-03 19:00//2020-06-03 20:00
6/20일정//2020-06-20 19:00//2020-06-20 20:00
;
6/1일정3//2020-06-01 20:00//2020-06-01 19:00
```

▲ 초기 202006 파일 내용

Title	StartTime	EndTime	Memos
6/3일정	2020-06-03T19:00	2020-06-03T20:00	null

▲ 2020 6/3 버튼을 클릭했을 때

Title	StartTime	EndTime	Memos
-------	-----------	---------	-------

▲ 2020 6/25 버튼을 클릭했을 때

202006 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
6/1일정//2020-06-01 18:00//2020-06-01 19:00//null
6/1일정2//2020-06-01 19:00//2020-06-01 20:00//null
6/3일정//2020-06-03 19:00//2020-06-03 20:00//null
6/20일정//2020-06-20 19:00//2020-06-20 20:00//null
```

▲ Day Schedule GUI에서 Save 버튼을 한 번 누른 후 문법에 맞는 문장만 들어가도록 수정된 202006 파일 내용

1) 2020 6/1, 2020 6/3, 2020 6/20 파일 클릭했을 때 모두 앞서 콘솔 창에 출력되었던 문법에 맞지 않는 문장을 제외하고 Day Schedule GUI에 나타남.

2) 초기 '202006'파일에 해당 일자와 관련된 정보가 없는 2020 6/25 파일을 클릭하면 아무런 JTextField가 나타나지 않음.

< 1-1 > 초기 동작 ③ <버튼, >버튼 구현 입증 + 해당 년+월 파일 잘 찾는 지 입증

▲ 초기화면에서 <버튼을 클릭했을 때 + 콘솔창에 <버튼 눌렀음을 출력

▲ '202005' txt파일 속 초기 내용 + 콘솔창에 메모장 내용 중 문법에 오류가 있는 문장 색출하여 알림

▲ 5월화면에서 >버튼을 두 번 클릭하여 (콘솔창에 >버튼 눌렀음을 두 번 출력) 7월화면으로 이동

▲ 바탕화면에 '202007' txt파일이 처음부터 존재하지는 않음 (콘솔창에 해당 내용 출력)

1) 콘솔창에 <버튼 or >버튼을 클릭했을 때 '< Button Clicked' or '> Button Clicked'가 출력됨

2) '202005' 초기 txt파일 속 정보 중 문법에 오류가 있는 줄 색출

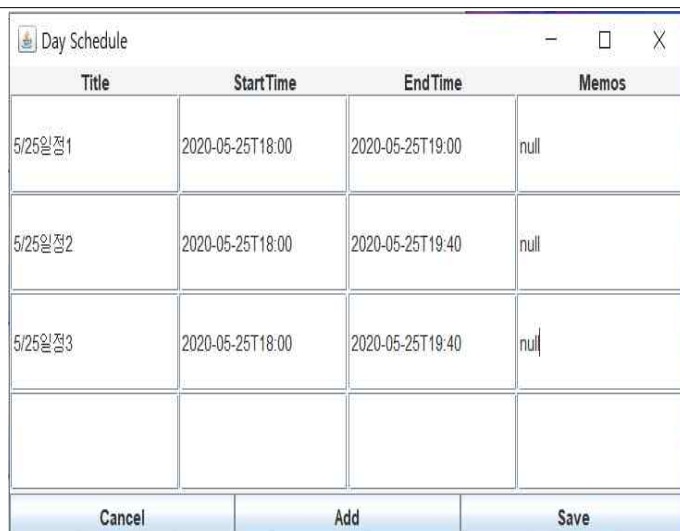
① 6번째 줄 -> title이 없음 -> 콘솔창에 'No Schedule Title' 출력

3) 7월 MonthSchedule GUI가 보일 때는 초기에 '202007' txt파일이 없었으므로 7월의 특정 일자에 DaySchedule을 Add하고 Save하면 '202007' txt파일이 새로 생기도록 구현

-> 추후에 add동작, save동작이 됨을 보여주며 입증할 예정

-> '202007' txt파일이 존재하지 않기 때문에 'Unknown File. Create New File.' 내용이 콘솔창에 출력됨

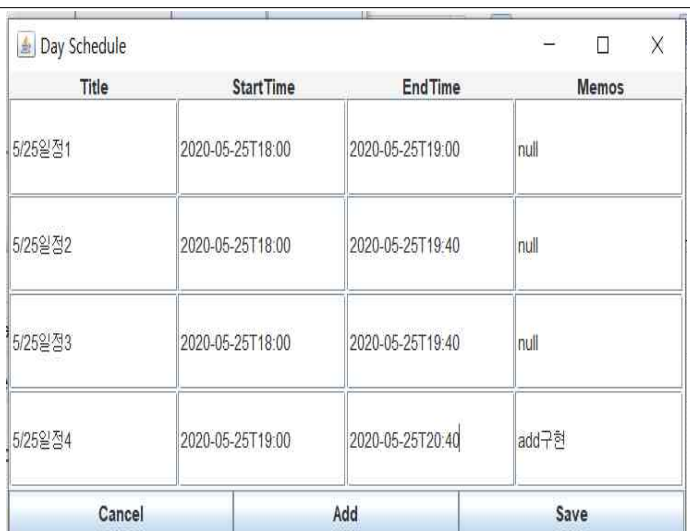
< 1-2 > Add 동작



Title	StartTime	EndTime	Memos
5/25일정1	2020-05-25T18:00	2020-05-25T19:00	null
5/25일정2	2020-05-25T18:00	2020-05-25T19:40	null
5/25일정3	2020-05-25T18:00	2020-05-25T19:40	null

Buttons: Cancel, Add, Save

▲ Add버튼 클릭했을 때 JTextField 생성 된 모습



Title	StartTime	EndTime	Memos
5/25일정1	2020-05-25T18:00	2020-05-25T19:00	null
5/25일정2	2020-05-25T18:00	2020-05-25T19:40	null
5/25일정3	2020-05-25T18:00	2020-05-25T19:40	null
5/25일정4	2020-05-25T19:00	2020-05-25T20:40	add구현

Buttons: Cancel, Add, Save

▲ 그 이후 새로운 일정을 입력한 모습

1) 2020 5월 MonthSchedule GUI에서 25일 버튼을 클릭한 후 Add 버튼을 클릭했을 때 새로운 JTextField가 생성된 모습 + 새로운 일정을 입력한 모습

< 1-3 > Cancel 동작

Title	StartTime	EndTime	Memos
6/1일정	2020-06-01T18:00	2020-06-01T19:00	null
6/1일정2	2020-06-01T19:00	2020-06-01T20:00	null

Buttons: Cancel, Add, Save

▲ 2020 6/1 Day Schedule GUI를 열었을 때 초기 모습

Title	StartTime	EndTime	Memos
6/1일정	2020-06-01T18:00	2020-06-01T19:00	null
	2020-06-01T19:00	2020-06-01T20:00	null
cancel test	2020-06-01T19:00	2020-06-01T20:00	akjele

Buttons: Cancel, Add, Save

▲ Add 버튼을 누르고 정보를 입력 및 수정한 후 Save 버튼을 누르지 않고 Cancel 버튼을 누름

Title	StartTime	EndTime	Memos
6/1일정	2020-06-01T18:00	2020-06-01T19:00	null
6/1일정2	2020-06-01T19:00	2020-06-01T20:00	null

Buttons: Cancel, Add, Save

▲ 다시 2020 6/1 Day Schedule GUI를 열었을 때 모습

1) JTextField를 add하고 정보를 입력하거나 형식에 맞지 않게 수정하는 등에 상관없이 Save를 누르지 않았기 때문에 Cancel을 누르면 원래의 상태로 돌아옴을 확인

< 1-4 > Save 동작 ① Add 후 Save가 됨을 구현

202005 - Windows 메모장	202005 - Windows 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)	파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
5/24일정1//2020-05-24 13:00//2020-05-24 13:50//Test	5/24일정1//2020-05-24 13:00//2020-05-24 13:50//Test
5/25일정1//2020-05-25 18:00//2020-05-25 19:00//	5/25일정1//2020-05-25 18:00//2020-05-25 19:00//null
5/25일정2//2020-05-25 18:00//2020-05-25 19:40//	5/25일정2//2020-05-25 18:00//2020-05-25 19:40//null
;	5/25일정3//2020-05-25 18:00//2020-05-25 19:40//null
5/25일정3//2020-05-25 18:00//2020-05-25 19:40//	5/25일정4//2020-05-25 19:00//2020-05-25 20:40//add구현
//2020-05-25 18:00//2020-05-25 19:40//dfs	
;	

▲ 초기 202005 파일 내용

▲ Day Schedule GUI에서 Save 버튼을 한 번 누른 후

- 1) ‘;’ 이나 문법에 오류가 있는 문장은 삭제되고 새로 추가 된 5/25일정4(오른쪽 txt파일 그림 속 5번째 줄)이 저장됨
- 2) 일정에서 memo가 없었을 때 -> memo란에 null이라고 표시되도록 함
(교수님 과제 설명 영상에서도 null이라고 표현되어 있는 걸 확인)
- 3) 프로젝트 3 당시 오류 문장이라고 판단하라고 주어진 조건(Title 내용이 없으면 오류, 시작시간이 종료 시간보다 늦으면 오류, 시간형식이 맞지 않으면 오류 등) 외의 형태로 입력이 잘못 된다면 Save 버튼이 클릭 되지 않게 함. -> 형식에 맞게 수정 후 Save가 눌리도록 구현
* 주요 결정 사항) 내가 구현한 코드는 ScheduleList class에서 문법 오류 체크를 하고 Schedule 정보를 List에 넣는 구조이므로 오류라고 판단하지 못할 오류 문장이 입력되면 디버깅 과정에서 오류로 인식하여 Save버튼이 클릭 안 되는 것이라고 생각함
- 4) 프로젝트 3 당시 오류 문장이라고 판단하라고 주어진 조건(Title 내용이 없으면 오류, 시작시간이 종료 시간보다 늦으면 오류, 시간 형식이 맞지 않으면 오류 등)의 형태로 입력이 된다면 Save 버튼 클릭 시 Save가 되지만 다시 해당 파일을 읽을 때 오류 문장이라고 판단함. 그러므로 다시 DaySchedule GUI를 띄우면 오류 문장은 보이지 않음

< 1-4 > Save 동작 ② 일정 제거를 위해 Title을 지우고 Save를 누르기

202005 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
5/24일정1//2020-05-24 13:00//2020-05-24 13:50//Test
5/25일정1//2020-05-25 18:00//2020-05-25 19:00//null
5/25일정2//2020-05-25 18:00//2020-05-25 19:40//null
5/25일정3//2020-05-25 18:00//2020-05-25 19:40//null
5/25일정4//2020-05-25 19:00//2020-05-25 20:40//add구현
```

202005 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
5/24일정1//2020-05-24 13:00//2020-05-24 13:50//Test
5/25일정1//2020-05-25 18:00//2020-05-25 19:00//null
5/25일정2//2020-05-25 18:00//2020-05-25 19:40//null
//2020-05-25 18:00//2020-05-25 19:40//null
5/25일정4//2020-05-25 19:00//2020-05-25 20:40//add구현
```

▲ Title 지우고 Save하기 전 '202005' txt파일 내용

Title	StartTime	EndTime	Memos
5/25일정1	2020-05-25T18:00	2020-05-25T19:00	null
5/25일정2	2020-05-25T18:00	2020-05-25T19:40	null
	2020-05-25T18:00	2020-05-25T19:40	null
5/25일정4	2020-05-25T19:00	2020-05-25T20:40	add구현

▲ Title 지우고 Save한 후 '202005' txt파일 내용

Title	StartTime	EndTime	Memos
5/25일정1	2020-05-25T18:00	2020-05-25T19:00	null
5/25일정2	2020-05-25T18:00	2020-05-25T19:40	null
5/25일정4	2020-05-25T19:00	2020-05-25T20:40	add구현

▲ 5/25일정3을 없애기 위해 Title을 지운 후 Save

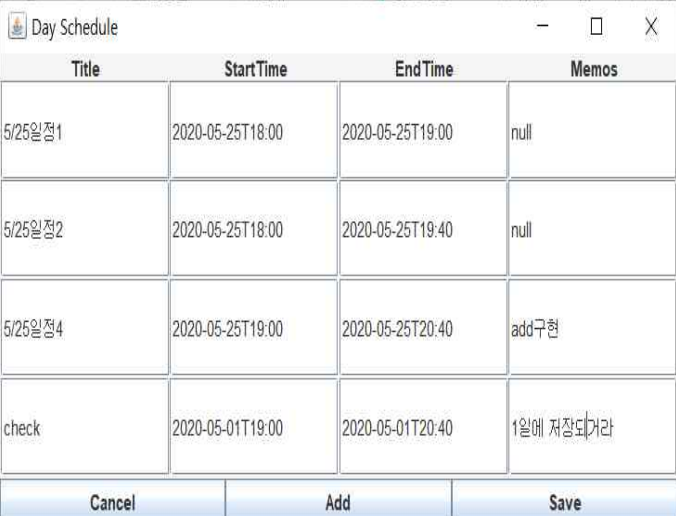
▲ 그 후에 2020 5/25 DaySchedule GUI를 연 모습

1) Title 지우고 Save 버튼을 누르면 '202005' txt파일에 '5/25일정3' 글자가 사라짐

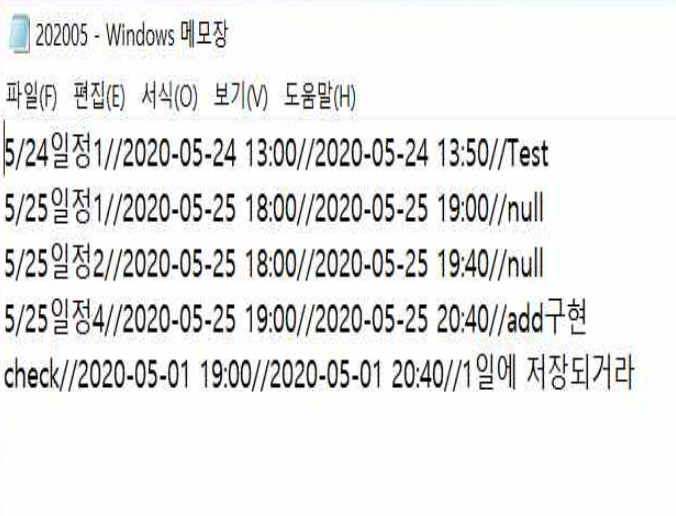
2) 그 후에 다시 2020 5/25 DaySchedule GUI를 열면 '202005' txt파일 내용을 읽는 과정에서 Title 내용이 없는 4번째 줄은 오류 문장으로 인식하기 때문에 오류 문장을 제외하고 GUI에 나타나게 됨

3) 그 후에 다시 Save를 누르면 '202005' txt파일 내용은 GUI에 보이는 내용으로 업데이트되므로 txt파일에서도 4번째 줄은 사라짐

< 1-5 > 기타 비정상 동작 ① 25일 Day Schedule GUI에서 1일에 대한 정보를 넣으면?



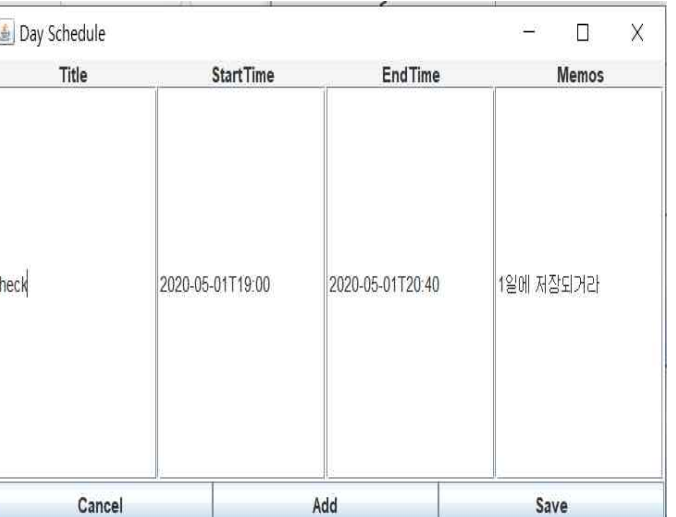
Title	StartTime	EndTime	Memos
5/25일정1	2020-05-25T18:00	2020-05-25T19:00	null
5/25일정2	2020-05-25T18:00	2020-05-25T19:40	null
5/25일정4	2020-05-25T19:00	2020-05-25T20:40	add구현
check	2020-05-01T19:00	2020-05-01T20:40	1일에 저장되거라

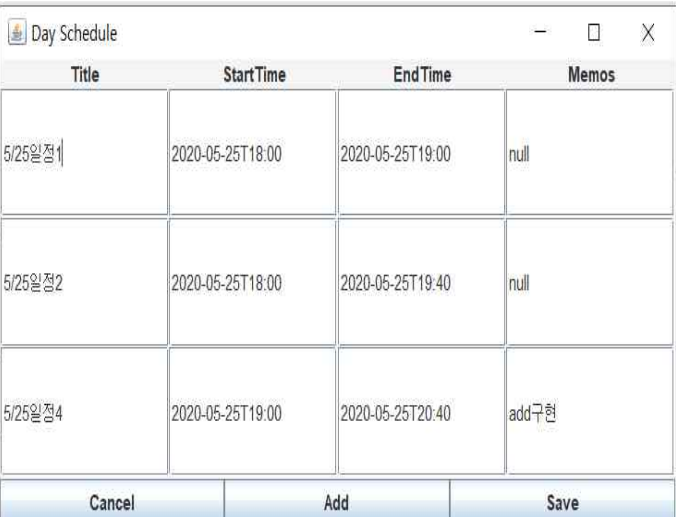


```

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
5/24일정1//2020-05-24 13:00//2020-05-24 13:50//Test
5/25일정1//2020-05-25 18:00//2020-05-25 19:00//null
5/25일정2//2020-05-25 18:00//2020-05-25 19:40//null
5/25일정4//2020-05-25 19:00//2020-05-25 20:40//add구현
check//2020-05-01 19:00//2020-05-01 20:40//1일에 저장되거라
        
```

▲ 2020 5/25 DaySchedule GUI에서 1일에 대한 정보를 넣고 Save함 + 그 후 '202005' txt파일 내용





▲ 2020 5/1 Day Schedule GUI를 연 모습

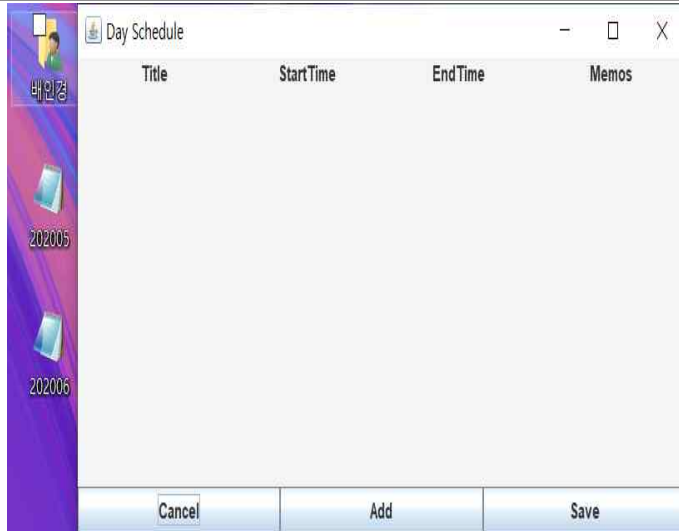
▲ 2020 5/25 Day Schedule GUI를 연 모습

1) 2020 5/25 DaySchedule GUI에서 StartTime과 EndTime에 2020 5/1에 대해 정보를 넣으면 시간 양식에는 맞으므로 오류 문장이 아님

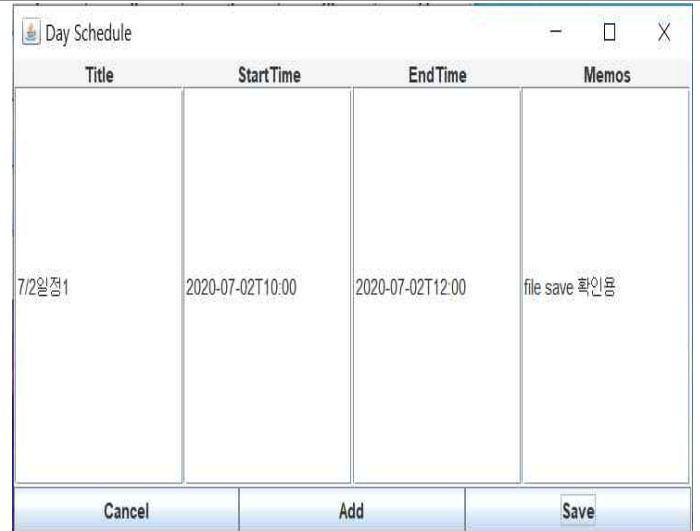
2) 그러므로 Save 버튼이 실행되고 '202005' txt파일에도 업데이트 되어 저장이 됨

3) 다시 '202005' txt파일을 읽을 때 새로 추가된 정보가 1일에 대한 정보임을 파악하기 때문에, Title이 'check'인 일정 정보가 2020 5/25 DaySchedule GUI에서 추가한 정보더라도 25일 GUI에 다시 모습을 보이지 않고 1일 GUI에 정상적으로 들어감을 확인

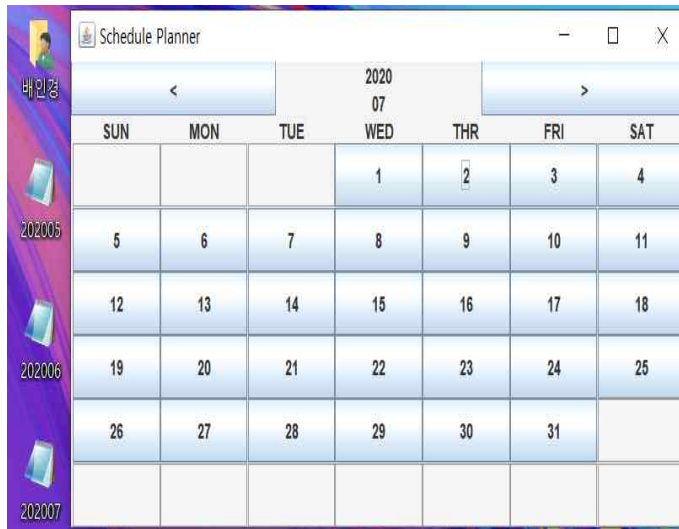
< 1-5 > 기타 비정상 동작 ② 새로운 txt 파일 생성



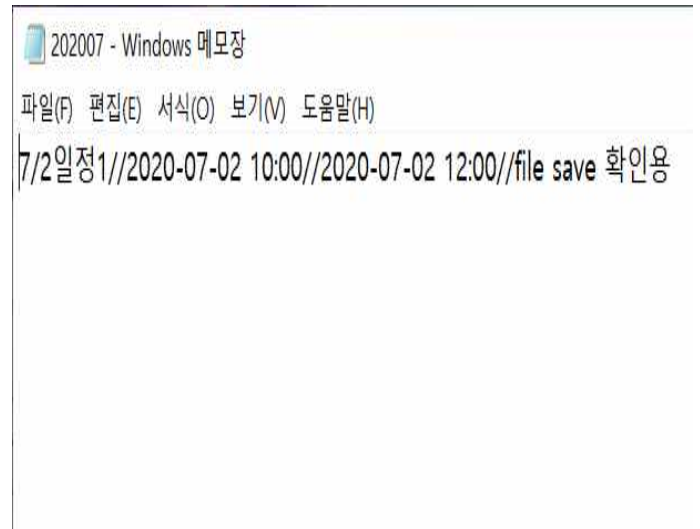
▲ add & save 전에는 '202007' txt파일이 없음



▲ 2020 7/2 DaySchedule GUI에 스케줄 add 후 save



▲ save후 바탕화면에 '202007' txt파일이 생성 됨



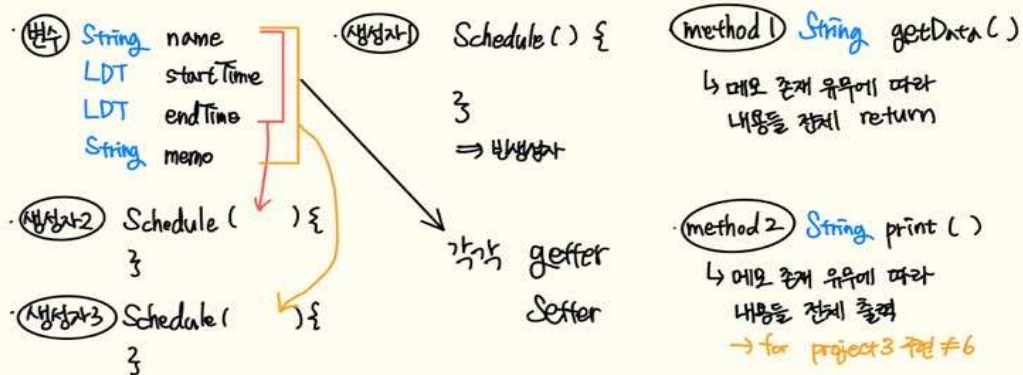
▲ '202007' txt파일 내용

1) 빈 파일이라도 '모든 년+월'에 대한 txt파일이 필요한 것이 아니라 새로운 일정이 생기는 년, 월에 대해서는 일정을 add 및 save할 때 새로운 txt파일이 생성됨을 보임

<2> 설계 노트

- * 프로그램 구성(Class 구조) -> 그림으로 표현(이미지)
- * member visibility -> 말로 작성

Schedule class



- 모든 클래스(`Schedule`, `ScheduleList`, `DaySchedule`, `MonthSchedule`) 들은 `com.company` 패키지임
- 모든 클래스(`Schedule`, `ScheduleList`, `DaySchedule`, `MonthSchedule`) 들은 `public` 선언함
- Property(변수) : 모두 클래스 외부에서 직접 접근 안되도록 `private` 선언함
- 생성자1, 2, 3 : 모두 `default` 선언함
- getter & setter : 모두 클래스 외부에서 직접 접근 되도록 `public` 선언함
- method 1, 2 : 모두 클래스 외부에서 직접 접근 되도록 `public` 선언함

ScheduleList class

· (변수) ArrayList<Schedule> array
 Writer file
 String fileName
 String pattern → LDT 패턴

· (생성자) ScheduleList(String fileName) {
 try
 ① Text → 공백 or ;로 되어있나?
 ② Text → "//" 기준으로 나누기
 ③ 예외처리 → 기타 ... else...
 catch
 → 파일 없을 때

method 1 removeSchedule(int i)
 → array.remove(i)

method 2 String getFileName()
 → fileName

method 3 int numSchedules()
 → array.size()

method 4 Schedule getSchedule(int i)
 → array.get(i)

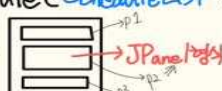
method 5 saveScheduleList(ScheduleList sList, String fileName)
 → 파일 새로 쓰기 + 저장


method 6 addSchedule(스케줄 변수 3개(+memo))
 → array.add(new Schedule(..., ..., ...))


- Property(변수) : 모두 클래스 외부에서 직접 접근 안되도록 private 선언함
- 생성자 : default 선언함
- method 1~6 : 모두 클래스 외부에서 직접 접근 되도록 public 선언함


DaySchedule class JFrame


· (변수) ArrayList<JPanel> jpList
 JPanel p2
 ArrayList<JPanel> jpBackup
 ScheduleList sl

· (생성자) DaySchedule(ScheduleList sl)
 ↳ JPanel 
 ↳ setSize
 setTitle

method 1 adding()
 ↳ JPanel p2Contents 
 ↳ jpList.add(p2Contents)

method 2 insert(Schedule S)
 ↳ JPanel p2Content 
 jpList.add(p2Content)

method 3 backupList(Schedule S)
 ↳ JPanel p2Content2 
 jpBackup.add(p2Content2)

method 3 initData()
 ↳ p2 = 
 p2.add(jpList)

inner class DayButtonListener
 ① Save
 → 기존 스케줄은 백업해두었으니 삭제
 → 수정한 JTextField 값 바탕으로 새로운 스케줄 생성해서 다시 sl에 add
 ⇒ 새로운 스케줄 + 수정한 스케줄 둘 다 '새로운 Schedule 객체'로서 들어가게 됨
 ② Add
 → removeAll
 initData()
 repaint
 ③ Cancel
 → jpList에 jpBackup 값들만 넣어주기
 → removeAll
 initData()
 repaint

- DaySchedule class는 JFrame class를 상속 받음
- Property(변수) : 모든 클래스 외부에서 직접 접근 안되도록 private 선언함
- 생성자 : default 선언함
- method 1, 2, 3, 4 : 모두 클래스 외부에서 직접 접근 되도록 public 선언함
- inner class : ActionListener class 상속 받음 / default class

Month Schedule Class JFrame

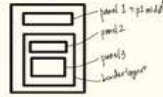
Dayschedule[] days
 LocalDate first, 1d
 ScheduleList newSche
 JButton[] daysbutton
 JButton left, right

inner class MonthButtonListener
 ① "<"
 ↳ MonthSchedule ms1 → 이전 달 (true)
 ② ">"
 ↳ MonthSchedule ms2 → 다음 달 (true)

inner class dateJButtonListener
 String dayNum
 → days[dayNum-1] → (true) *Day GUI days*

MonthSchedule (int input_year, int input_month)

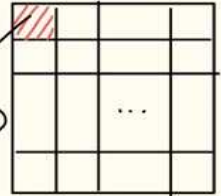
↳ JPanel



↳ days = DaySchedule [monthLength]

for 총 일 수 days[0]

DaySchedule (newSche)



for 해당월 스케줄 수 ↓
 days[해당달자-1].insert (Schedule)
 days[해당달자-1].backuptList (Schedule)
 for 총 일 수
 days[0].initData()

↳ setTitle
 setSize

- MonthSchedule class는 JFrame class를 상속 받음
- Property(변수) : 클래스 안에 있는 모든 인스턴스 변수들은 클래스 외부에서 직접 접근 안되도록 private 선언함
- 생성자 : default 선언함
- inner class들 : ActionListener class 상속 받음 / default class

<3> Project 6에서 구현한 코드 설명 (구조)

* 주요 결정사항 및 근거 * 한계/문제점 * 해결 방안

* 파란 글씨는 조건 만족을 위해 고려해 준 부분 or 코드를 짜며 생각해 본 것들

< 1-1 > ScheduleList class : Array형 변수를 ArrayList형태로 바꿈

```
13 public class ScheduleList {
14
15     private ArrayList<Schedule> array = new ArrayList<>();
16     private Writer file;
17     private String fileName;
18     private static String pattern = "(^\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}$)";
```

1) Project 3에서 일반 array로 구현한 Schedule들의 list를 ArrayList를 사용해서 확장함

* 주요 결정 사항 및 근거) 기존 Project들과는 달리 Schedule들의 list의 항목 수가 고정되어 있지 않고 무한으로 add될 수 있으므로 담을 수 있는 정보 수가 정해져 있는 일반 array 보다는 ArrayList를 사용하는 것이 옳다고 판단함

< 1-2 > ScheduleList class : addSchedule method 생성

```
129 public void saveScheduleList(ScheduleList s_list, String fileName) {
130     PrintWriter newFile = null;
131     try {
132         newFile = new PrintWriter(fileName);
133     }
134     catch(Exception e) {
135         System.out.println("error");
136     }
137
138     for(int i=0; i<s_list.numSchedules();i++)
139         newFile.println(s_list.getSchedule(i).getData());
140     newFile.close();
141 }
142
143 public void addSchedule(String name, String startTime, String endTime) {
144     array.add(new Schedule(name, LocalDateTime.parse(startTime), LocalDateTime.parse(endTime)));
145 }
146
147
148 public void addSchedule(String name, String startTime, String endTime, String memo) {
149     array.add(new Schedule(name, LocalDateTime.parse(startTime), LocalDateTime.parse(endTime), memo));
150 }
151
152
153 }
```

1) ScheduleList class의 saveScheduleList method는 정보를 save할 때 파일을 새로 쓰기 위한 함수임 -> 프로젝트3에서 구현했었음

2) ScheduleList class의 addSchedule method는 DaySchedule class에서 save동작 시 수정된 부분과 수정되지 않은 부분을 새로 작성할 때 사용되는 함수로 ArrayList에 넣는 용도

< 2-1 > DaySchedule class : 새로운 변수 생성

```
9 public class DaySchedule extends JFrame {
10
11     private ArrayList<JPanel> jpList = new ArrayList<JPanel>();
12     private JPanel p2;
13     private ArrayList<JPanel> jpBackup = new ArrayList<JPanel>();
14     private ScheduleList sl;
```

1) Project 5에서와는 달리 일정의 개수가 지정되어 있지 않으므로 일정을 담을 ArrayList<JPanel> jpList변수와 ArrayList<JPanel> jpBackup변수 생성

* 문제점) 처음에는 일반 Array로 구현하며 for문 등을 요긴하게 사용하려고 도전하였음. 그러나 DaySchedule GUI에서 Schedule을 표시하는 JTextField 부분을 넣을 p2의 row를 add나 save 과정에서 정확하게 알 수가 없어서 화면 배열이 이상해지는 오류가 발생함

ex1) 다음 스케줄은 이전 스케줄 다음 줄에 출력되어야 하는데 한 줄에 이어서 출력 됨

ex2) 모든 스케줄 TextField가 GUI 상에서 보이지 않음

* 해결방안) ArrayList<JPanel> 형태의 변수를 생성해 JPanel 형태의 변수 p2 안에 ArrayList<JPanel>형태의 정보를 add하는 방식을 취하여 해결

2) ScheduleList형 sl 객체를 생성해 ScheduleList와 DaySchedule을 연동 시킴

< 2-2 > DaySchedule class : inner class_DayButtonLister

```
17 class DayButtonListener implements ActionListener{
18     public void actionPerformed(ActionEvent e){
19         switch (e.getActionCommand()){
20
21             case "Save" :
22                 System.out.println("Save Button Clicked");
23
24                 for (int i = 0; i < sl.numSchedules(); i++) {
25                     for (JPanel a : jpBackup) {
26                         if (((JTextField)a.getComponent( n: 0)).getText().equals(sl.getSchedule(i).getName())) {
27                             sl.removeSchedule(i);
28                         }
29                     }
30                 }
31
32                 for (JPanel b : jpList) {
33                     final String name = ((JTextField)b.getComponent( n: 0)).getText();
34                     final String startTime = ((JTextField)b.getComponent( n: 1)).getText();
35                     final String endTime = ((JTextField)b.getComponent( n: 2)).getText();
36                     final String memo = ((JTextField)b.getComponent( n: 3)).getText();
37                     sl.addSchedule(name, startTime, endTime, memo);
38                 }
39                 sl.saveScheduleList(sl, sl.getFileName());
40                 setVisible(false);
41                 break;
42             }
43         }
```

```

44         case "Add" :
45             System.out.println("Add Button Clicked");
46             adding();
47             remove(p2);
48             p2.removeAll();
49             initData();
50             p2.revalidate();
51             p2.repaint();
52             break;
53
54         case "Cancel" :
55             System.out.println("Cancel Button Clicked");
56             jpList = null;
57             jpList = new ArrayList<>(jpBackup);
58             remove(p2);
59             p2.removeAll();
60             initData();
61             revalidate();
62             repaint();
63             setVisible(false);
64             break;
65
66     }
67 }
68 }
69 }

```

1) Save, Add, Cancel 동작을 위한 ActionListener

2) Save

* 오류가 잦아 코드 수정을 가장 많이 반복했던 부분

* 파일에는 해당 DaySchedule 일자의 내용 뿐만 아니라 다른 일자의 내용들도 저장되어 있음

* 생각 해 본 점) 수정 된 내용을 고려하여 파일을 저장하는 방법은?

수정 된 내용을 백업 데이터와 비교해서 지웠다가 다시 쓰는 방법은?

수정 된 일자의 내용만 업데이트하고 기존 다른 일자의 내용들은 그대로 저장시키기 위해서는 전체 코드 구조를 바꿔야할까?

* 주요 결정 사항) 기존 sl에 있는 내용은 JTextField에서 수정한다고 값이 바뀌는 것이 아니기 때문에 백업해놓은 JPanel들에 있는 JTextFieldef은 처음 데이터를 불러왔을 때와 그대로 값이 저장되어 있음. 그걸 이용해 원래 있었던 스케줄들을 sl에서 제거해주고, 수정한 JTextField의 값을 바탕으로 새로운 스케줄들을 생성해서 sl에 다시 넣어줌. 따라서 새로 만든 스케줄이랑, 수정한 스케줄 둘 다 '새 Schedule 객체'로서 들어가게 됨

3) Add

* 해당 일자의 마지막 일정 다음 줄에 JTextField(1, 4)를 삽입함

* 주요 결정 사항) DaySchedule Class method인 adding()으로 빈 칸을 삽입하도록 구현함

* 교수님께서 강의에서 언급하셨던 repaint 사용

3) Cancel

* jpList내용을 수정 전인 jpBackup으로 바꾼 후 GUI 창이 꺼지도록 구현함

* 교수님께서 강의에서 언급하셨던 repaint 사용

< 2-3 > DaySchedule class : initData method 생성

```
141 public void initData() {  
142     p2 = new JPanel(new GridLayout(jpList.size(), cols: 1));  
143     for (JPanel a : jpList)  
144         p2.add(a);  
145  
146     this.add(p2, BorderLayout.CENTER);  
147 }
```

- 1) p2 안에 ArrayList<JPanel>로 저장되어 있는 스케줄 정보들을 넣기 위한 함수
* 설계노트 그림 참고하면 더 쉬운 이해 (리포트 p.11)

< 2-4 > DaySchedule class : insert method 생성

```
149 @ public void insert(Schedule S){  
150     JPanel p2Content = new JPanel();  
151     p2Content.setLayout(new GridLayout( rows: 1, cols: 4));  
152     JTextField text_title = new JTextField(String.valueOf(S.getName()));  
153     JTextField text_stime= new JTextField(String.valueOf(S.getStartTime()));  
154     JTextField text_etime = new JTextField(String.valueOf(S.getEndTime()));  
155     JTextField text_memos = new JTextField(String.valueOf(S.getMemo()));  
156  
157     p2Content.add(text_title);  
158     p2Content.add(text_stime);  
159     p2Content.add(text_etime);  
160     p2Content.add(text_memos);  
161  
162     jpList.add(p2Content);  
163 }
```

- 1) Schedule형 S를 인자로 받아 S의 내용을 JTextField에 넣는 함수
* 설계노트 그림 참고하면 더 쉬운 이해 (리포트 p.11)

< 2-5 > DaySchedule class : backupList method 생성

```
164 @ public void backupList (Schedule S){  
165     JPanel p2Content2 = new JPanel();  
166     p2Content2.setLayout(new GridLayout( rows: 1, cols: 4));  
167     JTextField text_title2 = new JTextField(String.valueOf(S.getName()));  
168     JTextField text_stime2= new JTextField(String.valueOf(S.getStartTime()));  
169     JTextField text_etime2 = new JTextField(String.valueOf(S.getEndTime()));  
170     JTextField text_memos2 = new JTextField(String.valueOf(S.getMemo()));  
171  
172     p2Content2.add(text_title2);  
173     p2Content2.add(text_stime2);  
174     p2Content2.add(text_etime2);  
175     p2Content2.add(text_memos2);  
176  
177     jpBackup.add(p2Content2);  
178 }  
179 }
```

- 1) save버튼 동작이나 cancel버튼 동작을 위해 수정 전 내용을 백업 시키는 함수

< 3-1 > MonthSchedule class : inner class①_MonthButtonListner

```
21 class MonthJButtonListner implements ActionListener {
22     public void actionPerformed(ActionEvent e){
23         switch (e.getActionCommand()){
24             case "<":
25                 System.out.println("< Button Clicked");
26                 setVisible(false);
27                 final MonthSchedule ms1 = new MonthSchedule(first.minusMonths(1).getYear(), first.minusMonths(1).getMonthValue());
28                 ms1.setVisible(true);
29                 break;
30             case ">":
31                 System.out.println("> Button Clicked");
32                 setVisible(false);
33                 final MonthSchedule ms2 = new MonthSchedule(first.plusMonths(1).getYear(), first.plusMonths(1).getMonthValue());
34                 ms2.setVisible(true);
35                 break;
36         }
37     }
38 }
```

- 1) '<'버튼과 '>'버튼을 눌렀을 때 이전 달 or 다음 달로 넘어가기 위해 만든 ActionListener
 - 2) 새로운 MonthSchedule 객체(이전 달 or 다음 달 정보를 담음)를 생성해 기존 GUI창은 끄고 (setVisible(false)) 새로운 GUI창을 보이게 하는 원리
- * 생각해 본 다른 방법) 기존 MonthSchedule 생성자의 인자(int input_year, int input_month) 값을 변경시키는 방법도 가능할 것 같음

< 3-2 > MonthSchedule class : inner class②_dateJButtonListner

```
40 class dateJButtonListner implements ActionListener {
41     public void actionPerformed(ActionEvent e) {
42
43         final String dayNum = e.getActionCommand();
44         days[Integer.parseInt(dayNum) - 1].setVisible(true);
45     }
46 }
```

- 1) 각 day버튼을 클릭했을 때 DaySchedule GUI가 보이게 하기 위한 ActionListener
- 2) DaySchedule[]형 days[] 변수이므로 DaySchedule class에 따라 DaySchedule GUI를 setVisible(true) 하게 하여 나타냄

< 3-3 > MonthSchedule class : 생성자 부분 수정 -> Month와 Day 연결

```
137     days = new DaySchedule[monthLength];
138
139     for(int i=0;i<monthLength;i++){
140         days[i]= new DaySchedule(newSche);
141     }
142
143     for(int i=0;i<newSche.numSchedules();i++){
144         Schedule thisSchedule = newSche.getSchedule(i);
145         int dayNum = thisSchedule.getStartTime().getDayOfMonth();
146         days[dayNum - 1].insert(thisSchedule);
147         days[dayNum - 1].backupList(thisSchedule);
148     }
149
150     for (int i = 0; i < days.length; i++) {
151         days[i].initData();
152     }
```

1) 각 날짜에 해당하는 정보를 GUI에 각각 나타내기 위해 구현한 원리

* 주요 결정 사항)

- > days[] = new DaySchedule(newSche)를 그 달의 총 일 수만큼 반복해 객체 생성
- > int dayNum 변수로 텍스트파일 정보가 몇 일에 해당하는 GUI에 넣어야하는 지 판단
- > initData를 해주며 해당 일에 해당하는 스케줄의 총 개수만큼 GUI에 넣어 줌

<4> 부록 (전체 코드)

< 1-1 > Schedule class

```
DaySchedule.java × MonthSchedule.java × Schedule.java × ScheduleList.java × Main.java ×
1 package com.company;
2
3 import java.time.LocalDateTime;
4 import java.time.format.DateTimeFormatter;
5
6 public class Schedule {
7     private String name;
8     private LocalDateTime startTime;
9     private LocalDateTime endTime;
10    private String memo;
11    Schedule(){
12
13    }
14    Schedule (String name2, LocalDateTime sTime, LocalDateTime eTime) {
15        name = name2;
16        startTime = sTime;
17        endTime = eTime;
18    }
19    Schedule (String name2, LocalDateTime sTime, LocalDateTime eTime, String memo2) {
20        this(name2, sTime, eTime);
21        memo = memo2;
22    }
23
24    public String getName() { return name; }
27
28    public void setName(String name) { this.name = name; }
31
32    public LocalDateTime getStartTime() { return startTime; }
35
36    public void setStartTime(LocalDateTime startTime) { this.startTime = startTime; }
39
40    public LocalDateTime getEndTime() { return endTime; }
43
44    public void setEndTime(LocalDateTime endTime) { this.endTime = endTime; }
47
48    public String getMemo() { return memo; }
51
52    public void setMemo(String memo) { this.memo = memo; }
55
56    public String getData() {
57        if (memo != null) return name + "/" + startTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")) + "/"
58            + endTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")) + "/" + memo;
59        else return name + "/" + startTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")) + "/"
60            + endTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm"));
61    }
62
63    public void print() {
64        if(memo==null)
65            System.out.println(name + "/" + startTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm"))
66                + "/" + endTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")));
67        else
68            System.out.println(name + "/" + startTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm"))
69                + "/" + endTime.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")) + "/" + memo);
70    }
71
72 }
```

< 2-1 > ScheduleList class

```
DaySchedule.java × MonthSchedule.java × Schedule.java × ScheduleList.java × Main.java ×
1 package com.company;
2
3 import javax.swing.*;
4 import java.util.ArrayList;
5 import java.util.Scanner;
6 import java.util.regex.Pattern;
7 import java.io.File;
8 import java.time.LocalDateTime;
9 import java.time.format.DateTimeFormatter;
10 import java.io.PrintWriter;
11 import java.io.Writer;
12
13 public class ScheduleList {
14
15     private ArrayList<Schedule> array = new ArrayList<>();
16     private Writer file;
17     private String fileName;
18     private static String pattern = "(^\\d{4}-\\d{2}-\\d{2} \\d{2}:\\d{2}$)";
19
20
21     ScheduleList(String fileName) {
22
23         this.fileName = fileName;
24         File file = new File(fileName);
25         try {
26
27             Scanner input = new Scanner(file);
28             while(input.hasNext()) {
29                 String line = input.nextLine();
30                 line = line.trim();
31
32                 if (line.isBlank() || line.charAt(0) == ';')
33                     continue;
34
35                 String[] div = line.split( regex: "/" );
36
37                 String tempStartTime, tempEndTime;
38
39                 String name;
40                 String memo;
41                 LocalDateTime startTime;
42                 LocalDateTime endTime;
43
44                 name = div[0].trim();
45                 tempStartTime = div[1].trim();
46                 tempEndTime = div[2].trim();
47                 if(div.length > 3) {
48                     memo = div[3].trim();
49                 }
50                 else {
51                     memo = null;
52                 }
53             }
54         }
55     }
56 }
```

```

53
54     if (name.isBlank()) {
55         System.out.print("No Schedule Title ; Skip the input line : ");
56         System.out.println(line);
57         continue;
58     }
59
60     if (!Pattern.matches(pattern, tempStartTime)) {
61         System.out.print("Wrong Date Format ; Skip the input line : ");
62         System.out.println(line);
63         continue;
64     }
65     else {
66         startTime = LocalDateTime.parse(tempStartTime, DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm"));
67     }
68
69     if (!Pattern.matches(pattern, tempEndTime)) {
70         System.out.print("Wrong Date Format ; Skip the input line : ");
71         System.out.println(line);
72         continue;
73     }
74     else {
75         endTime = LocalDateTime.parse(tempEndTime, DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm"));
76     }
77
78     if (startTime.compareTo(endTime) > 0) {
79         System.out.print("Start Time is later than End Time ; Skip the input line : ");
80         System.out.println(line);
81         continue;
82     }
83
84     if (div.length < 3) {
85         System.out.print("Put too little information ; Skip the input line : ");
86         System.out.println(line);
87         continue;
88     }
89
90     if (div.length > 4) {
91         System.out.print("Put too much information ; Skip the input line : ");
92         System.out.println(line);
93         continue;
94     }
95
96
97     if (memo == null) {
98         Schedule schedule = new Schedule(name, startTime, endTime);
99         array.add(schedule);
100     } else {
101         Schedule schedule = new Schedule(name, startTime, endTime, memo);
102         array.add(schedule);
103     }
104 }
105 input.close();
106
107 }
108 catch (Exception e) {
109     System.out.println("Unknown File. Create New File.");
110 }
111 }
112

```

```

112
113 public void removeSchedule(int i) {
114     array.remove(i);
115 }
116
117 public String getFileName() {
118     return fileName;
119 }
120
121 public int numSchedules() {
122     return array.size();
123 }
124
125 public Schedule getSchedule(int i) {
126     return array.get(i);
127 }
128
129 public void saveScheduleList(ScheduleList s_list, String fileName) {
130     PrintWriter newFile = null;
131     try {
132         newFile = new PrintWriter(fileName);
133     }
134     catch(Exception e) {
135         System.out.println("error");
136     }
137
138     for(int i=0; i<s_list.numSchedules();i++)
139         newFile.println(s_list.getSchedule(i).getData());
140     newFile.close();
141 }
142
143 public void addSchedule(String name, String startTime, String endTime) {
144     array.add(new Schedule(name, LocalDateTime.parse(startTime), LocalDateTime.parse(endTime)));
145 }
146
147
148 public void addSchedule(String name, String startTime, String endTime, String memo) {
149     array.add(new Schedule(name, LocalDateTime.parse(startTime), LocalDateTime.parse(endTime), memo));
150 }
151
152
153 }

```


< 3-1 > DaySchedule class

```
DaySchedule.java x MonthSchedule.java x Schedule.java x ScheduleList.java x Main.java x
1 package com.company;
2
3 import java.awt.*;
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6 import java.util.ArrayList;
7 import javax.swing.*;
8
9 public class DaySchedule extends JFrame {
10
11     private ArrayList<JPanel> jpList = new ArrayList<JPanel>();
12     private JPanel p2;
13     private ArrayList<JPanel> jpBackup = new ArrayList<JPanel>();
14     private ScheduleList sl;
15
16
17     class DayButtonListener implements ActionListener{
18         public void actionPerformed(ActionEvent e){
19             switch (e.getActionCommand()){
20
21                 case "Save" :
22                     System.out.println("Save Button Clicked");
23
24                     for (int i = 0; i < sl.numSchedules(); i++) {
25                         for (JPanel a : jpBackup) {
26                             if (((JTextField)a.getComponent( n: 0)).getText().equals(sl.getSchedule(i).getName())) {
27                                 sl.removeSchedule(i);
28                             }
29                         }
30                     }
31
32                     for (JPanel b : jpList) {
33                         final String name = ((JTextField)b.getComponent( n: 0)).getText();
34                         final String startTime = ((JTextField)b.getComponent( n: 1)).getText();
35                         final String endTime = ((JTextField)b.getComponent( n: 2)).getText();
36                         final String memo = ((JTextField)b.getComponent( n: 3)).getText();
37                         sl.addSchedule(name, startTime, endTime, memo);
38                     }
39
40                     sl.saveScheduleList(sl, sl.getFileName());
41                     setVisible(false);
42                     break;
43
44                 case "Add" :
45                     System.out.println("Add Button Clicked");
46                     adding();
47                     remove(p2);
48                     p2.removeAll();
49                     initData();
50                     p2.revalidate();
51                     p2.repaint();
52                     break;
53

```

```

54         case "Cancel" :
55             System.out.println("Cancel Button Clicked");
56             jpList = null;
57             jpList = new ArrayList<>(jpBackup);
58             remove(p2);
59             p2.removeAll();
60             initData();
61             revalidate();
62             repaint();
63             setVisible(false);
64             break;
65
66
67     }
68 }
69 }
70
71
72
73 DaySchedule(ScheduleList sl) {
74     this.sl = sl;
75     JPanel p1 = new JPanel();
76     p1.setLayout(new GridLayout( rows 1, cols 4));
77     JPanel p3 = new JPanel();
78     p3.setLayout(new GridLayout( rows 1, cols 3));
79
80
81     JLabel label_title = new JLabel( text: "Title");
82     JLabel label_stime = new JLabel( text: "StartTime");
83     JLabel label_etime = new JLabel( text: "EndTime");
84     JLabel label_memos = new JLabel( text: "Memos");
85
86     label_title.setHorizontalAlignment(SwingConstants.CENTER);
87     label_stime.setHorizontalAlignment(SwingConstants.CENTER);
88     label_etime.setHorizontalAlignment(SwingConstants.CENTER);
89     label_memos.setHorizontalAlignment(SwingConstants.CENTER);
90
91     p1.add(label_title);
92     p1.add(label_stime);
93     p1.add(label_etime);
94     p1.add(label_memos);
95
96     p2 = new JPanel();
97
98     add(p1, BorderLayout.NORTH);
99
100     JButton button_right = new JButton( text: "Save");
101     JButton button_mid = new JButton( text: "Add");
102     JButton button_left = new JButton( text: "Cancel");
103
104     DayButtonListener jbListner = new DayButtonListener();
105     button_right.addActionListener(jbListner);
106     button_mid.addActionListener(jbListner);
107     button_left.addActionListener(jbListner);
108
109     button_right.setHorizontalAlignment(SwingConstants.CENTER);
110     button_mid.setHorizontalAlignment(SwingConstants.CENTER);
111     button_left.setHorizontalAlignment(SwingConstants.CENTER);
112
113     p3.add(button_left);
114     p3.add(button_mid);
115     p3.add(button_right);

```

```

117     add(p3, BorderLayout.SOUTH);
118     this.setSize( width: 600, height: 300);
119     this.setTitle("Day Schedule");
120     setLocationRelativeTo(null); // Center the frame
121     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // x누르면 GUI 다 꺼짐
122 }
123
124 public void adding () {
125     JPanel p2Contents = new JPanel();
126     p2Contents.setLayout(new GridLayout( rows: 1, cols: 4));
127     JTextField text_title = new JTextField();
128     JTextField text_stime= new JTextField();
129     JTextField text_etime = new JTextField();
130     JTextField text_memos = new JTextField();
131
132     p2Contents.add(text_title);
133     p2Contents.add(text_stime);
134     p2Contents.add(text_etime);
135     p2Contents.add(text_memos);
136
137     jpList.add(p2Contents);
138 }
139
140 public void initData() {
141     p2 = new JPanel(new GridLayout(jpList.size(), cols: 1));
142     for (JPanel a : jpList)
143         p2.add(a);
144
145     this.add(p2, BorderLayout.CENTER);
146 }
147
148 @
149 public void insert(Schedule S){
150     JPanel p2Content = new JPanel();
151     p2Content.setLayout(new GridLayout( rows: 1, cols: 4));
152     JTextField text_title = new JTextField(String.valueOf(S.getName()));
153     JTextField text_stime= new JTextField(String.valueOf(S.getStartTime()));
154     JTextField text_etime = new JTextField(String.valueOf(S.getEndTime()));
155     JTextField text_memos = new JTextField(String.valueOf(S.getMemo()));
156
157     p2Content.add(text_title);
158     p2Content.add(text_stime);
159     p2Content.add(text_etime);
160     p2Content.add(text_memos);
161
162     jpList.add(p2Content);
163 }
164 @
165 public void backupList (Schedule S){
166     JPanel p2Content2 = new JPanel();
167     p2Content2.setLayout(new GridLayout( rows: 1, cols: 4));
168     JTextField text_title2 = new JTextField(String.valueOf(S.getName()));
169     JTextField text_stime2= new JTextField(String.valueOf(S.getStartTime()));
170     JTextField text_etime2 = new JTextField(String.valueOf(S.getEndTime()));
171     JTextField text_memos2 = new JTextField(String.valueOf(S.getMemo()));
172
173     p2Content2.add(text_title2);
174     p2Content2.add(text_stime2);
175     p2Content2.add(text_etime2);
176     p2Content2.add(text_memos2);
177
178     jpBackup.add(p2Content2);
179 }

```

< 4-1 > MonthSchedule class

```
DaySchedule.java x MonthSchedule.java x Schedule.java x ScheduleList.java x Main.java x
1 package com.company;
2
3 import java.awt.*;
4
5 import javax.swing.*;
6 import java.awt.event.ActionEvent;
7 import java.awt.event.ActionListener;
8 import java.time.*;
9
10 public class MonthSchedule extends JFrame {
11
12     private static DaySchedule[] days;
13     LocalDate first;
14     LocalDate ld;
15     private static ScheduleList newSche;
16     private static JButton[] daysbutton;
17     private static JButton left;
18     private static JButton right;
19
20
21     class MonthJButtonListener implements ActionListener {
22     @Override
23     public void actionPerformed(ActionEvent e){
24         switch (e.getActionCommand()){
25             case "<":
26                 System.out.println("< Button Clicked");
27                 setVisible(false);
28                 final MonthSchedule ms1 = new MonthSchedule(first.minusMonths(1).getYear(), first.minusMonths(1).getMonthValue());
29                 ms1.setVisible(true);
30                 break;
31
32             case ">":
33                 System.out.println("> Button Clicked");
34                 setVisible(false);
35                 final MonthSchedule ms2 = new MonthSchedule(first.plusMonths(1).getYear(), first.plusMonths(1).getMonthValue());
36                 ms2.setVisible(true);
37                 break;
38         }
39     }
40
41     class dateJButtonListner implements ActionListener {
42     @Override
43     public void actionPerformed(ActionEvent e) {
44         final String dayNum = e.getActionCommand();
45         days[Integer.parseInt(dayNum) - 1].setVisible(true);
46     }
47     }
48
49     MonthSchedule(int input_year, int input_month) {
50
51         first = LocalDate.of(input_year, input_month, dayOfMonth(1));
52         newSche = new ScheduleList( fileName: "C:\\Users\\배인경\\Desktop\\" + first.getYear() + String.format("%02d", first.getMonthValue()) + ".data");
53
54         JPanel borderLayout = new JPanel(new BorderLayout());
55         JPanel panel3 = new JPanel();
56         panel3.setLayout(new GridLayout( rows: 6, cols: 7));
57         JPanel panel2 = new JPanel(new GridLayout( rows: 1, cols: 7));
58
59         ld = LocalDate.parse(input_year+"-"+String.format("%02d", input_month) + "-01");
60         int monthLength = ld.lengthOfMonth();
61         int startPosition = ld.getDayOfWeek().getValue();
```



```

61
62
63 JLabel day1 = new JLabel( text: "MON");
64 JLabel day2 = new JLabel( text: "TUE");
65 JLabel day3 = new JLabel( text: "WED");
66 JLabel day4 = new JLabel( text: "THR");
67 JLabel day5 = new JLabel( text: "FRI");
68 JLabel day6 = new JLabel( text: "SAT");
69 JLabel day7 = new JLabel( text: "SUN");
70 day1.setHorizontalAlignment(SwingConstants.CENTER);
71 day2.setHorizontalAlignment(SwingConstants.CENTER);
72 day3.setHorizontalAlignment(SwingConstants.CENTER);
73 day4.setHorizontalAlignment(SwingConstants.CENTER);
74 day5.setHorizontalAlignment(SwingConstants.CENTER);
75 day6.setHorizontalAlignment(SwingConstants.CENTER);
76 day7.setHorizontalAlignment(SwingConstants.CENTER);
77
78 panel2.add(day7);
79 panel2.add(day1);
80 panel2.add(day2);
81 panel2.add(day3);
82 panel2.add(day4);
83 panel2.add(day5);
84 panel2.add(day6);
85
86 for (int i = 0; i < startPosition; i++) {
87     JButton temp = new JButton();
88     temp.setEnabled(false);
89     panel3.add(temp);
90 }
91
92 daysbutton = new JButton[monthLength];
93
94 for (int i = 0; i < monthLength; i++) {
95     daysbutton[i] = new JButton(Integer.toString(i+1));
96     daysbutton[i].addActionListener(new dateJButtonListener());
97 }
98
99 for (int i = 0; i < monthLength; i++) {
100     panel3.add(daysbutton[i]);
101 }
102 int last = 42 - monthLength - startPosition;
103 for(int i=0;i<last;i++) {
104     JButton temp = new JButton();
105     temp.setEnabled(false);
106     panel3.add(temp);
107 }
108
109 JPanel panel1 = new JPanel(new BorderLayout());
110 panel1.setLayout(new GridLayout( rows: 1, cols: 3));
111 JPanel p1middle = new JPanel(new BorderLayout());
112 p1middle.setLayout(new GridLayout( rows: 2, cols: 1));
113
114 left = new JButton( text: "<");
115 JLabel year = new JLabel(Integer.toString(input_year));
116 JLabel month = new JLabel(String.format("%02d", input_month));
117
118 year.setHorizontalAlignment(SwingConstants.CENTER);
119 month.setHorizontalAlignment(SwingConstants.CENTER);
120 p1middle.add(year);
121 p1middle.add(month);
122

```



```

123     right = new JButton( text: ">");
124     panel1.add(left);
125     panel1.add(p1middle);
126     panel1.add(right);
127
128     MonthJButtonListener mjbListner = new MonthJButtonListener();
129     left.addActionListener(mjbListner);
130     right.addActionListener(mjbListner);
131
132     add(panel1, BorderLayout.NORTH);
133     BorderLayout.add(panel3, BorderLayout.CENTER);
134     BorderLayout.add(panel2, BorderLayout.NORTH);
135     add(BorderLayout, BorderLayout.CENTER);
136
137     days = new DaySchedule[monthLength];
138
139     for(int i=0;i<monthLength;i++){
140         days[i]= new DaySchedule(newSche);
141     }
142
143     for(int i=0;i<newSche.numSchedules();i++){
144         Schedule thisSchedule = newSche.getSchedule(i);
145         int dayNum = thisSchedule.getStartTime().getDayOfMonth();
146         days[dayNum - 1].insert(thisSchedule);
147         days[dayNum - 1].backupList(thisSchedule);
148     }
149
150     for (int i = 0; i < days.length; i++) {
151         days[i].initData();
152     }
153
154
155     setTitle("Schedule Planner");
156     setSize( width: 600, height: 600);
157     setLocationRelativeTo(null); // Center the frame
158     setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
159 }
160
161
162
163 }

```

< 5-1 > Main class

```

DaySchedule.java × MonthSchedule.java × Schedule.java × ScheduleList.java × Main.java ×
1 package com.company;
2
3 public class Main {
4
5     public static void main(String[] args) {
6         MonthSchedule frame = new MonthSchedule( input_year: 2020, input_month: 6);
7         frame.setVisible(true);
8     }
9 }

```

평가표 (즉, 리포트 작성시 체크 사항)

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"> - “초기” 동작 - “Add” 동작 - “Cancel” 동작 - “Save” 동작 - 기타 비정상 동작 실험 (구현 한계 점검) 	<ul style="list-style-type: none"> - 초기 동작 (p. 2 ~ 5) - Add 동작 (p. 5) - Cancel 동작 (p. 6) - Save 동작 (p. 7 ~ 8) - 기타 비정상 동작 (p. 9 ~ 10) <p>=> 프로젝트 6 과제 설명에서 구현하라고 한 모든 상황들에 대해 동작이 잘 됨</p>		
<p>설계 노트</p> <ul style="list-style-type: none"> - 주요 결정사항 및 근거 - 한계/문제점 - 해결 방안 - 필수내용: * 프로그램 구성(Class 구조) * member visibility 	<ul style="list-style-type: none"> ● p. 11 ~ 13 설계 노트 - 프로그램 구성(Class 구조) -> 그림으로 표현(이미지) - member visibility -> 말로 작성 ● p. 14 ~ 19 주요 코드 설명 - 주요 결정사항 및 근거 - 한계 / 문제점 - 해결 방안 => 모두 ' * 파란글씨 ' 로 표현 		
<p>리포트</p> <ul style="list-style-type: none"> - 평가자 시각으로 리포트 검토 - 위의 평가 요소들이 명확하게 기술되었는가? 	<ul style="list-style-type: none"> - 평가자 시각에서 봤을 때 확인하고 싶은 동작을 모두 동작시키고 인증함 - 프로그램 구조를 설계 노트 그림을 이용해 보기 쉽게 나타냄 - 바탕화면에 생성된 파일을 통해 '배인경' 폴더로 본인이 직접 프로그래밍하였음을 인증함 - 평가 요소들을 명확한 표현으로 기술 		
<p>총평/계</p>	<ul style="list-style-type: none"> - 부록으로 전체 코드를 캡처하여 첨부함 - java에 대한 이해도가 매우 높아짐 		

* 학생 자체 평가는 점수에 반영되지 않음.

* 학생 스스로 자신의 보고서를 평가하면서, 체계적으로 프로젝트를 마무리하도록 유도하는 것이 목적임.