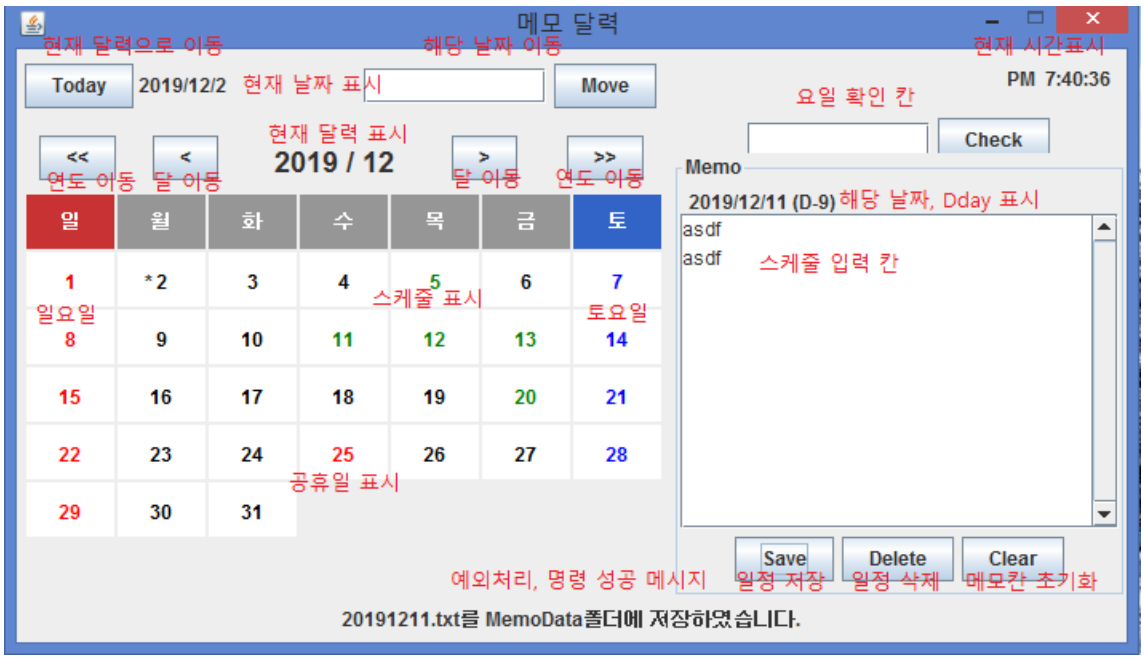
객체지향 프로그래밍 설계

- 최종보고서 -

IT정보공학과 201515310 주성현

1. 최종 설계결과물 개요

달력의 기능 설명



• 사용자에게 편리하게 이용할 수 있는 GUI(Graphical User Interface)를 이용해 그래픽 이용한 달력을 구성하고 그 외에 다른 기능등을 포함하는 화면을 구성하고, 마우스를 통해 사용자가 입력을 편리하게 하도록 작성하는 User Interface를 제공한다. Java의 Swing을 라이브러리를 통하여 비교적 가볍게 실행할 수 있다.

• 처음 실행하면 현재 날짜의 달력이 출력되어 User가 현재의 달력으로 이동하지 않고 현재 날짜의 달력을 볼 수 있다.

• 왼쪽 상단에 현재 날짜가 출력되어 현재 날짜를 간편하게 알 수 있다.

• 왼쪽 상단에 ‘Today’라는 버튼을 통해서 달력이 어느 달에 달력이 있든지 현재 날짜의 달력으로 출력된다.

• 달력 내에서 현재 날짜는 ‘*’표시를 통해서 현재 날짜와 요일을 쉽게 알 수 있다.

• User가 원하는 달의 달력을 보기를 원하면 위쪽 상단에 JTextField에 YYYY-mm 형식에 맞춰 연과 월을 입력하고 ‘Move’ 버튼을 누르면 해당 년도, 해당 월의 달력이 출력된다.

• User가 보이는 달력에서 이전 달과 다음 달의 달력을 보고 싶으면 ‘<’과 ‘>’버튼을 클릭하면 이전 달과 다음 달의 달력을 볼 수 있고, User가 현재 보이는 달력이 기준이 되어 이전 달과 다음 달의 달력을 볼 수 있다.

• User가 보이는 달력에서 이전 연도와 다음 연도의 달력을 보고 싶으면 ‘<<’과 ‘>>’버튼을 클릭하면 이전 연도와 다음 연도의 달력을 볼 수 있고, User가 현재 보이는 달력이 기준이 되어 이전 연도와 다음 연도의 달력을 볼 수 있다.

• 달력에서 평일은 검정색 글씨, 일요일은 빨간색 글씨, 토요일은 파란색 글씨로 보여 User가 쉽게 평일과 일요일, 토요일 구분을 할 수 있다.

• 공휴일과 국경일은 빨간색으로 출력되어 User가 쉽게 공휴일을 알 수 있으며 메모에 공휴일이나 국경일의 날이 출력되며 해당 날짜의 24절기도 출력된다.

• User가 원하는 날짜에 스케줄을 우측 ‘Memo’칸에 입력할 수 있으며 ‘Save’버튼을 누르면 저장이 되고 해당 날짜가 초록색으로 변하여 User가 쉽게 해당 날짜의 스케줄 확인을 할 수 있다. 스케줄이 저장이 되면 해당 날짜를 누르면 ‘Memo’창에 저장된 스케줄을 확인할 수 있다. ‘Delete’버튼은 저장된 스케줄을 삭제시키는 기능이며 Save되어 초록색으로 변한 날짜도 원래의 날짜 색으로 바뀌며 스케줄이 삭제된다. ‘Delete’버튼은 ‘Memo’칸에 내용을 초기화시켜주는 기능이다.

• ‘Memo’칸 안에 해당 날짜가 출력되며 ‘()’안에는 해당 날짜와 현재 날짜의 차를 계산하

여 D-day를 출력한다. 현재 날짜보다 이전 날짜는 'D+'로 출력되며 현재 날짜보다 이후 날짜는 'D-'로 출력된다.

• 해당 날짜를 클릭하면 'Memo'칸에 날짜가 해당 날짜로 변하여 User가 쉽게 구분 할 수 있도록 한다.

• 해당 날짜를 클릭하면 달력에 네모난 칸이 생기며 User가 쉽게 해당 날짜의 작업을 알 수 있게 한다.

• 우측 상단에 현재 시간이 표시되며 12시간 기준시간표를 사용하며 AM/PM으로 구분된다.

• 우측 상단에 빈칸에 'YYYY-mm-DD'형식으로 날짜를 입력하면 해당 날짜의 요일이 하단 중간에 출력된다.

• User의 잘못된 명령시에 예외처리 메시지가 하단 중간에 출력된다.(메모저장, 삭제, 형식에 맞지 않는 날짜입력, File 예외처리)

• 해당 버튼에 위에 마우스 커서를 올려두면 해당 버튼에 대한 도움말이 풍선 형태로 나타나 User가 해당 버튼에 대한 기능 사용법을 알 수 있다.

2. 중간보고서 이후 진행 내용

• **GUI** : 중간보고서에는 Eclipse의 콘솔창으로 달력과 기능을 출력했지만, 최종보고서에는 Java의 Swing을 이용하여 User가 이용하기 쉽고 편리한 GUI창을 통하여 달력과 각종 기능을 출력했다.

• **색상 표시** : 평일과 토요일, 일요일의 날짜 색상을 다르게 하여 User가 쉽게 해당 주중과 주말을 User가 쉽게 인식할 수 있도록 하였다.

• **공휴일 처리** : hashmap으로 날짜를 key로 받고 공휴일을 value로 저장하고 저장한 key 값으로 hollyday라는 디텍토리에 key.txt의 파일생성후 Filewriter을 이용해 key.txt에 value값을 저장하였다.(해당 날짜.txt에 공휴일 저장) 해당하는 파일이 있을 경우 공휴일을

빨간색글씨로 처리하고 메모장에 value값(공휴일)을 출력했다.

• **24절기 처리** : hashmap으로 날짜를 key로 받고 24절기를 value로 저장하고 저장한 key값으로 Seasonal이라는 디텍토리에 key.txt의 파일 생성후 Filewriter을 이용해 key.txt에 value값을 저장하였다.(해당 날짜.txt에 공휴일 저장) 해당하는 파일이 있을 경우 메모장에 value값(24절기)을 출력했다.

• **현재 달력으로 이동** : cal.getCalender를 통해 현재 날짜를 받아와 'Today' 버튼으로 현재 달력으로 이동 할 수 있는 기능추가.

• **연도 이동** : <<, >> 버튼을 통해 지난 연도와 다음 연도의 달력을 출력할 수 있는 기능추가.

• **스케줄 기능1** : 해당 날짜마다 스케줄을 입력할 수 있는 'Memo'칸이 있어 달력에서 해당 날짜에 쉽게 스케줄이 입력 가능하다.

• **스케줄 기능2** : 스케줄을 'Save', 'Delete' 'Clear'를 할 수 있는 버튼을 추가하였고 'Save'버튼 누를 시 해당 날짜의 스케줄이 삭제되며 해당 날짜의 색이 기본 날짜의 색으로 바뀜.

• **현재 시간 출력** : Thread를 이용하여 동시 프로그래밍을 하여 현재시간을 계속 표시하며 Calender클래스로 현재시간을 받아왔다.

• **명령처리 메시지** : 명령 성공시 성공 메시지를 출력한다. 또한, User의 잘못된 명령시 예외처리 메시지를 중앙 하단에 출력하여 User가 작업의 실패원인을 쉽게 알도록 한다.(메모 저장, 삭제, 형식에 맞지 않는 날짜입력, File 예외처리)

• **현재 날짜 표시** : 달력에 현재날짜에 '*'표시를 하여 User가 쉽게 현재 날짜를 알 수 있도록 했다.

• **D-Day 기능** : 현재날짜와 해당 날짜의 차를 구하여 D-day를 표시한다.(현재 날짜 이후 날짜 D-로 표시, 현재 날짜 이전 날짜 D+로 표시)

• **해당 날짜 표시** : 달력에 해당 날짜를 클릭하면 해당 날짜에 네모난 칸이 생성되어 User가 해당 날짜에 대한 작업을 이해 할 수 있게 한다.

• **코드 수정** : 지난 달, 다음 달의 달력으로 이동할 때 해당 달의 날짜를 다 표시하지 못하는 오류를 수정하였고 완벽하게 구현하였다.

3. 코드 설명

- 중요 코드와 기능들의 코드만 설명

• 달력을 만드는 Class 및 중요 전역 변수

```
class CalendarDateManager{ //
static final int CAL_WIDTH = 7; //달력의 세로크기
final static int CAL_HEIGHT = 6; // 달력의 가로크기
int calDates[][] = new int[CAL_HEIGHT][CAL_WIDTH]; //달력의 날짜를 만들 크기
int calYear; //년도를 저장할 변수
int calMonth; // 월을 저장할 변수
int calDayOfMon; //현재 월의 날짜를 가져올 변수
final int calLastDateOfMonth[]={31,28,31,30,31,30,31,31,30,31,30,31};// 월의 날짜 표시
int calLastDate; // 이전 달의 마지막날을 저장할 변수
Calendar today = Calendar.getInstance(); // Calendar인스턴스 생성
Calendar cal; //Calendar인스턴스를 가져올 변수
HashMap<String, String> hollyMap;//공휴일을 저장할 hashmap
HashMap<String, String> SeasonMap;//24절기를 저장할 hashmap
String day;//요일을 저장할 String 변수
```

• 현재 날짜를 가져올 메소드

```
public void setToday(){
calYear = today.get(Calendar.YEAR); // 현재 년도를 가져옴
calMonth = today.get(Calendar.MONTH); // 현재 월을 가져옴
calDayOfMon = today.get(Calendar.DAY_OF_MONTH); // 현재 일을 가져옴
makeCalData(today); //달력을 만드는 함수에 오늘의 날짜를 매개변수로 실행
}
public void makeCalData(Calendar cal){ //달력을 만드는 함수
```

```
// 1일의 위치와 마지막 날짜를 구함
int calStartingPos = (cal.get(Calendar.DAY_OF_WEEK)+7-
(cal.get(Calendar.DAY_OF_MONTH))%7)%7; // 현재달 1의 위치를 받음
if(calMonth == 1) { //1월일경우 윤년을 확인하여 일수확인
calLastDate = calLastDateOfMonth[calMonth] + leapCheck(calYear);
}
else {//1월이아닌경우 원래의 일수사용
calLastDate = calLastDateOfMonth[calMonth];
}
// 달력 배열 초기화
for(int i = 0 ; i<CAL_HEIGHT ; i++){
for(int j = 0 ; j<CAL_WIDTH ; j++){
calDates[i][j] = 0;}                              }
}
// 달력 배열에 값 채워넣기
for(int i = 0, num = 1, k = 0 ; i<CAL_HEIGHT ; i++){
if(i == 0) k = calStartingPos;
else k = 0;
for(int j = k ; j<CAL_WIDTH ; j++){
if(num <= calLastDate) calDates[i][j]=num++;
}
|
        }
```

• 이전 달과 다음 달로 이동하는 메소드

```
public void moveMonth(int mon){ // 현재달로 부터 n달 전후를 받아 달력 배열을
만드는 함수(1년은 +12, -12달로 이동 가능)
calMonth += mon; // 원래의 calmMonth 값에 매개변수 mon을 받아서 달력 옮기기
if(calMonth>11) { //12월일때 다음년도 1월로 넘어가는 if문
while(calMonth>11){ // 원래의 M
calYear++;
calMonth -= 12;
}
} else if (calMonth<0) {//1월일때 지난년도 12월로 넘어가는 if문
while(calMonth<0){
calYear--;
calMonth += 12;
```

```
}
}
cal = new GregorianCalendar(calYear, calMonth, calDayOfMon); //if문을 통해여 나온 값을 새로운 cal변수로 지정
makeCalData(cal);//나온 cal을 달력으로 만듬
}
}
```

• 창의 구성요소와 배치를 만들 MemoCalendar class

```
public class MemoCalendar extends
CalendarDataManager{ //CalendarDataManager를 상속받아 창 구성요소와 배치를 만들 class
```

• 빈칸에 원하는 연도의 달을 입력하면 해당 하는 달력을 보여줄 수 있게하는 이벤트 처리

```
MoveMonth.addActionListener(new ActionListener(){//원하는 달력을 볼 수 있게하는 이동버튼 이벤트처리
public void actionPerformed(ActionEvent arg0) {
try {
cal = Calendar.getInstance();//cal instance받아오기
String get_date = MoveField.getText();//field에 있는 값 String화
int idx = get_date.indexOf("-");//'-' 에따른 구분 int값
String get_year = get_date.substring(0,idx);//'-'이전까지 String 나누기 -> year
String get_mon = get_date.substring(idx+1);//'-' 이후 String 나누기 -> month
int move_year = Integer.parseInt(get_year);//String year -> int형으로 바꾸기
int move_mon = Integer.parseInt(get_mon);//String month -> int형으로 바꾸기
cal.set(Calendar.YEAR, move_year);//가져온 year로 calendar set.
cal.set(Calendar.MONTH, move_mon-1);//가져온 mon로 calendar set.
cal.set(Calendar.DATE, 1);//date 1일로 가져옴
calYear = move_year; //전역변수 year을 받은 year로 변경
calMonth = move_mon-1;//전역변수 month를 받은 mon로 변경

curMMYYYYLab.setText("<html><table width=100><tr><th><font size=5>"+calYear+
" / "+((calMonth+1)<10?" ":"")+(calMonth+1)+"</th></tr></table></html>");
///해당 연도와 월을 보여줄 수 있는 표시
MakeCalData(cal);//받은 cal로 달력만들기
showCal();//달력실행
}
catch(Exception e) {
```

```
bottomInfo.setText("yyyy-mm 형식으로 작성해주세요.");// 예외처리 - 작성 형식 요청
}
}
});
```

• 달력을 만들 코드 부분, 평일, 일요일,토요일 색상표시

```
for(int i=0 ; i<CAL_WIDTH ; i++){//달력의 월화수목금토일 요일 표시
weekDaysName[i]=new JButton(WEEK_DAY_NAME[i]);
weekDaysName[i].setBorderPainted(false);//버튼의 외각선 없애
        weekDaysName[i].setContentAreaFilled(false);//내용영역 채우지 않음.
        weekDaysName[i].setForeground(Color.WHITE);//글짜색을 흰색으로 설정
if(i == 0) {
weekDaysName[i].setBackground(new Color(200, 50, 50));//일요일 - 배경색을
빨간색으로 설정
}
else if (i == 6) {
weekDaysName[i].setBackground(new Color(50, 100, 200));//토요일 - 배경색을
파란색으로 지정
}
else {
weekDaysName[i].setBackground(new Color(150, 150, 150));//평일 배경색회색
}
weekDaysName[i].setOpaque(true);//불투명하게설정
weekDaysName[i].setFocusPainted(false);//테두리 설정안함
calPanel.add(weekDaysName[i]);//weekDaysName 패널추가
}
```

• 날짜를 입력하면 해당 날짜의 요일을 출력해주는 이벤트처리

```
checkBu.addActionListener(new ActionListener(){//버튼 누를시 이벤트처리
public void actionPerformed(ActionEvent arg0) {
try {
String get_date = dayfield.getText();//Textfiled를 String형식으로 바꿈
Date date = new SimpleDateFormat("yyyy-MM-dd").parse(get_date);//String을
Date형식으로 바꿈.
cal = Calendar.getInstance();//cal 인스탠스처리
cal.setTime(date);//받은 일정을 claendar set.
```

```
switch (cal.get(Calendar.DAY_OF_WEEK)) {//Day_of_week 따른 요일switch
		case 1:
		day = "일";
		break;
		case 2:
		day = "월";
		break;
		case 3:
		day = "화";
		break;
		case 4:
		day = "수";
		break;
		case 5:
		day = "목";
		break;
		case 6:
		day = "금";
		break;
		case 7:
		day = "토";
		break;
}
bottomInfo.setText(day+"요일 입니다.");//요일 출력

}
catch(Exception e) {
ottomInfo.setText("yyyy-mm-dd 형식으로 작성해주세요.");//예외처리 - 작성 형식 요청
}
}
});
```

• 스케줄 입력창에 Save버튼 클릭시 이벤트 처리

```
saveBut.addActionListener(new ActionListener(){//save버튼 클릭시
public void actionPerformed(ActionEvent arg0) {
try {
File f= new File("MemoData");//MemoDate라는 file생성
```

```
if(!f.isDirectory()) f.mkdir();//디렉토리가 없을경우 디렉토리생성

String memo = memoArea.getText();//memoArea에있는 메모추출

if(memo.length()>0){//memo에 글이 있을경우
BufferedWriter out = new BufferedWriter(new
FileWriter("MemoData/"+calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDayOfMo
n<10?"0":"")+calDayOfMon+".txt"));
//Buffer형식으로 현재날짜의 날짜의 파일생성
String str = memoArea.getText();//String형식으로 memo출력
out.write(str); //buffer형식의 파일에 memo저장
out.close();//다쓴후 종료

        bottomInfo.setText(calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDayOf
Mon<10?"0":"")+calDayOfMon+".txt"+SaveButMsg1);

//완료후 저장메세지출력
}
else {
        bottomInfo.setText(SaveButMsg2);//메모작성요청 메세지 출력
        }

catch (IOException e) {
bottomInfo.setText(SaveButMsg3);//예외처리
}
showCal(); //종료후 다시 달력출력
}
});
```

• 스케줄입력창에 Delete버튼 클릭시 이벤트 처리

```
delBut.addActionListener(new ActionListener(){//메모삭제 이벤트발생
public void actionPerformed(ActionEvent e) {
memoArea.setText("");//메모초기화
File f =new
File("MemoData/"+calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDayOfMon<10?"
0":"")+calDayOfMon+".txt");
//생성되어있는 파일 초기화
if(f.exists()){//파일존재하면
```

```
f.delete();//파일삭제
showCal();//달력출력
            bottomInfo.setText(DelButMsg1);//메모삭제 메세지 출력
}
else
{
bottomInfo.setText(DelButMsg2);        //파일존재하지 않을때의 메세지 출력
}
});
```

• 파일에서 공휴일과 24절기, 스케줄의 메시지를 가져와 메모에 표시하는 함수

```
private void readMemo(){//파일에서 메모를 가져오는 메소드
try{
File f = new
File("MemoData/"+calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDayOfMon<10?"
0":"")+calDayOfMon+".txt");
//해당날짜 파일 받아오기
File holly = new
File("HollyDate/"+calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDayOfMon<10?"0
":"")+calDayOfMon+".txt");
//공휴일 파일 받아오기
File season = new
File("SeasonalDate/"+calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDayOfMon<1
0?"0":"")+calDayOfMon+".txt");
//24절기 파일 받아오기
if(holly.exists()) {
BufferedReader hollyin = new BufferedReader(new
FileReader("HollyDate/"+calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDayOfMo
n<10?"0":"")+calDayOfMon+".txt"));
//buffer단위로 가져옴
String hollyText = new String();
//파일 내용을 저장할 string 초기화
while(true) {
String temStr = hollyin.readLine();//라인단위로가져옴
if(temStr == null) break;//끝날때까지 반복
hollyText = hollyText + temStr + System.getProperty("line.separator");//원래의
```

```
데이터와 현재데이터 메시지저장
}
memoArea.setText(hollyText); // memo창에 보이기하기
hollyin.close();//파일 종료
}
if(season.exists()) {
BufferedReader Seasonin = new BufferedReader(new
FileReader("SeasonalDate/"+calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDayO
fMon<10?"0":"")+calDayOfMon+".txt"));
//buffer단위로 파일가져오기
String SeasonText = new String();
//파일 내용을 저장할 string
while(true) {
String temStr = Seasonin.readLine();//line단위로가져옴
if(temStr == null) break;//끝날때 ㅏ지 반복
SeasonText = SeasonText + temStr + System.getProperty("line.separator");//원래의
데이터와 현재데이터 메시지저장
}
memoArea.setText(SeasonText);// memo창에 보이기하기
Seasonin.close();//파일 종료
}
if(f.exists()){//파일이 존재할경우
BufferedReader in = new BufferedReader(new
FileReader("MemoData/"+calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDayOfM
on<10?"0":"")+calDayOfMon+".txt"));
//buffer단위로 파일출력
String memoAreaText= new String();
//String생성
while(true){
String tempStr = in.readLine();//line단위로 입력
if(tempStr == null) break;//다입력한경우 종료
memoAreaText = memoAreaText + tempStr + System.getProperty("line.separator");
//파일에 저장위치 가져와서 입력한 글과 text저장
}
memoArea.setText(memoAreaText);//일정을 입력한 메시지 set, 저장
in.close();        //file입력 후 종료
```

```
}
catch (IOException e) {//예외처리
e.printStackTrace();//예외정보 출력                }
        }
```

• 달력 출력 그래픽 메소드

```
private void showCal(){ //달력 출력 graphic
for(int i=0;i<CAL_HEIGHT;i++){
for(int j=0;j<CAL_WIDTH;j++){
String fontColor="black"; // 평일 날짜 글씨 색 블랙
if(j==0) fontColor="red";// 일요일 날짜 글씨색 빨강
else if(j==6) fontColor="blue"; // 토요일 날짜 글씨색 파랑

File f =new
File("MemoData/"+calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDates[i][j]<10?"0
":"")+calDates[i][j]+".txt");
File holly = new
File("HollyDate/"+calYear+((calMonth+1)<10?"0":"")+(calMonth+1)+(calDates[i][j]<10?"0"
:"")+calDates[i][j]+".txt");
//해당 날짜의 일정 파일 확인

if(f.exists()){
fontColor = "Green"; // 일정이 있는 경우 날짜를 초록색으로 지정
dateButs[i][j].setText("<html><b><font
color="+fontColor+">"+calDates[i][j]+"</font></b></html>");
}else if(holly.exists()){//공휴일인경우 날짜 빨간색으로 지정
fontColor = "red";
dateButs[i][j].setText("<html><b><font
color="+fontColor+">"+calDates[i][j]+"</font></b></html>");
}
lse dateButs[i][j].setText("<html><font
color="+fontColor+">"+calDates[i][j]+"</font></html>");
//평일인 경우 검정색으로 출력
JLabel todayMark = new JLabel("<html><font >*</html>"); //오늘이면 *라벨 추가
dateButs[i][j].removeAll();
if(calMonth == today.get(Calendar.MONTH) &&
calYear == today.get(Calendar.YEAR) &&
```

```
calDates[i][j] == today.get(Calendar.DAY_OF_MONTH)){ //if문을 통해서 오늘 조건확인
dateButs[i][j].add(todayMark); // todayMark '*'달력에 추가
        dateButs[i][j].setToolTipText("Today");//(today)메모입력                    }

if(calDates[i][j] == 0) dateButs[i][j].setVisible(false); // 빈칸 입력(달력빈칸) 보여주지않기
else dateButs[i][j].setVisible(true); // 해당 달의 날이면 표시하기
}
}
}
```

• 국경일을 HashMap으로 받아서 File처리(내용상 2019년의 국경일만 포함)

```
public void hollyDay() throws IOException {//국경일 휴일
File f= new File("HollyDate");//hallyDate라는 file생성
if(!f.isDirectory()) f.mkdir();//디텍토리가 없을경우 디텍토리생성
FileWriter fw;
hollyMap = new HashMap<String, String>();
// 공휴일을 저장할 hashmap key값으로 날짜 value값으로 공휴일 저장
// 2010~2028년까지의 공휴일 저장

//2019년 공휴일 hollyMap에 해당 날짜 key값으로 저장, value값 공휴일 저장.
hollyMap.put("20190101.txt", "신년");
hollyMap.put("20190204.txt", "설날");
hollyMap.put("20190205.txt", "설날");
hollyMap.put("20190206.txt", "설날");
hollyMap.put("20190301.txt", "삼일절");
hollyMap.put("20190505.txt", "어린이날");
hollyMap.put("20190512.txt", "석가탄신일");
hollyMap.put("20190606.txt", "현충일");
hollyMap.put("20190815.txt", "광복절");
hollyMap.put("20190912.txt", "추석");
hollyMap.put("20190913.txt", "추석");
hollyMap.put("20190914.txt", "추석");
hollyMap.put("20191003.txt", "개천절");
hollyMap.put("20191009.txt", "한글날");
hollyMap.put("20191225.txt", "크리스마스");
Set<String> keySet = hollyMap.keySet();//hashmap의 key값만 받을 set
Iterator<String> keyIterator = keySet.iterator();//hashmap의 반복자얻기
```

```
while(keyIterator.hasNext()) {//저장된 객체수 만큼 루핑
String key = keyIterator.next();//해당 Set에서 key값을 받아옴
f = new File("HollyDate", key);//'hollydate'라는 파일에 key값의 파일명으로 설정
fw = new FileWriter(f);//파일을 쓸 filewriter
String value = hollyMap.get(key);//해당 키의 밸류값을 저장
fw.write(value);//f파일에 value값을 씀
fw.flush();//다 쓴후 비우기작업
}
}
```

• 24절기를 HashMap으로 받아서 File처리(내용상 2019년의 24절기만 포함)

```
public void Seasonal() throws IOException {//24절기
File sf= new File("SeasonalDate");//SeasonalDate라는 file생성
if(!sf.isDirectory()) sf.mkdir();//디텍토리가 없을경우 디텍토리생성
FileWriter fw1;
SeasonMap = new HashMap<String, String>(); // 절기를 저장할 key값 날짜, value값 24절기 명칭을 저장할 hashmap
// 24절기를 저장할 hashmap key값으로 날짜 value값으로 공휴일 저장
// 2017~2022년까지의 24절기 저장
//2019년 24절기, SeasonMap의 hashMap에 key : 날짜.txt, value : 24절기 저장
SeasonMap.put("20190204.txt", "입춘");
SeasonMap.put("20190219.txt", "우수");
SeasonMap.put("20190306.txt", "경칩");
SeasonMap.put("20190321.txt", "춘분");
SeasonMap.put("20190405.txt", "청명");
SeasonMap.put("20190420.txt", "곡우");
SeasonMap.put("20190506.txt", "입하");
SeasonMap.put("20190521.txt", "소만");
SeasonMap.put("20190605.txt", "망종");
SeasonMap.put("20190622.txt", "하지");
SeasonMap.put("20190707.txt", "소서");
SeasonMap.put("20190723.txt", "대서");
SeasonMap.put("20190808.txt", "입추");
SeasonMap.put("20190823.txt", "처서");
SeasonMap.put("20190908.txt", "백로");
SeasonMap.put("20190923.txt", "추분");
```

```
SeasonMap.put("20191008.txt", "한로");
SeasonMap.put("20191024.txt", "상강");
SeasonMap.put("20191108.txt", "입동");
SeasonMap.put("20191122.txt", "소설");
SeasonMap.put("20191207.txt", "대설");
SeasonMap.put("20191222.txt", "동지");
SeasonMap.put("20200106.txt", "소한");
SeasonMap.put("20200120.txt", "대한");
Set<String> keySet = SeasonMap.keySet();//key값을 받아올 Set자료구조
Iterator<String> keyIterator = keySet.iterator();//반복자얻기
while(keyIterator.hasNext()) {//객체수만큼루핑
String key = keyIterator.next();//해당 set의 key값을 받아옴
sf = new File("SeasonalDate", key);//SeasonalDate위치에 key값의 파일명 저장
fw1 = new FileWriter(sf);//해당하는 file에 writer명령
String value = SeasonMap.get(key);//해당 key값에대한 value값저장
fw1.write(value);//value값 쓰기
fw1.flush();//다쓴후 비우기
}
}
```

• 달과 연도의 이동 버튼을 눌렀을 때의 이벤트처리

```
private class ListenForCalOpButtons implements ActionListener{ //달과 연도 이동
이벤트처리class
public void actionPerformed(ActionEvent e) {
if(e.getSource() == todayBut){//오늘 버튼눌렀을때
setToday(); // 오늘 메소드 실행하여 오늘 날짜로 감
lForDateButs.actionPerformed(e);//이벤트처리
focusToday();//오늘 달력으로 넘어감
memoArea.setText(null);//이동할때 메모장 초기화
bottomInfo.setText(" ");//하단 info 초기화
}
else if(e.getSource() == lYearBut) {
moveMonth(-12);//'<<'눌렀을때 이전 년도로넘어감
memoArea.setText(null);//이동할때 메모장초기화
bottomInfo.setText(" ");//하단 info 초기화
}else if(e.getSource() == lMonBut) {
```

```
moveMonth(-1);// '<'눌렀을때 이전 달로 넘어감
memoArea.setText(null);//이동할때 메모장초기화
bottomInfo.setText(" ");//하단 info 초기화
}
else if(e.getSource() == nMonBut) {
moveMonth(1); // '>' 눌렀을때 다음 달로 넘어감
memoArea.setText(null);//이동할때 메모장 초기화
bottomInfo.setText(" ");//하단 info 초기화
}
else if(e.getSource() == nYearBut) {
moveMonth(12);//'>>' 눌렀을 때 다음 년도로 넘어감
memoArea.setText(null);//이동할때 메모장초기화
bottomInfo.setText(" ");//하단 info 초기화
}

curMMYYYYLab.setText("<html><table width=100><tr><th><font size=5>"+calYear+" /
"+((calMonth+1)<10?" ":"")+(calMonth+1)+"</th></tr></table></html>");
//해당 연도와 월을 보여줄 수 있는 표시 ,달 이동할때 년월표시
showCal();//달력실행
}
}
```

• 'Today'버튼을 눌렀을 때 이벤트처리.

```
private class listenForDateButs implements ActionListener{ //오늘 버튼 눌렀을때
이벤트처리, Dday처리
public void actionPerformed(ActionEvent e) { //오늘 버튼눌렀을때
int k=0,l=0;
for(int i=0 ; i<CAL_HEIGHT ; i++){
for(int j=0 ; j<CAL_WIDTH ; j++){
if(e.getSource() == dateButs[i][j]){
k=i;
l=j;
//이전의 i같과 j값을 k와 l값 today눌렀을때 확인용으로 저장
}
}
}
if(!(k ==0 && l == 0)) calDayOfMon = calDates[k][l]; //today버튼을 눌렀을때도 이
```

actionPerformed함수가 실행되기 때문에 넣은 부분

```
cal = new GregorianCalendar(calYear,calMonth,calDayOfMon);//한국날짜가져오기
        String dDayString = new String();
int dDay=((int)((cal.getTimeInMillis() -
today.getTimeInMillis())/1000/60/60/24));//해당날짜와 현재날짜를 빼서 Dday설정
if(dDay == 0 && (cal.get(Calendar.YEAR) == today.get(Calendar.YEAR))
&& (cal.get(Calendar.MONTH) == today.get(Calendar.MONTH))
&& (cal.get(Calendar.DAY_OF_MONTH) == today.get(Calendar.DAY_OF_MONTH)))
dDayString = "Today";
//cal와 today의 값이 같으면 Today저장
else if(dDay >=0) dDayString = "D-"+(dDay+1);
//dday가 0보다 같거나 크면 D-로 출력
else if(dDay < 0) dDayString = "D+"+(dDay)*(-1);
//dday가 0보다 작으면 D+로 출력
selectedDate.setText("<Html><font
size=3>"+calYear+"/"+(calMonth+1)+"/"+calDayOfMon+" ("+dDayString+")</html>
");
//해당 날짜와 현재로부터의 dday표시
bottomInfo.setText(" ");
readMemo(); // 일정 메모메소드 실행
}
```

• Thread를 이용하여 현재시간을 다중 프로세스화하여 달력창에 표시

```
private class ThreadConrol extends Thread{ //현재시간 보여주는 thread
public void run(){
boolean msgCntFlag = false;
int num = 0;//buttonInfo.getText() 출력 number
String curStr = new String();// getText() check용
while(true){//while문으로 무한반복
try{
today = Calendar.getInstance();//오늘 날짜 받아옴
String amPm = (today.get(Calendar.AM_PM)==0?"AM":"PM");//0일때 오전 1일때 오후
String hour;
if(today.get(Calendar.HOUR) == 0) hour = "12"; //0일때 12시
else if(today.get(Calendar.HOUR) == 12) hour = " 0"; // 12시 일때0시로
```

```
Am시간단위표시
else hour = (today.get(Calendar.HOUR)<10?" ":"")+today.get(Calendar.HOUR);//
String min =
(today.get(Calendar.MINUTE)<10?"0":"")+today.get(Calendar.MINUTE);//현재분저장
String sec =
(today.get(Calendar.SECOND)<10?"0":"")+today.get(Calendar.SECOND);//현재 초저장
infoClock.setText(amPm+" "+hour+":"+min+":"+sec); //시간표시
sleep(1000);//1000mmtime후 종료

String infoStr = bottomInfo.getText();//infostr에 저장

if(infoStr != " " && (msgCntFlag == false || curStr != infoStr)){//infostr가 있을경우
curStr저장
        num = 5;
        msgCntFlag = true;
        curStr = infoStr;
}
else if(infoStr != " " && msgCntFlag == true){//num이 줄어들면서 text없애는 if문
        if(num > 0) num--;
        else{
        msgCntFlag = false;
        bottomInfo.setText(" ");
}}                                          }
catch(InterruptedException e){//쓰레드 예외처리
        System.out.println("Thread:Error");
        }
}}}}
```

4. 최종 결과물 장점 설명

• GUI를 이용하여 작업한 달력이므로 User가 편리하게 마우스와 키보드로 달력의 기능을 사용할 수 있다.

• 현재 보이는 달에서 '<', '>' 버튼을 누르면 이전 달의 달력과 다음 달의 달력을 쉽게 볼 수 있으며 '<<', '>>' 버튼을 누르면 이전 연도의 달력과 다음 연도의 달력을 User가 쉽게 볼 수 있다.

• 좌측 상단에 ‘Today’버튼을 누르면 지금 보이는 달력이 현재의 달력과 멀리 떨어진 달력이라도 현재의 달력을 보여준다.

• 현재 보여지는 달력 위에 그 해당 연도와 달이 표시되어 있어서 해당 달력의 연도와 월을 쉽게 알 수 있다.

• 좌측 상단 ‘Today’ 우측에 현재 날짜가 표시되어 있어서 User가 쉽게 현재 날짜를 알 수 있다.

• 상단 중앙에 유저가 보고 싶은 달력을 YYYY-mm형식으로 입력하여 ‘Move’버튼을 누르면 해당 달력을 볼 수 있다.

• 평일은 검정색, 토요일은 파란색, 일요일은 빨간색의 글씨로 달력에 표시되어 있어서 User가 쉽게 주중과 일요일, 토요일 구분이 가능하다.

• 공휴일이 달력에 저장되어있어 공휴일의 해당 날짜를 클릭하면 ‘Memo’창에서 공휴일을 알 수 있고 해당 날짜가 빨간색으로 표시되어있어서 User가 쉽게 공휴일을 알 수 있다.

• 24절기가 달력에 저장되어있어 24절기의 해당 날짜를 클릭하면 ‘Memo’창에서 24절기를 알 수 있고 해당 User가 쉽게 24절기를 알 수 있다.

• 해당 날짜에 스케줄을 ‘Memo’칸에 입력 할 수 있고 ‘Save’버튼을 누르면 스케줄이 저장되며 해당 날짜의 색깔이 초록색으로 바뀌어 스케줄 있는 해당 날짜가 표시되며 User가 해당날짜에 대한 스케줄을 확인 할 수 있다. 또한 ‘Delete’버튼을 누르면 User가 저장했던 스케줄이 삭제되며 날짜의 색도 기본 색깔 바뀐다. ‘Clear’버튼을 통해 Memo장을 초기화 할 수 있다.

• 우측 상단에 현재 시간이 표시되어 User가 쉽게 현재 시간을 확인 할 수 있다.

• 해당 날짜를 클릭하면 ‘Memo’칸에 해당 날짜로 바뀌며 현재 날짜로 부터의 D-day를 알 수 있다.

• 해당 날짜를 클릭하면 해당 날짜에 네모난 박스가 생성되어 User가 해당 날짜로 이동을

확인 할 수 있다.

• 달력에서 현재 날짜는 초록색의 '*'로 표시되어 달력에서 쉽게 요일과 날짜를 확인 할 수 있다.

• User의 잘못된 명령을 달력창의 하단 중앙에 예외처리 message가 출력되어 옳은 명령 방법을 확인 할 수 있다.

• 버튼위에 마우스 커서를 올려두면 해당 버튼에 대한 도움말이 풍선형태의 message가 출력되어 User가 쉽게 기능을 알 수 있다.

• User의 명령에 대한 성공/실패 message가 하단 중앙에 표시되어 User가 명령 성공/실패를 알 수 있다.

• 해당 날짜의 요일을 알고 싶을 때 우측 메모장위에 칸에 (yyyy-mm-dd)형식으로 해당 날짜를 입력하면 해당 날짜에 요일이 중앙 하단에 표시된다.