Schedule Management System

– Object Oriented Programming –

팀 장	20206434 염경하
팀 원	20215725 고하영
팀 원	20202850 김지민
팀 원	20212571 이지수
팀 원	20214466 정재일
팀 원	20201776 허 인

I. Brief Information on Project

1. Summary on Project

The goal of this project is to make a program for implementing the schedule management system, especially for students. As the program consists of managing schedules for assignments and exams, its main target is students though anyone could use this if they want. On the program, the user simply could make an input about the information on the assignment or the exam, then look at the overview of its own schedule on the screen.

2. Compile & Execution

The program was designed using JAVA. To compile and execute this program, you should do the following.

(1) Download json-simple.1.1.1.jar on the browser.
(2) Open the Package(Folder) OOP_proj4 using JAVA Eclipse.
(3) Go to the Package Explorer, which is on the left side of the screen and then click the Package using the right button of the mouse.
(4) Go to Properties and click Java Build Path on the left side.
(5) Go to menu Libraries and click Classpath
(6) Click the menu Add External JARs on the right side, and add the json-simple-1.1.1.jar that was downloaded in the previous time.
(7) Go to(Double Click) file SaveInfo.java
(8) Click Run button on the top of the screen

The information on the specific implementation of the program would be explained below.

II. Design about Program Structure

1.Functionality of the Program

The overall program works like the following. The user makes an input on the information on assignment or exam. Information for assignment are subject name, title(title of the assignment), date(due date, submission date, end date), importance, difficulty and memo for detailed information. Information of exam is similar except for the title. Instead of the title, the user should make an input of the test range.

After the user makes an input of the information, the user can see the whole information in two ways. First way is a calendar, which is a visual thing. If the user clicks a certain date on the calendar on the screen, it displays a task(exam or assignment) corresponding to the date.

Second method is to see the task just listing by letters, showing all the information of the task. List part shows the information of assignment and exam separately. There's a 'Enter subject name' on the screen.

If the user makes an input of the subject name, it shows the information corresponding to the name, for both assignment and exam. The program was designed to be case-sensitive, so that the user should be careful of upper and lower case when making an input of the name at the first time to add the information, and also when to search information. If a user clicks a 'D-DAY' button, it shows the subject and its information in the order which should be handled first(which has less time to the due date). If the user clicks it one more time, it would show the information in the reverse, and would change the order of it again whenever clicking it. It applies the same to the PRIORITY and DIFFICULTY menu. If the user clicks PRIORITY, it shows the task in the order of having much more priority. If the user clicks DIFFICULTY, the program shows the task in the order of having much difficulty. And if the user clicks it again, it would show the result in the reverse order for both PRIORITY and DIFFICULTY.

2.Brief information on the source code

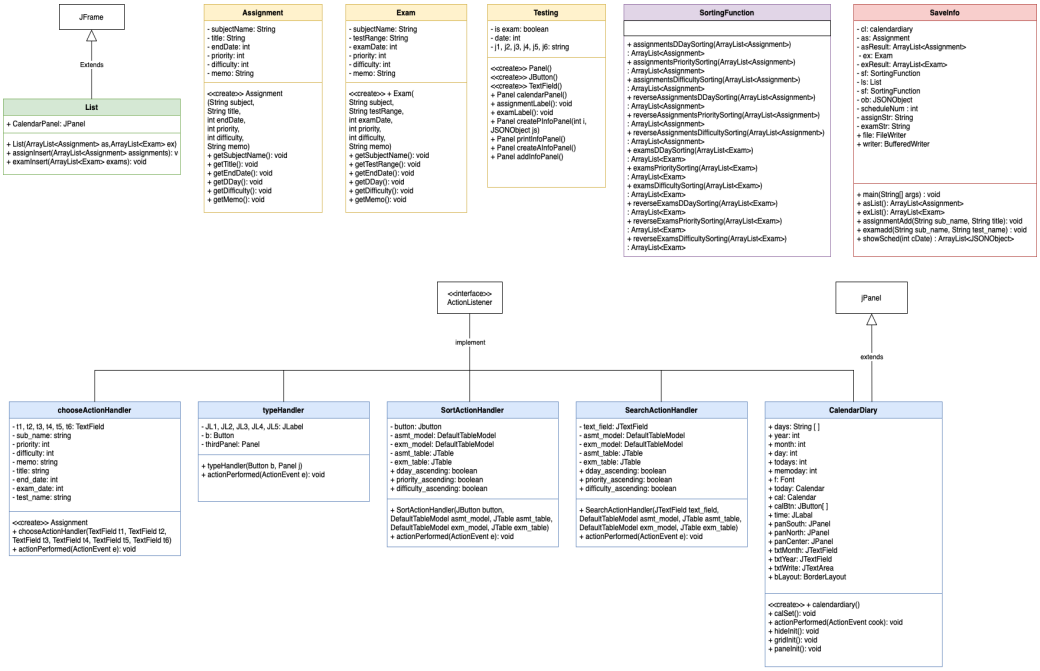
The program for the project consists of 7 files, (1)Assignment.java (2)calendardiary.java (3)Exam.java, (4)List.java (5)SaveInfo.java (6)SortingFunction.java and (7)Testing.Java. Each file has the following role.

(1) Assignment.java: This corresponds to the header file in c++. This contains a getter and a. constructor that has the attributes required for the assignment and can obtain the required attribute values.
(2) calendardiary.java: This provides a Calendar parts' calendar GUI. When clicking the date button, it sends the date information to Testing.java. Testing.java can print the information on the date.
(3) Exam.java: This is also a class that has the properties necessary for examination and has a getter and a constructor.
(4) List.Java file: This handles the overall GUI of our program, especially the 'List' part.
(5) Saveinfo.java: This is a class which manages the JSON file information.
(6) sortingFunction.java: This is a class that helps you sort assignments and exams in the order of d-day, priority, and difficulty.
(7) Testing.java: This handles the overall GUI of our program, especially the 'Calendar' part. And it is a class which also can put some information to a JSON file. Input information can be sent to SaveInfo.java which can help the program to work properly.

III. Module Specification

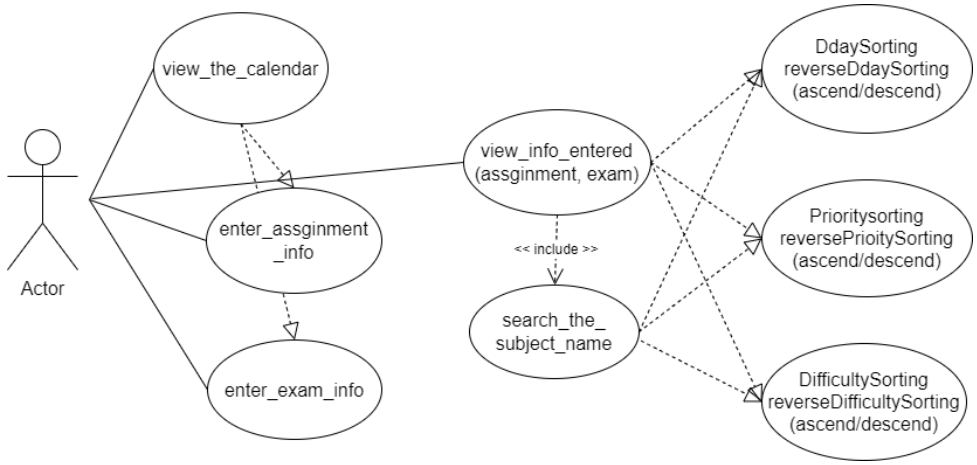
Before looking at the detailed information on each file(class), here is brief information about the class we made using a diagram. These are class diagram, use-case diagram and activity diagram.

1. class diagram



Class diagram describes the overall structure of the system. It displays objects, attributes, and methodsbelonging to each class. It also represents an inheritance relationship between classes. Assignment, Exam, and Testing class manage the data needed for the project. The 'extends' inheritance relationship between the class and JFrame, JPanel used in UI was indicated. and The 'implements' inheritance relationship between ActionLister and several classes we made is indicated in the class diagram.

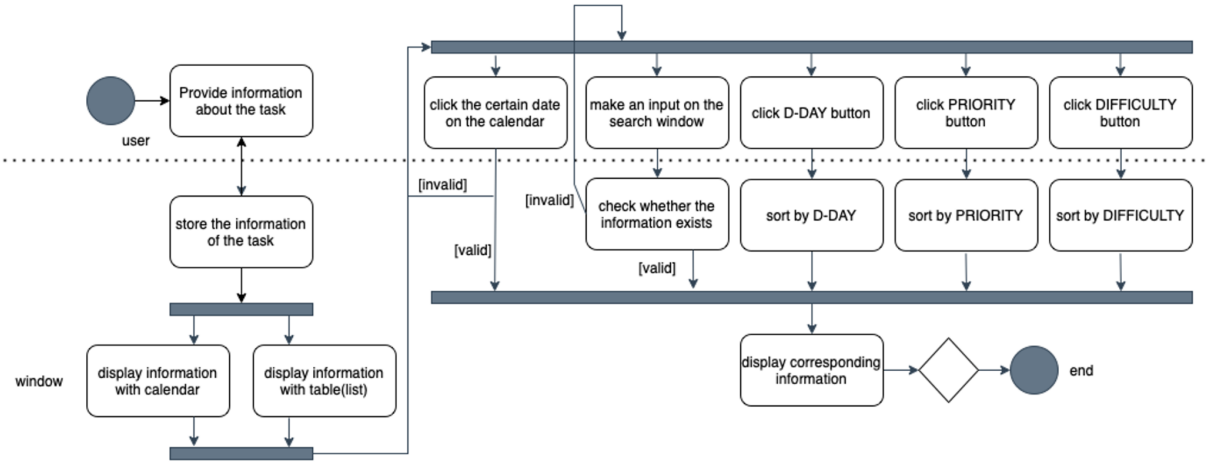
2. use-case diagram



The use-case diagram shows the functions provided by the system with respect to the user, indicates their goals as use-cases, and indicates dependencies between use-cases. The user can enter the assignment and exam information, check the entered information through the calendar, and view it through the window by pushing the list button in another way. The information may be arranged in

ascending or descending order based on d-day, priority, and difficulty. In addition, the same task can be performed with items searched based on the subject name.

3. activity diagram



The activity diagram displays the overall workflow of the program. If the user provides information about the task, by making an input, the window displays that information in 2 ways, calendar or table. Then, the user could do whatever he wants. He could click the certain date on the calendar, or make an input on the search window or click the button for sorting the data. When the user clicks the certain date on the calendar, it displays corresponding information when there's corresponding information. Otherwise, first. If there's a task on the date, it would display corresponding information, otherwise the user could do actions mentioned above. If the user input corresponds to the information stored, it would display corresponding information. Otherwise, the user could do actions mentioned above like the previous case. If a user clicks a button, it sorts the data by the button that the user clicked, and the window displays the corresponding information in sorted order.

IV. Implementation

1. Assignment.java

```
public class Exam {

    private String subjectName;
    private String testRange;
    private int examDate;
    private int priority;
    private int difficulty;
    private String memo;

    //getter
    public String getSubjectName() {
        return this.subjectName;
    }
```

The file Assignment.java has class Assignment. Class Assignment has the attributes shown above(subjectName, title, endDate, priority, difficulty, memo). Each attribute does each task corresponding to its name. Each means, the name of the subject that the assignment is given, title of the

assignment, submission date of the assignment, priority of the assignment which means how much the assignment is important, how difficult the assignment is, and the memo for detailed information on the assignment.

The class declares the attributes in the form of private, so the program uses a function to get the attribute, which corresponds to 'getter'. There's only one getter function on the screenshot of the code, but there are several getter functions which work also for other attributes. The report just shows only one case of getter function in short. endDate is especially needed to calculate the d-day which displays the remaining days to the due date. The program derives the D-day = endDate - today's date. The class also contains a constructor that works when a user makes a new input.

2. calendardidary.java

```
class calendardiary extends JPanel implements ActionListener
{
	String [] days = {"SUN","MON","TUE","WED","THU","FRI","SAT"};
	int year ,month,day,todays,memoday=0;

	Font f;

	Calendar today;
	Calendar cal;


	JButton[] calBtn = new JButton[49];

	JLabel time;

	JPanel panSouth;
	JPanel panNorth;
	JPanel panCenter;

	JTextField txtMonth,txtYear;

	JTextArea txtWrite;
	BorderLayout bLayout= new BorderLayout();

	Connection con = null;
	Statement stmt = null;
```

The calendardiary file stores information about the calendar and manages the GUI of the calendar. The calendardiary class has member variables related to the calendar panel, text, and font. Also, there are several member functions. The calendardiary function uses the default time zone and locale to get the calendar. Gregorian Calendar is Calendar's conceptual subclass and provides a standard calendar system. The panel is then divided into east, west, north, and south, and each calendar is assembled on the panel. Create a function to create and close the part corresponding to the calendar day and then position the frame window. The calcset function sets the calendar. It adds color when the calendar represents Sunday and Saturday. Also, the days when the schedule was saved were colored. Since we have to calculate after taking the day of the week, we added the number of buttons that took the day of the week and calculated the value with a value subtract 1 since the index starts from 0. The actionPerformed function converts the value of the button stored in character form into an integer when the button on the calendar is pressed to change the date you clicked. The hiddenInit function disables the rest of the buttons without a schedule. The girdInit function numbers the calendar panels. The panelInit function divides the panel

layout using a grid. The calInput function calculates the passing of the year by setting the standard as 12 months.

3. Exam.java

```
public class Exam {

    private String subjectName;
    private String testRange;
    private int examDate;
    private int priority;
    private int difficulty;
    private String memo;

    //getter
    public String getSubjectName() {
        return this.subjectName;
    }
```

The file Examt.java has class Exam. Class Exam has the same structure with the class Assignment except for the one attribute, testRange. Instead of using the title, which means the title of the assignment, class Exam contains an attribute testRange which shows the coverage of the content to which the user should study for the exam.

The class also has a getter similar to the class Assignment mentioned above, to get the attribute. The screenshot of the code on this class also shows the getter function just for the name of the subject, but a getter function also holds for other attributes. Capturing all the code on the report would be efficient, so the report only has one case. There's also a constructor which creates the instance of the class when the user makes an input of the information of the task(which are the attributes of the class).

4. List.java

Class SortActionHandler deals with the action when the user clicks the button D-DAY, PRIORITY or DIFFICULTY for sorting. The class has a constructor, which contains the parameter button, asmt_model and asmt_table.

The parameter button refers to the button which regards sorting, like D-DAY, PRIORITY and DIFFICULTY. asmt_table is a table to display the information of the assignment, and asmt_model regards the management of the table. exm_model and exm_table also goes the same with the asmt_table and asmt_model, which is the table for displaying information of exams and the one which works for its management. The program was also designed to print out the information of the task in decreasing or increasing order whenever clicking the button.

```
public SortActionHandler(JButton button, DefaultTableModel asmt_model, JTable asmt_table,
        DefaultTableModel exm_model, JTable exm_table) {

    this.button = button;
    this.asmt_model = asmt_model;
    this.asmt_table = asmt_table;
    this.exm_model = exm_model;
    this.exm_table = exm_table;
}
```

The class holds the function name actionPerformed(ActionEvent e). It handles sorting the information. The function first checks what button is pressed by the user. If the D-DAY button is pressed by the user, then it sorts the data(information of the task) in the order of which has small value on D-DAY. There's a variable with the type of bool that displays the data in the reverse order when the user clicks again. If the bool holds True, then it shows the data in increasing order, otherwise the decreasing order. When the variable ascending = True, the program uses DDaySortingFunction. In the opposite case, when the variable ascending = False, then the program uses a function named reverseDDaySorting. It goes the same for the PRIORITY and DIFFICULTY button. When they are pressed by the user, it also works similarly with the D-DAY button for sorting the data, just mentioned above.

There's another class named SearchActionHandler, which regards the action when the user clicks the button Search.

```
public SearchActionHandler(JTextField text_field, DefaultTableModel asmt_model,
        JTable asmt_table, DefaultTableModel exm_model, JTable exm_table) {
    this.text_field = text_field;
    this.asmt_model = asmt_model;
    this.asmt_table = asmt_table;
    this.exm_model = exm_model;
    this.exm_table = exm_table;
}
```

The class has a constructor which has quite similar parameters with the constructor of the class SortActionHandler. The only difference is text_field instead of the parameter button. text_field refers to the window for searching the information of the task by using the keyword(name of the subject). Other parameters have the same role as the constructor of the class SortActionHandler. If a user makes an input of the name of the subject on the window, that input is passed to the class SortingFunction and then being checked whether the input that the user made corresponds to the data originally existed.

```
ArrayList<Assignment> resultAssignment = SortingFunction.searchAssignment(assignments, subject);
ArrayList<Exam> resultExam = SortingFunction.searchExam(exams, subject);
```

If the input that the user made is the same with the information of the data, especially the name of the subject, then it shows that data.

The next class is class List, which creates an overall User Interface(UI).

```
CalendarPanel = new JPanel();
CalendarPanel.setBackground(new Color(207, 227, 250));
tab.add("Calendar", CalendarPanel);
CalendarPanel.add(Testing.calendarPanel());
CalendarPanel.add(Testing.printInfoPanel());
CalendarPanel.add(Testing.addInfoPanel());
```

The class first makes an entire frame for UI, and creates the window for the calendar like the above.

```
JPanel inputPanel = new JPanel();
inputPanel.setBackground(Color.WHITE);
JButton search_button = new JButton("SEARCH");
search_button.setBackground(new Color(207, 227, 250));
search_button.setLocation(130, 50);
JTextField text_field = new JTextField(20);
inputPanel.add(new JLabel("Enter subject name."));
inputPanel.add(text_field);
inputPanel.add(search_button);
ListPanel.setLocation(0, 30);
ListPanel.add(inputPanel);
```

The above one is the source code which handles the UI for Search, which displays the information of the task just by letters not in a visual way.

```
JPanel sortPanel = new JPanel();
sortPanel.setBackground(Color.WHITE);

JButton dday_button = new JButton("D-DAY");
dday_button.setBackground(new Color(207, 227, 250));
sortPanel.add(dday_button);
```

The above one is also part of the UI of search, but specifically which deals with the button for sorting the data. sortPanel is created, and it holds the button D-DAY, like described above. The report just made a screenshot on the code for the D-DAY, but it also goes the same with the PRIORITY and DIFFICULTY.

```
DefaultTableModel asmt_model = new DefaultTableModel(asmt_contents, asmt_header);
JTable asmt_table = new JTable(asmt_model);
JScrollPane asmt_scroll = new JScrollPane(asmt_table);
asmt_scroll.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
AssignmentPanel.add(asmt_scroll);
asmt_scroll.setPreferredSize(new Dimension(1100, 300));
ListPanel.add(AssignmentPanel, BorderLayout.CENTER);
```

Creating the window area for displaying the information of the task is shown above. To show the information in the form of a table, the program used JTable. It also adds table and scroll bar by creating the AssignmentPanel.

```
ArrayList<Assignment> assignments = new ArrayList<Assignment>();
assignments = as;

for (int i = 0; i < assignments.size(); i++) {
    String inputStr[] = new String[6];
    inputStr[0] = assignments.get(i).getSubjectName();
    inputStr[1] = assignments.get(i).getTitle();
    inputStr[2] = Integer.toString(assignments.get(i).getDDay());
    inputStr[3] = Integer.toString(assignments.get(i).getPriority());
    inputStr[4] = Integer.toString(assignments.get(i).getDifficulty());
    inputStr[5] = assignments.get(i).getMemo();
    asmt_model.addRow(inputStr);
}
```

Source code on the above just regards the display of the data on the window, as soon as the program compiles. This deals with the data stored in the previous time. If the user makes an input and then terminates the program, the program shows the information of the task(data being stored) when the user executes the program at the next time. Displaying the information of both assignment and task applies to the explanation mentioned above in this List.java part.

```
search_button.addActionListener(new SearchActionHandler(text_field, asmt_model, asmt_table, exm_model, exm_table));
text_field.addActionListener(new SearchActionHandler(text_field, asmt_model, asmt_table, exm_model, exm_table));

dday_button.addActionListener(new SortActionHandler(dday_button, asmt_model, asmt_table, exm_model, exm_table));
priority_button.addActionListener(
        new SortActionHandler(priority_button, asmt_model, asmt_table, exm_model, exm_table));
difficulty_button.addActionListener(
        new SortActionHandler(difficulty_button, asmt_model, asmt_table, exm_model, exm_table));
```

The code on the above is responsible for all the classes that are mentioned above, to be implemented when the user makes an input with the keyword or presses the button by using addActionListener to buttons and window for putting a keyword.

5. Saveinfo.java

```
public class SaveInfo {
    private static calendardiary cl = new calendardiary();
    private static Assignment as;
    private static ArrayList<Assignment> asResult = new ArrayList<>();
    private static Exam ex;
    private static ArrayList<Exam> exResult = new ArrayList<>();
    private static SortingFunction sf;
    private static List ls;

    public static JSONObject ob;
    public static int scheduleNum;
    private static String assignStr="assignment";
    private static String examStr="exam";
```

The Saveinfo file is responsible for storing and managing the information in the json file. The calendar, assignment, and exam array list are taken as member variables, and the information in the json file is parsed and added to the assginment and exam array list, respectively. A list merged assignment and exam lists created in this way is also created. Getter function makes to access private member variables. Through the showSched function, json object with the same date as argument of function is added to jslist to return jslist.

6. sortingFunction.java

```
public class SortingFunction {

    //dDay
    public static ArrayList<Assignment> assignmentDDaySorting(ArrayList<Assignment> assignments) {
        assignments.sort(Comparator.comparing(Assignment::getDDay));
        ArrayList<Assignment> result = new ArrayList<>();

        for(Assignment assignment : assignments) {
            result.add(assignment);
        }
        return result;
    }

//dDay reverse
public static ArrayList<Assignment> reverseAssignmentDDaySorting(ArrayList<Assignment> assignments) {
    assignments.sort(Comparator.comparing(Assignment::getDDay).reversed());
    ArrayList<Assignment> result = new ArrayList<>();

    for(Assignment assignment : assignments) {
        result.add(assignment);
    }
    return result;
}
```

The Sorting Function file is a file that implements various functions related to sort. First, assignmentDDaySorting, assignPrioritySorting, and assignmentsDifficultySorting are functions that return the result of ascending sort by D-Day, Priority, and Difficulty, respectively, and receive ArrayList as a argument to sort the order of and the list. The comparator is an interface that helps you sort objects, and as an implementation of the interface, you can override the existing sorting criteria and sort them with a new sorting criteria by passing them over to additional arguments in the sorting method, such as

Collections.sort(). Later, it was implemented by putting it into an ArrayList consisting of an Assignment called result through a for statement. ReverseAssignmentDDaySorting, reverseAssignmentPrioritySorting, and reverseAssignmentDifficultySorting are functions that return results sorted in descending order based on D-day, priority, and difficulty, respectively, and are implemented in a similar way to ascending order by giving additional reverse() methods.

```
public static ArrayList<Assignment> searchAssignment(ArrayList<Assignment> assignments, String subjectName) {
    ArrayList<Assignment> result = new ArrayList<Assignment>();
    for(Assignment assignment: assignments) {
        if(assignment.getSubjectName().equals(subjectName)) {
            result.add(assignment);
        }
    }
    return result;
}
```

In the case of searchAssignment and searchExam, when ArrayList and subjectName were received a arguments and the subject name was received as getSubjectName from Assignment or Exam in ArrayLi st, the result was implemented by comparing them one by one using the for and if statements.

7. Testing.java

```
class chooseActionHandler implements ActionListener
{
    private static ArrayList<Assignment> assignments = new ArrayList<Assignment>();
    //private static String type_name;
    private static TextField t1;
    private static TextField t2;
    private static TextField t3;
    private static TextField t4;
    private static TextField t5;
    private static TextField t6;

    private static String sub_name;
    private static int priority;
    private static int difficulty;
    private static String memo;

    private static String title;
    private static int end_date;

    private static int exam_date;
    private static String test_name;
```

The chooseActionHandler class has a member variable that allows the text field to match the values of the assignment and exam class. The actionPerformed function parses text and adds it to the exam or assignment list.

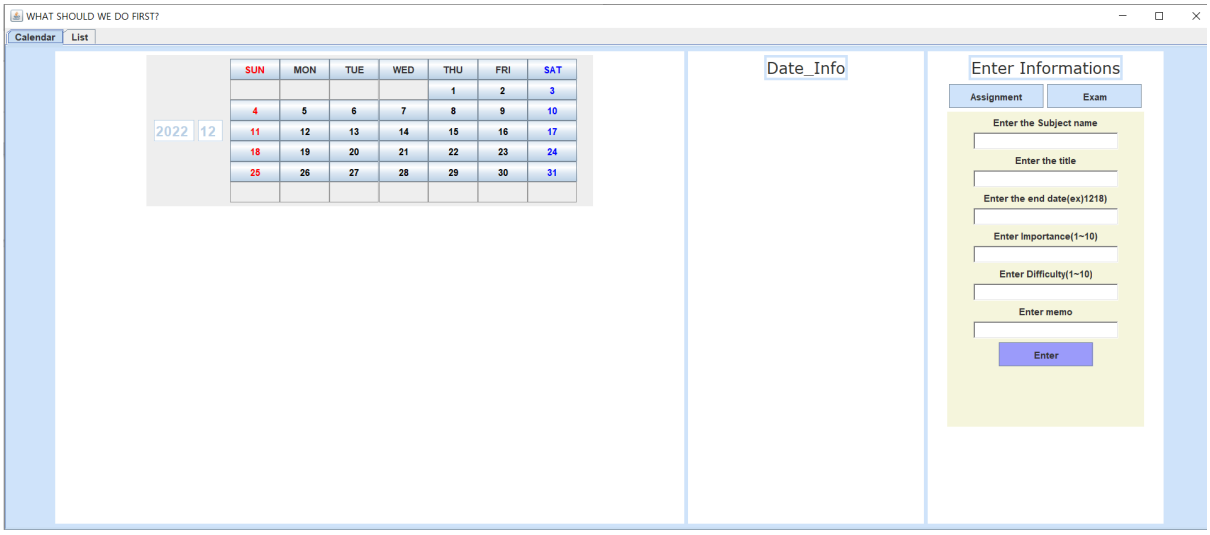
The typeHandler class is added to the assignment or exam panel according to the input of JButton. The test class has various member methods for outputting the desired screen. The createPInfoPanel function of the Testing class checks the type_name, adds the value of attribute to the panel according to each case, and returns it. The printInfoPanel function adds and returns visual elements such as the font, boundary, and background of the panel to the panel. The createAInfoPanel function returns this by adding text fields and buttons to display on the screen the window on which the user will receive input.

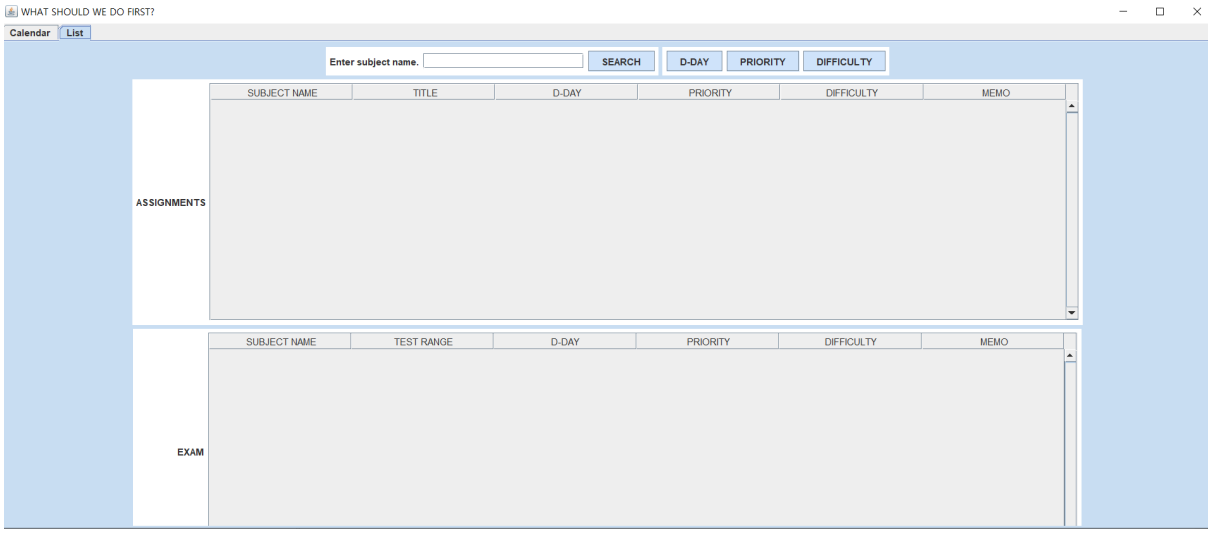
```java
class typeHandler implements ActionListener
{
    JLabel JL1;
    JLabel JL2;
    JLabel JL3;
    JLabel JL4;
    JLabel JL5;

    JButton b;
    JPanel thirdPanel;
    public typeHandler(JButton b,JPanel j)
    {
        this.b=b;
        this.thirdPanel=j;
    }
    public void actionPerformed(ActionEvent e) {
        if(b.getText().equals("Assignment"))
        {
            Testing.isexam=false;
            thirdPanel.add(Testing.createAInfoPanel());
        }
        else if(b.getText().equals("Exam"))
        {
            Testing.isexam=true;
            thirdPanel.add(Testing.createAInfoPanel());
        }
    }
}
```

The addInfoPanel function returns it by adding a title and button to display the menu above the window to be entered by the user on the panel.

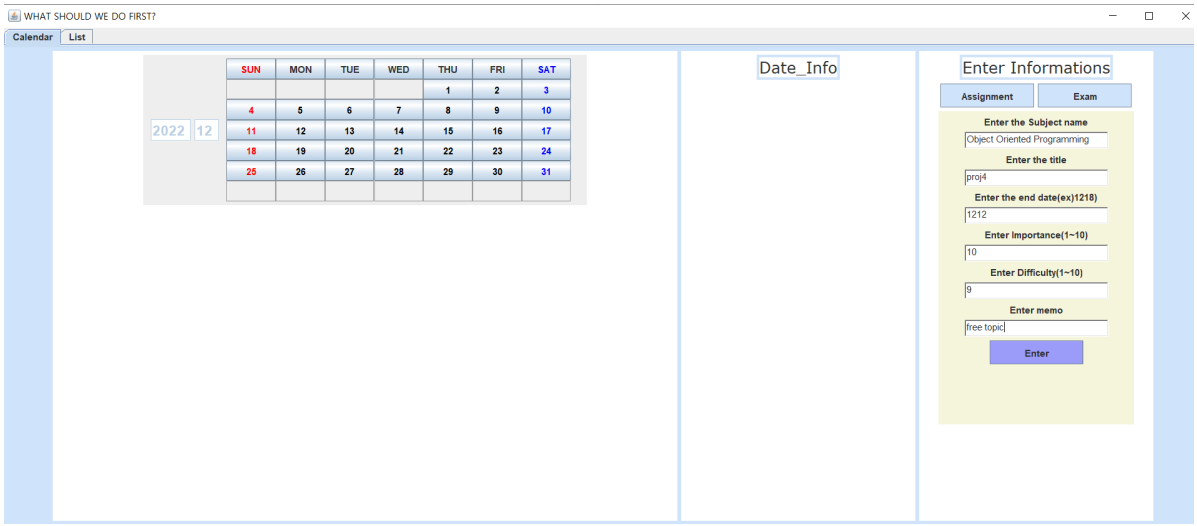
V. Execution results

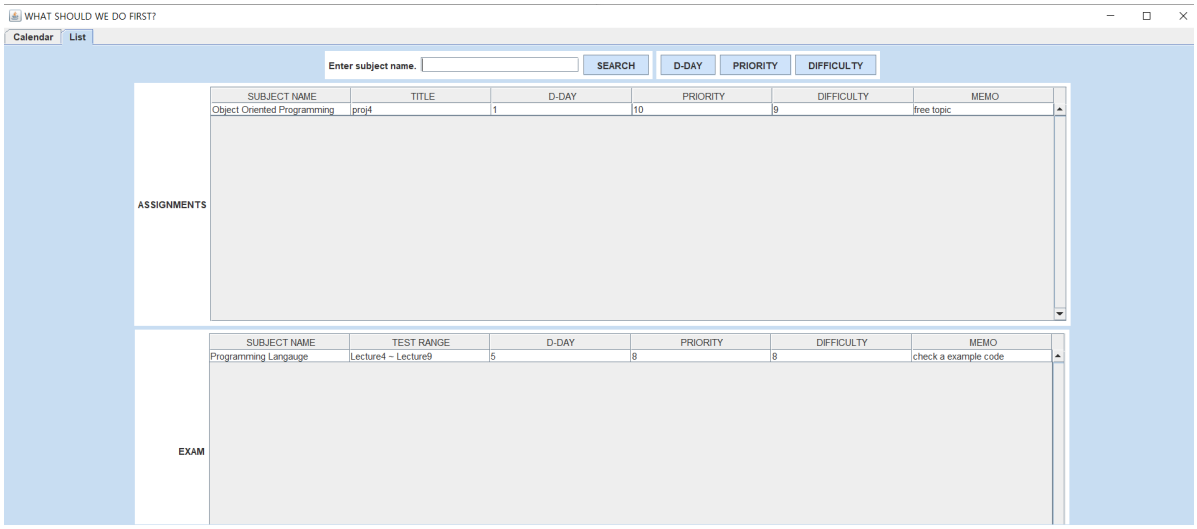
Let's take a brief look at the implementation of the code and the result of this program.



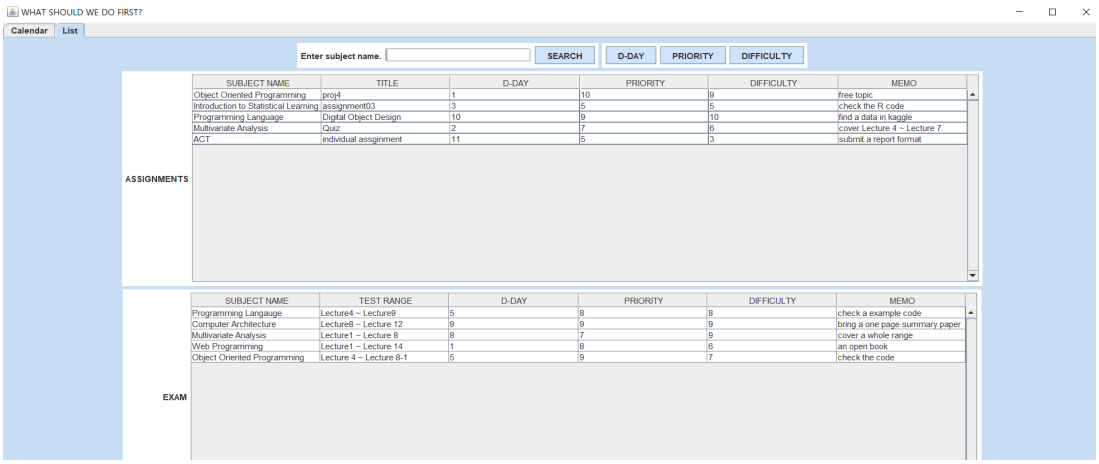


If you compile and execute the program like mentioned on the first page of the report, you could see the screen above. As the user did not make any input, there's no information on the screen both on the Calendar and List part. The user then makes an input like the below.

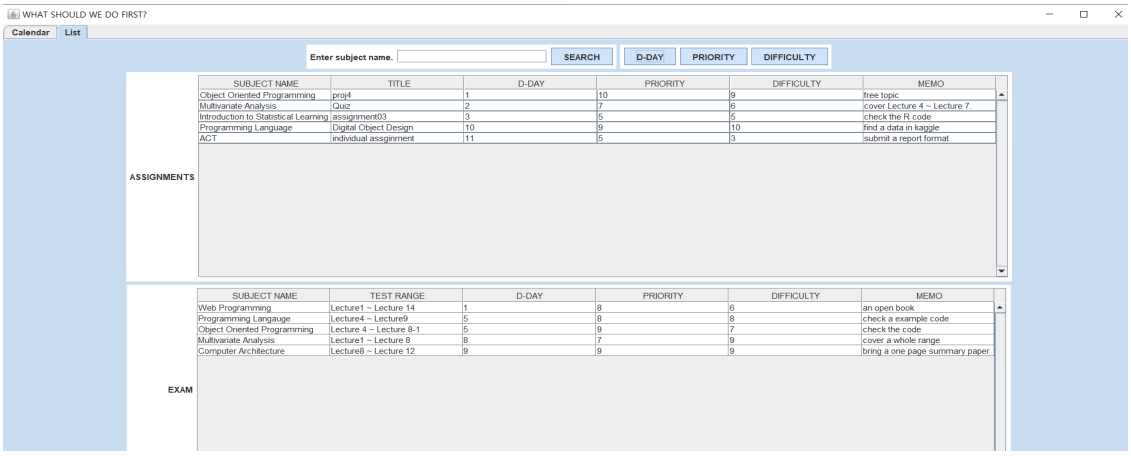




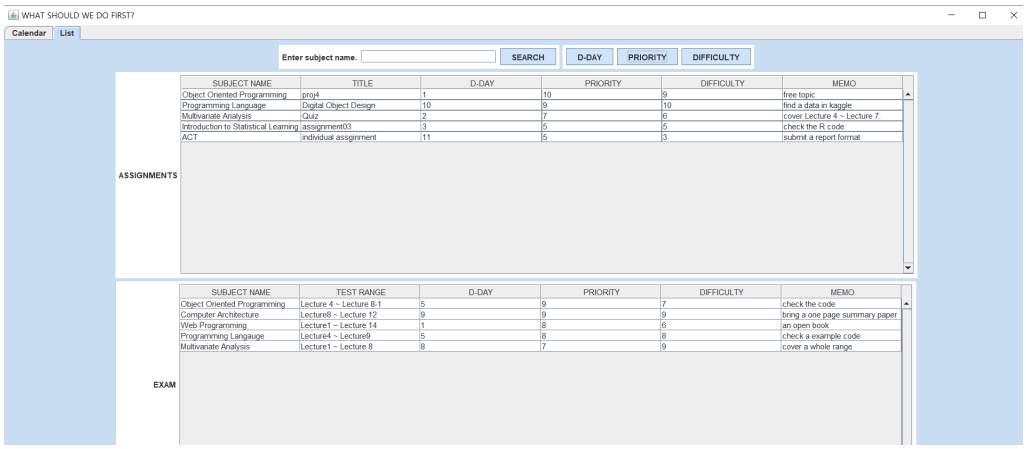
If the user makes an input, then you could see the corresponding information on the screen in the List part. To clarify the D-DAY, the date that implements this program is December 11th(12.11).



Now, a user made several inputs and the result is shown above. The result is shown in the order of making an input. If you click the button D-DAY, the data is shown like the below.

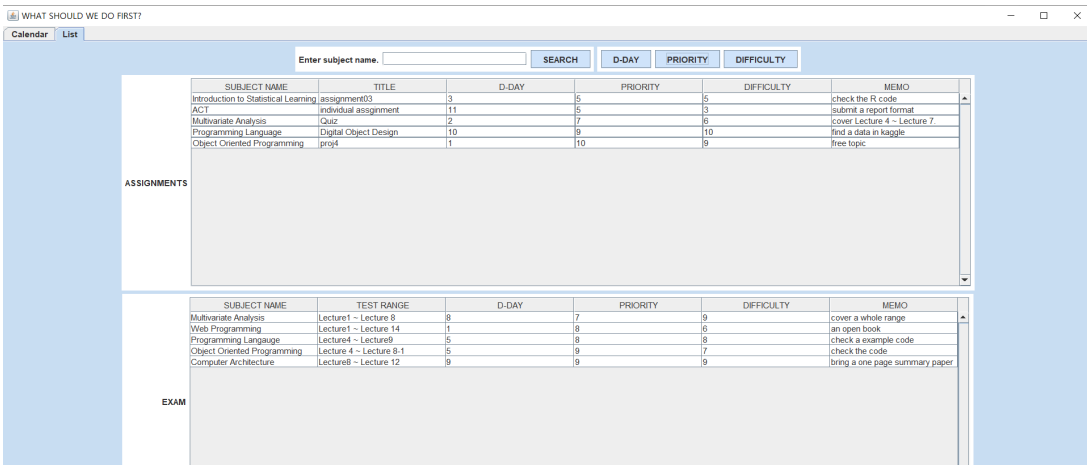


It shows the data by D-DAY in ascending order , which shows the task which has less days left. But, if the user clicks the D-DAY button one more time, the window displays the data in reverse order, which is descending order like the below. If the user clicks it again, then it shows the data again in ascending order. It just repeats the process.

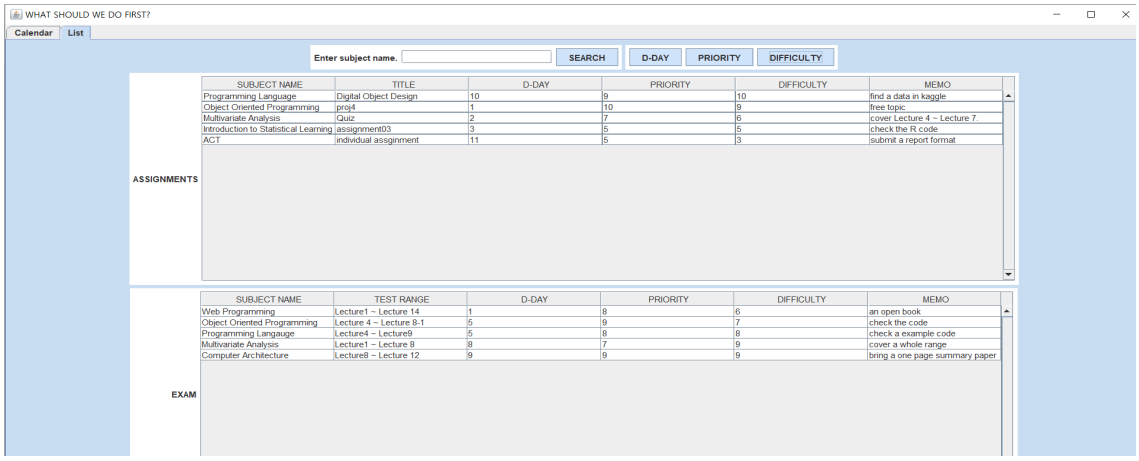


The above picture shows the result of sorting the data by PRIORITY in descending order, when the user just clicks once, which is different from the result of sorting by D-DAY which goes for ascending

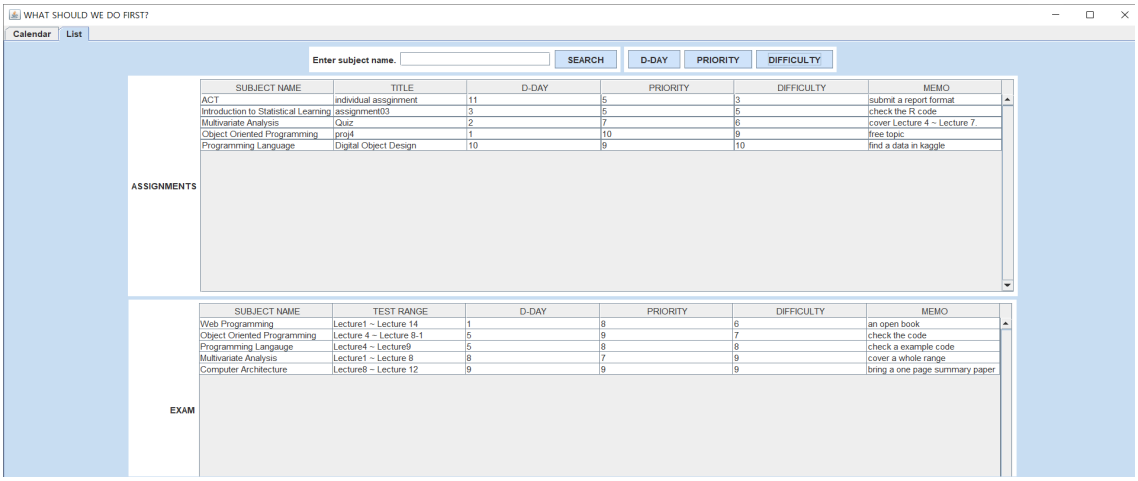
order. If the user clicks the button again, it shows the information in ascending order like the below.



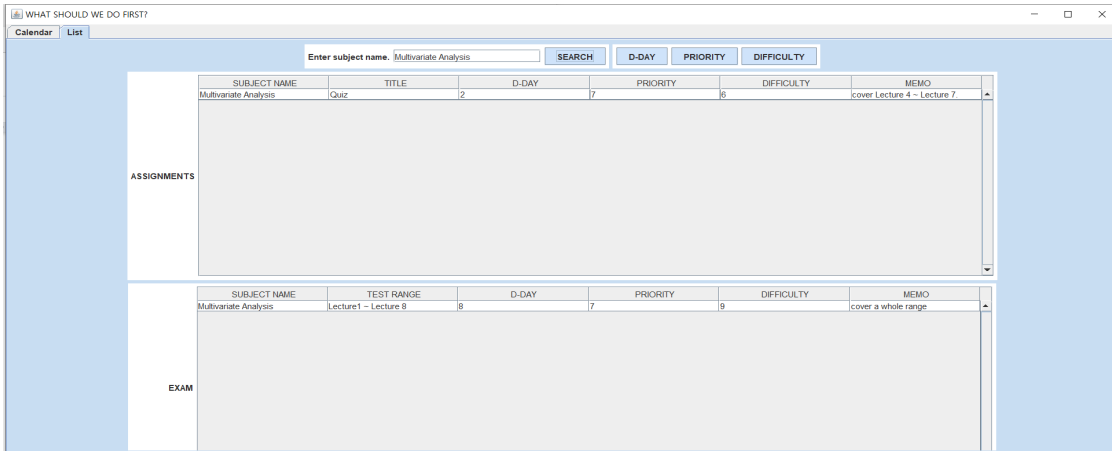
The picture below shows the result of sorting the data by DIFFICULTY in descending order, when the user just clicks once, just like PRIORITY.



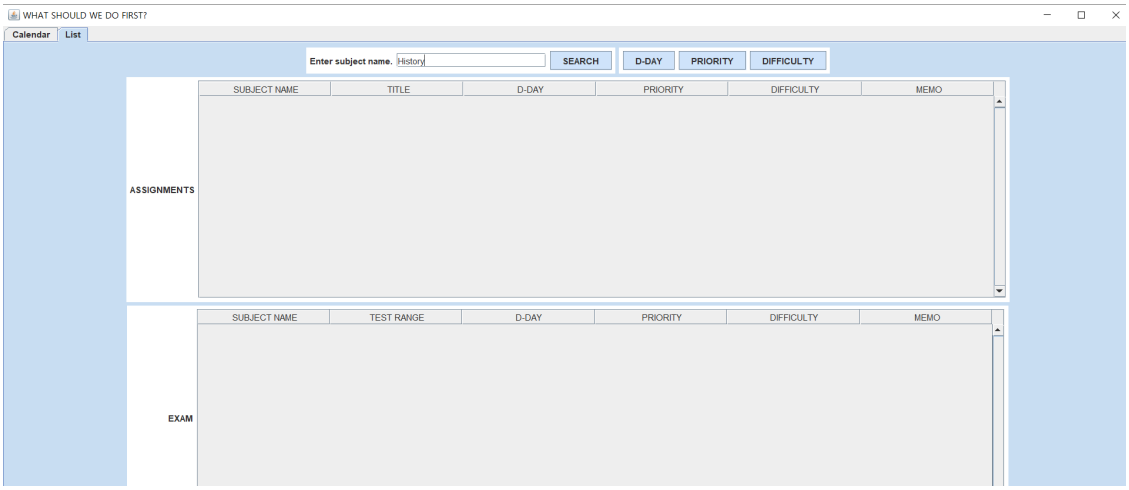
If the user clicks the button again, it displays the information in ascending order like the below.



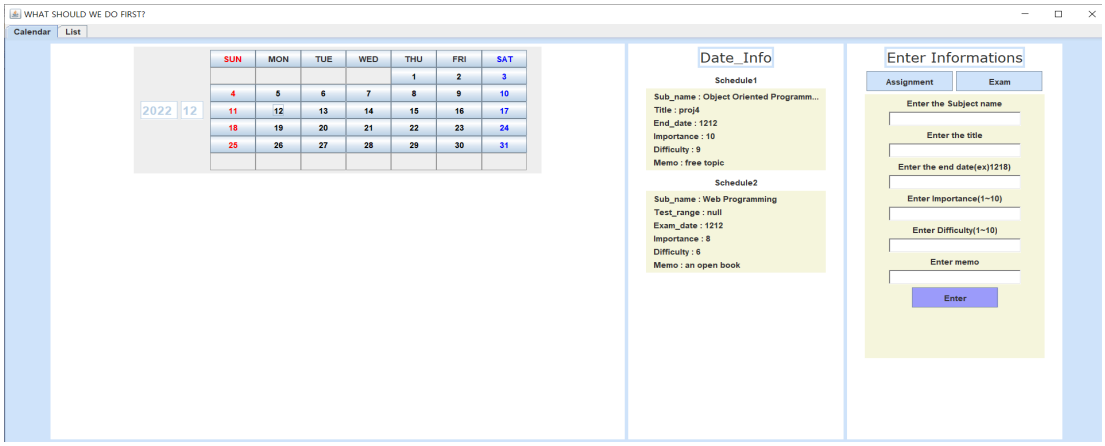
For both PRIORITY and DIFFICULTY buttons, if the user clicks the button again, it repeats the process like the button D-DAY.



The above one shows the result of doing SEARCH. If the user makes an input of a keyword, which is the name of the subject, and then clicks the button SEARCH, the window displays the corresponding information. The keyword that the user made exists in the data stored, so it prints out. However, if there's no data matching, then it prints nothing like the below.



If the user clicks a certain data on the Calendar, then it prints out the information of the task on that certain data like the below.



VI. Review on the project

1. Application of Object Oriented Programming

The entire software system was made by applying the concepts of object oriented programming. It was implemented as Java, a representative object-oriented language. The concept of encapsulation and inheritance, which are the core concepts of object orientation, was used in the project. First, the attribute related to the exam information in the 'Exam' class was set to a private member variable, and the value of each variable could be obtained through the getter function. In addition, a Constructor was created so that an instance could be created. 'Assignment' class is also similar. In addition, various classes created to implement functions are inherited using 'extends' and 'implements'. In the case of 'extends', both declarations and definitions are made by the parent, and the child can use the parent's methods and variables as they are without the need to override them. In the case of 'implements', the parent object is only declared, and the definition must be overridden by the child. Also, member variables or methods within a few classes were implemented to be accessible directly through names without creating objects using static as needed.

2. Opinion on the Project

Since this project was a free topic, many opinions were exchanged from the selection of the topic. It was not easy to collect opinions from the selection of the topic, but we thought about what projects we needed. In order to create a better project, there was a little difficulty in implementing the function because Gui had to be used. However, after applying several cases, we were able to gradually improve the project in a better direction.

VII. Conclusion

The project focused on implementing the task to consider priority and importance of assignments and exam schedules, output by sorting them in various ways, and check the schedule according to the date through the calendar. This project is meaningful in what we did in consultation with team members, from the topic to structural design and implementation. We think it would be a good opportunity for us to experience the whole project. Efforts were made to utilize various classes to apply the concept of object-oriented programming. In addition, GUI was used to make it possible for users to intuitively perform certain functions. We hope that this project can help us manage a better schedule.