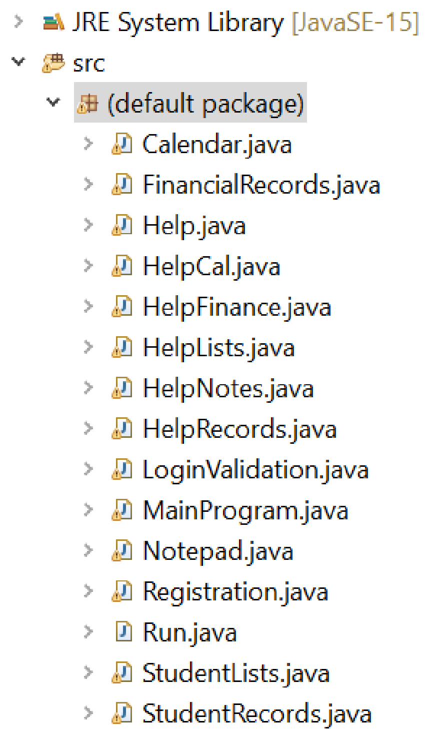
ICS4U7-3B Internal Assessment Criteria C

Introduction:

This program was written in Java. It is an interactive organizer for private music teachers that can be used and displayed by the client to manage information about schedules, students, finances, and notes. Inputted data is saved in a directory named after the username created.



```
> JRE System Library [JavaSE-15]
v src
  v (default package)
    > Calendar.java
    > FinancialRecords.java
    > Help.java
    > HelpCal.java
    > HelpFinance.java
    > HelpLists.java
    > HelpNotes.java
    > HelpRecords.java
    > LoginValidation.java
    > MainProgram.java
    > Notepad.java
    > Registration.java
    > Run.java
    > StudentLists.java
    > StudentRecords.java
```

Figure 1: A general list of the program’s functions.

Registration & Login Validation:

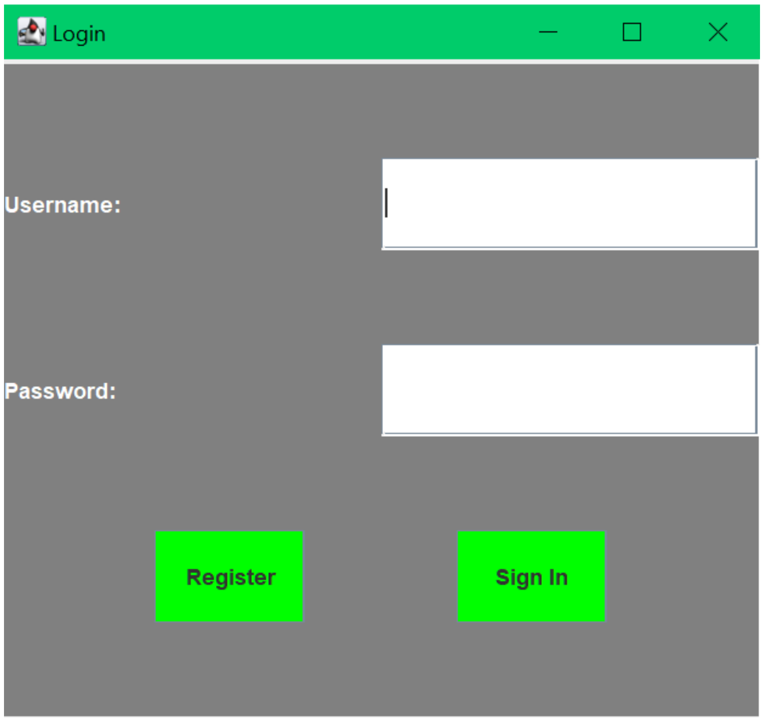


Figure 2: Login screen.

For the implementation of this screen, I used JPanels, JFrames, JLabels, JTextFields, and JButtons.

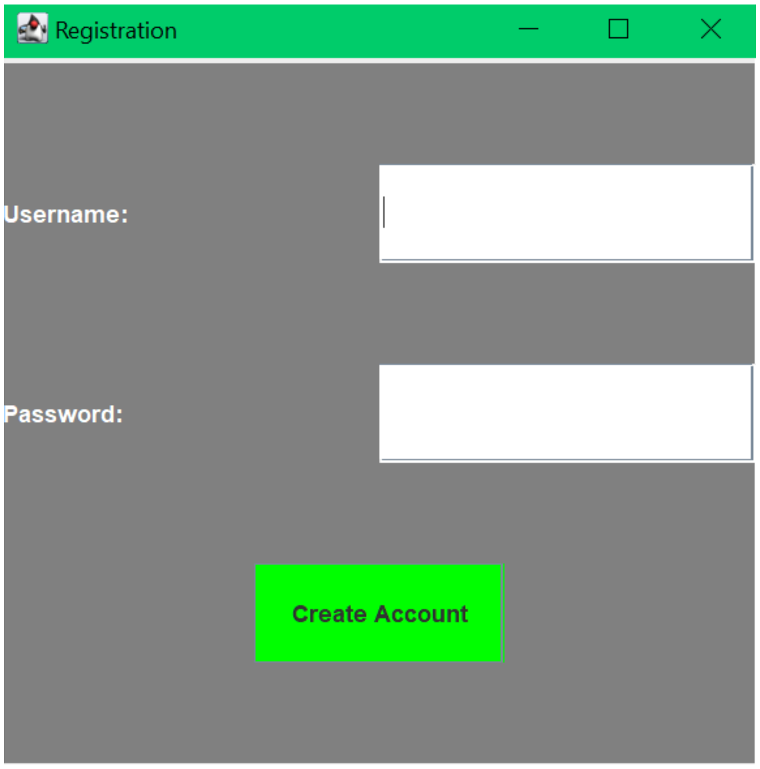
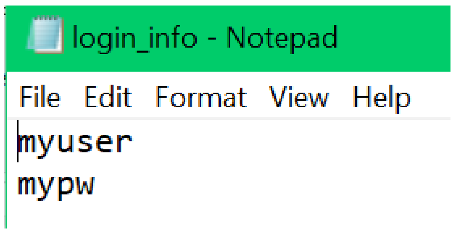


Figure 3: Registration screen.

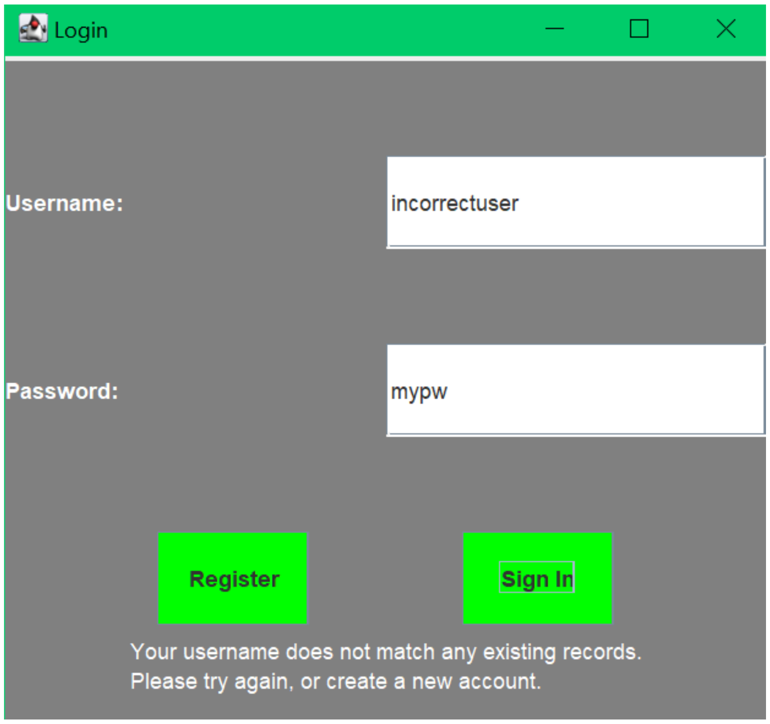
First-time users are required to register before signing in. After pressing Create Account, using FileReader and FileWriter, all of the information from the JTextFields are saved in a text file, named login_info.txt. The username is written on the first line, followed by the password on the second.



```
myuser
mypw
```

Figure 4: The saved username and password.

When the user attempts to sign in, the inputted data in the JTextFields are taken and compared to the saved text file. Thus, error messages are displayed through a JLabel when appropriate.



Your username does not match any existing records.
Please try again, or create a new account.

Figure 5: Error message when the user inputs a username not associated with any account.

Home screen:

Upon successfully entering the program, the user is presented with 7 buttons, labelled with different functions. Pressing one creates a new JFrame displaying the appropriate screen.

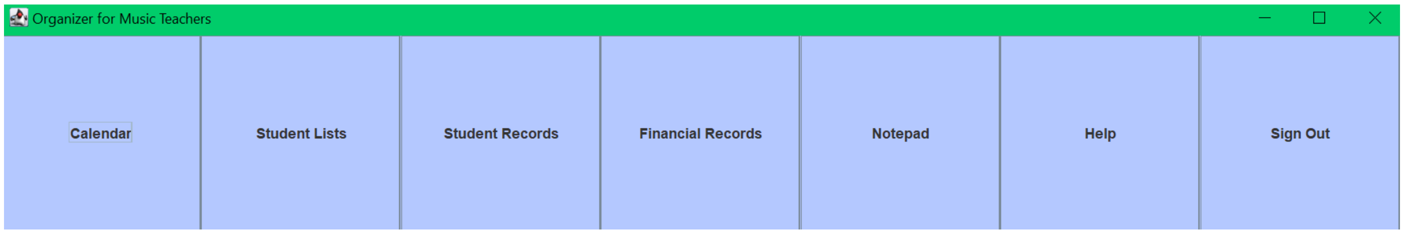
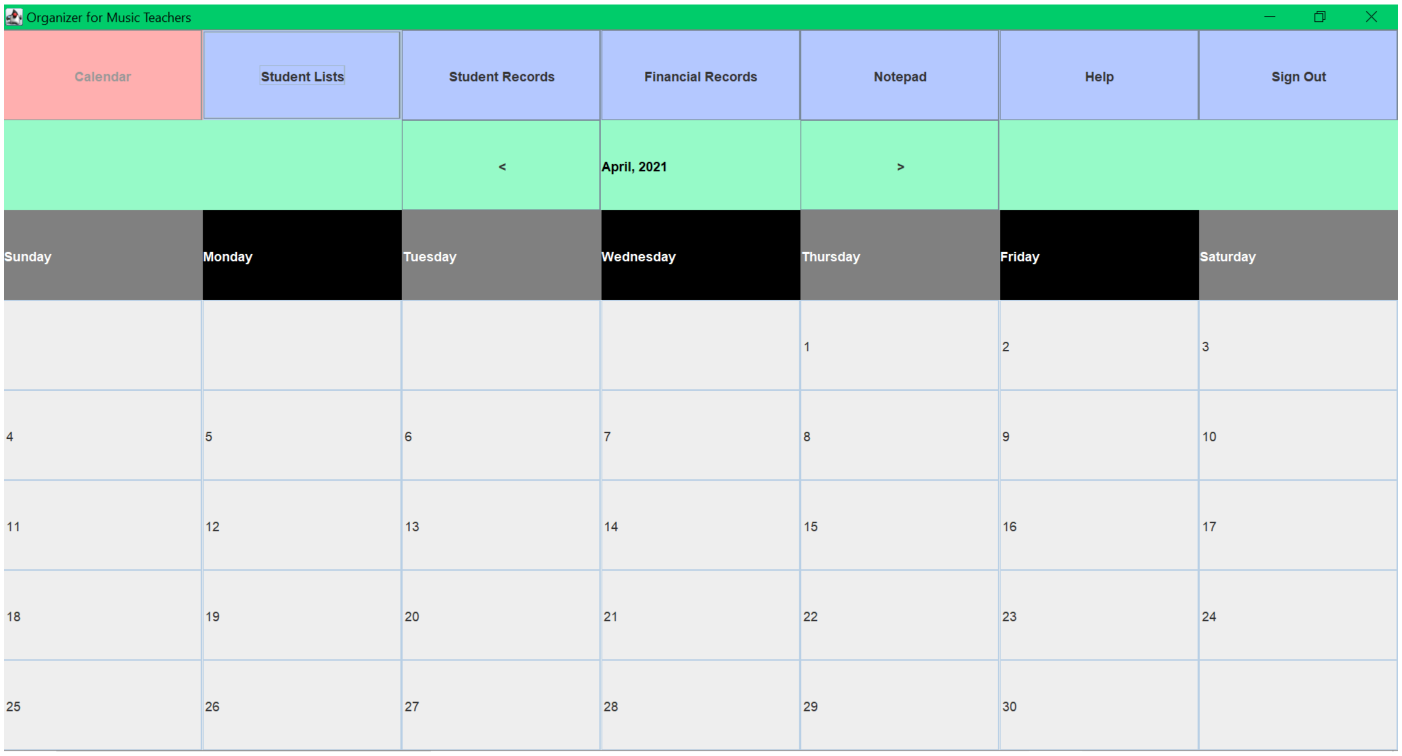


Figure 6: Home screen.

Calendar:

At the top of every screen, the same main buttons are carried over into the subclass from the superclass. In this case, the Calendar button is greyed out, as well as highlighted in a different colour for clarity. For accurate implementation of the calendar screen, I imported the java.time package.



Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Figure 7: Calendar.

When the arrows are pressed to scroll through time, a month is added or subtracted from the current (starting) date, and a method that updates the display is called.

Through the implementation of MouseListener, users can click on a specific date and trigger a new frame to pop up, prompting them to either display or add the day's events.

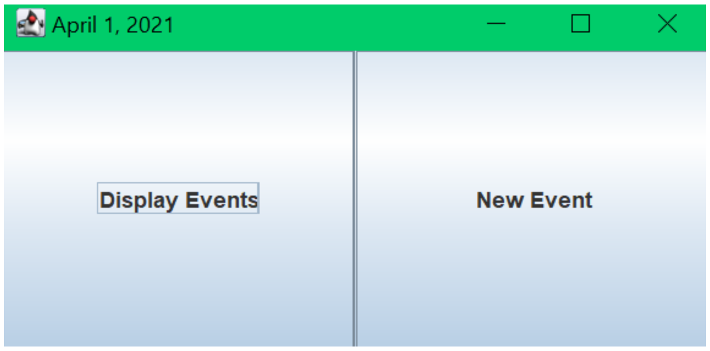


Figure 8: Day’s prompt.

When a new event is added, the input is saved on separate lines to a text file named after the date in a new directory, named Calendar. Afterwards, the event can be displayed by retrieving the file’s contents.

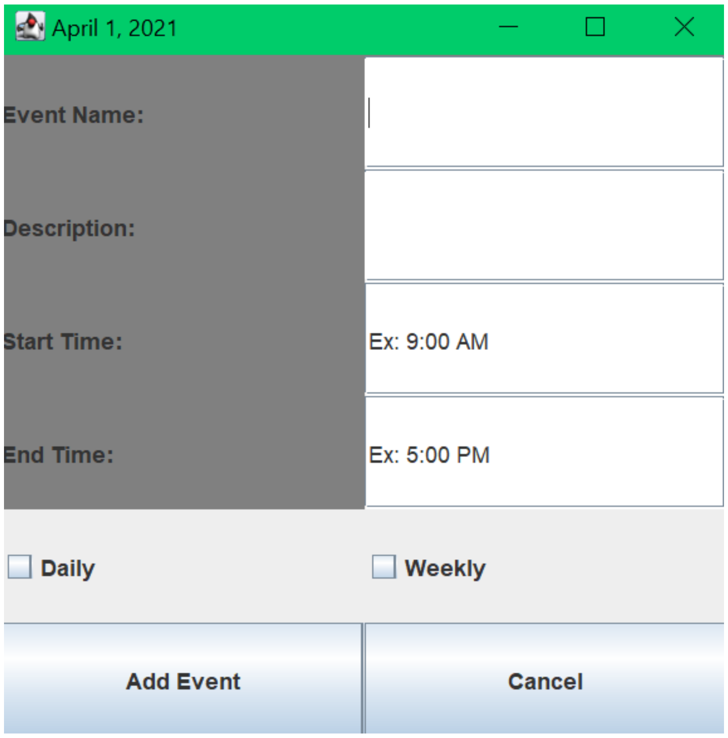
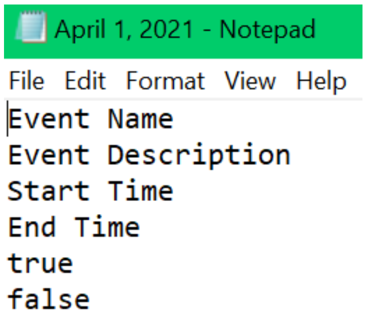
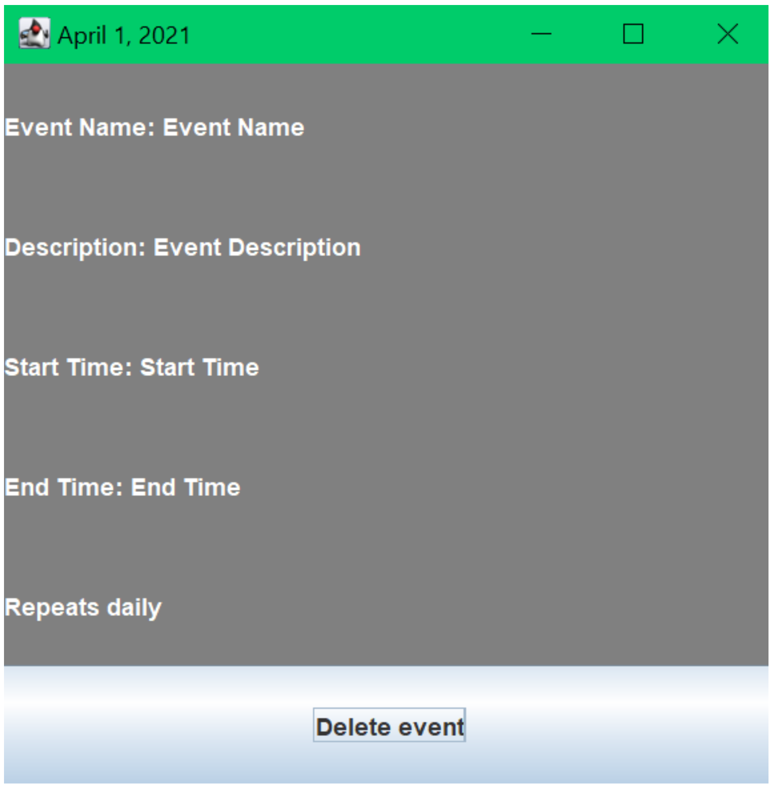


Figure 9: Prompt for adding a new event.



```
Event Name
Event Description
Start Time
End Time
true
false
```

Figure 10: Saved event information.



Event Name: Event Name

Description: Event Description

Start Time: Start Time

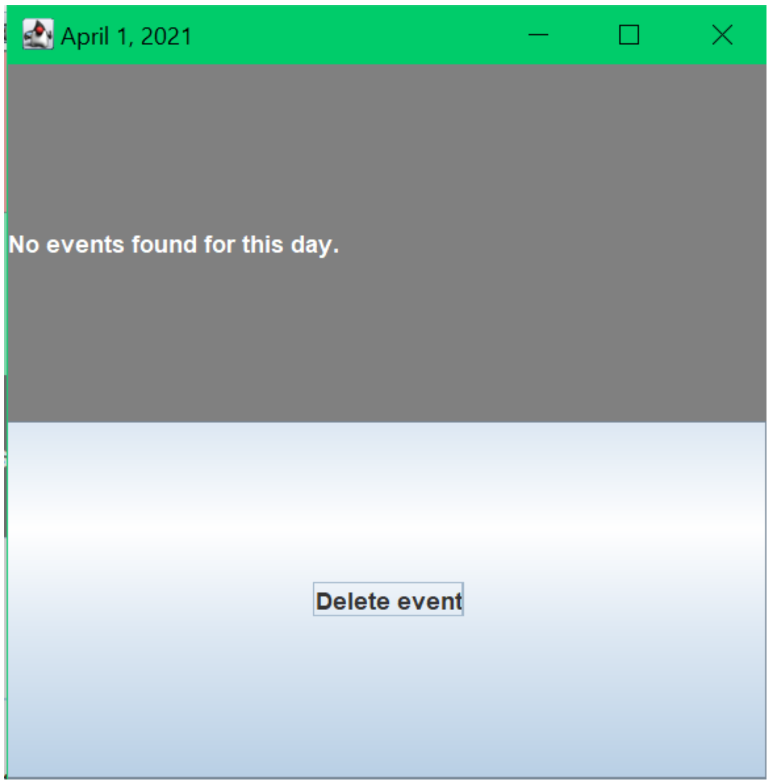
End Time: End Time

Repeats daily

Figure 11: Event display.

If the user selects the JCheckBox for repeating events, new text files are saved with the same event information for up to a month.

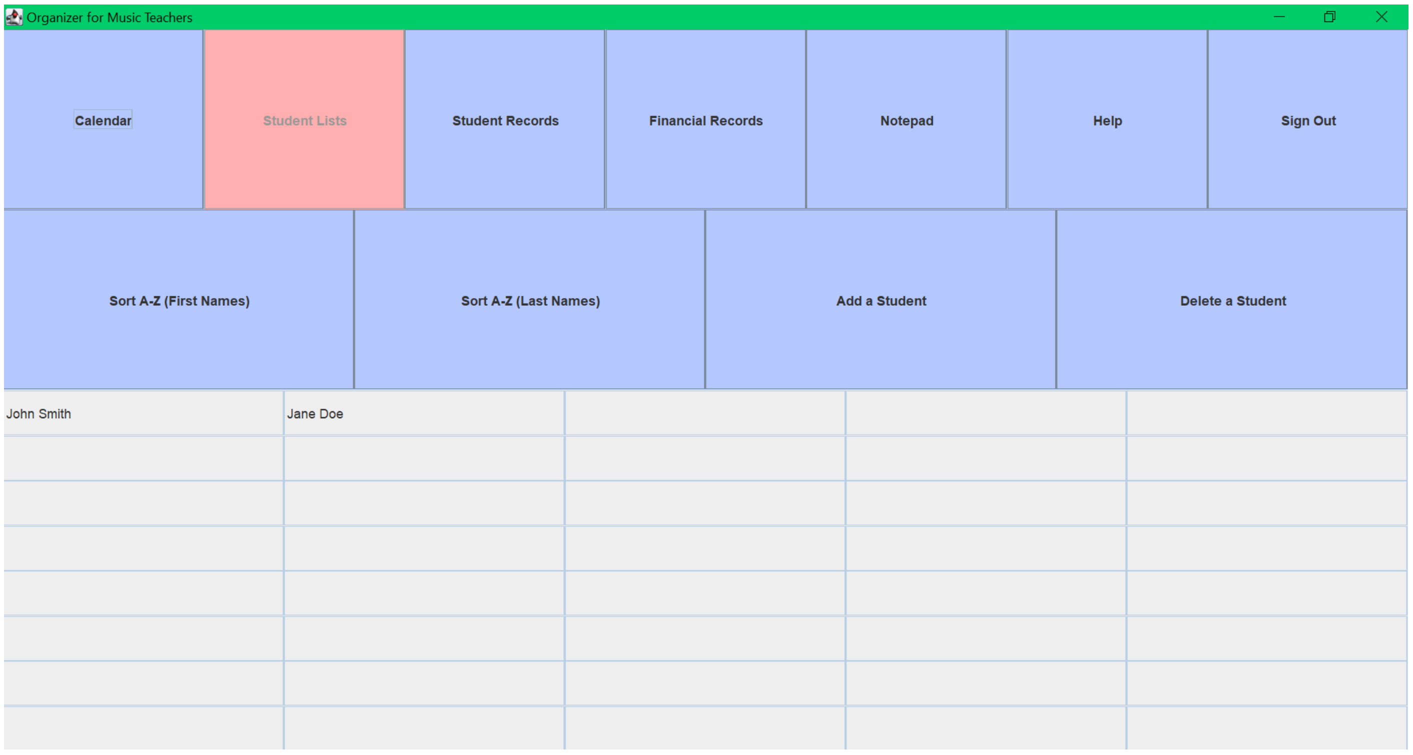
When the user deletes the event, the text file is retrieved, and all its contents are overwritten with an empty String. Furthermore, when dates are associated with no existing text files or with an empty one, a message informs the user.



No events found for this day.

Figure 12: Message for when no filled text file is found for the specified date.

Student Lists:



John Smith	Jane Doe			

Figure 13: Student Lists.

For this screen, users can add a student, executing a similar process to adding a Calendar event: a new text file named after the student's name would be created in a new directory, called StudentLists. However, a text file, named Students_MasterList.txt would be written upon the creation of the first student, and every addition would append the list.

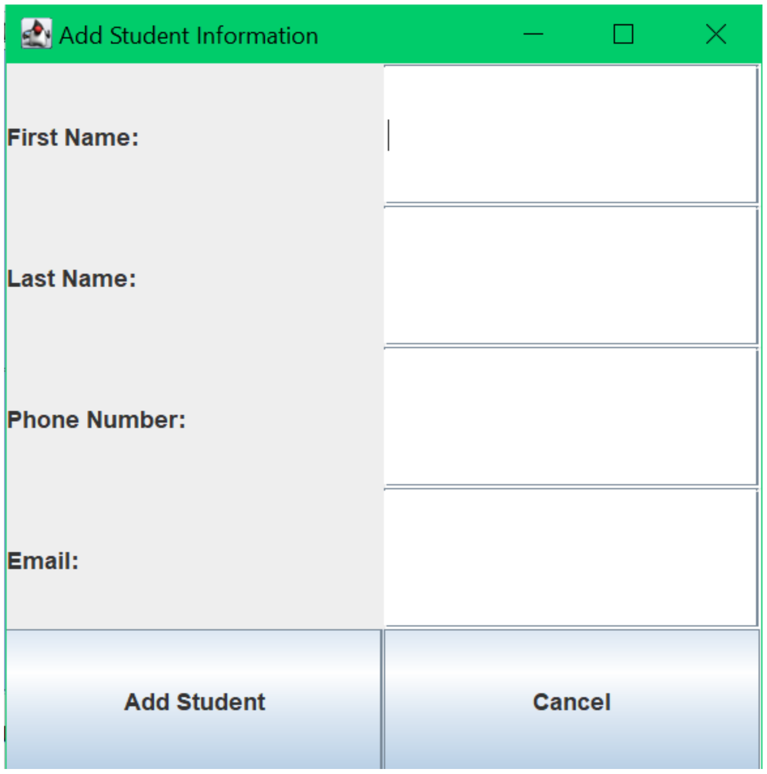


Figure 14: Prompt for adding a new student.

To delete a student, users are asked to select the name, using MouseListener. Then, the student's respective text file has its contents erased. The master list is also updated by searching for their name, then overwriting it with an empty String. Lastly, the main screen's display is updated by refreshing the frame and re-reading the contents of the master list.

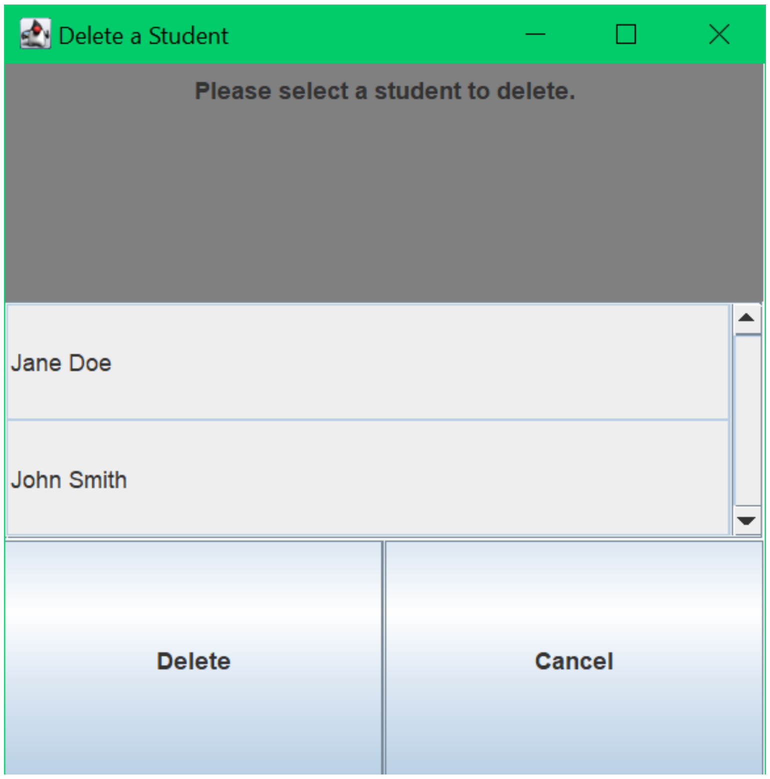
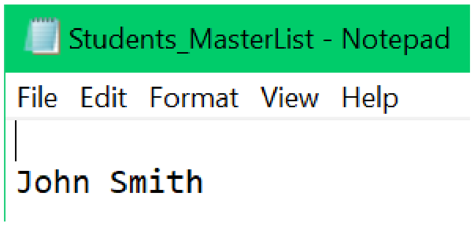


Figure 15: Deletion prompt.



```

John Smith
```

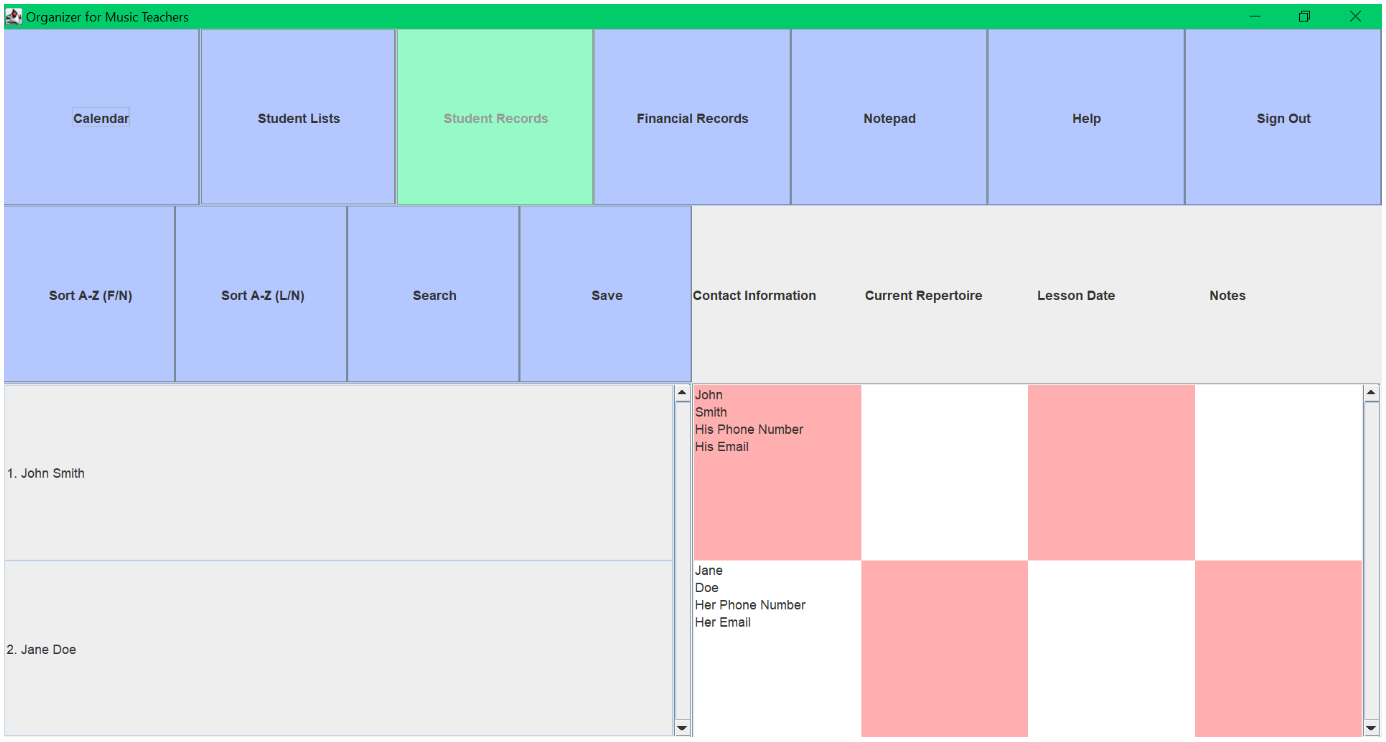
Figure 16: Master list file with Jane Doe's name replaced by an empty String.

When the user presses the sorting buttons, the names in the master list are retrieved and sent to an array, which is then sorted in alphabetical order through Selection sort. The main display is refreshed again to show the correct order of the names.

Student Records:

Using JTextFields and FileReader, this screen retrieves students from the master list and displays editable fields under four headings for the user to input information. After pressing Save, the entered input is taken and saved into text files in a new directory, StudentRecords.

The contact information fields retrieve the previously saved information from Student Lists and display it. Similarly, edits to that column are saved in Student Lists.



1. John Smith

2. Jane Doe

John
Smith
His Phone Number
His Email

Jane
Doe
Her Phone Number
Her Email

Figure 17: Student Records.

Using previously mentioned processes, the sorting buttons order the names alphabetically, and text fields' input is also taken and rearranged in a separate array.

Searching for a student prompts the user to enter a name. Since the maximum number of students is 40, a relatively small number, I used Linear search to return the row position of the first occurrence or an error message if the student is not found.

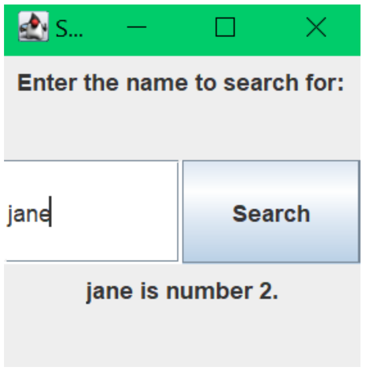


Figure 18: Prompt and result for “jane” search.

Financial Records:

Like the Student Records, this screen displays names and editable fields under headings. However, the input is, instead, saved in a new directory called FinancialRecords. The searching and sorting functions process the same way too.

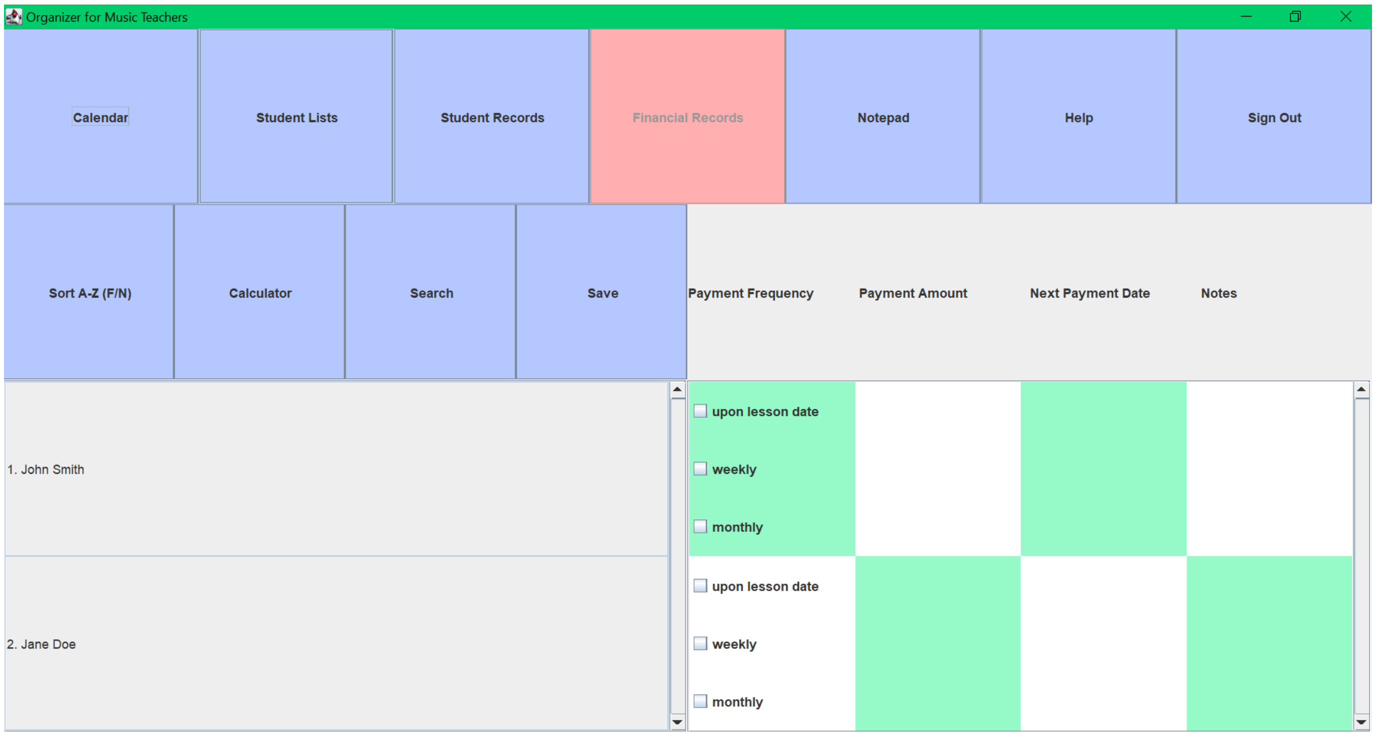


Figure 19: Financial Records.

The calculator function pops up as a new frame that allows the user to press buttons that refresh the display label, showing the text associated with the button.

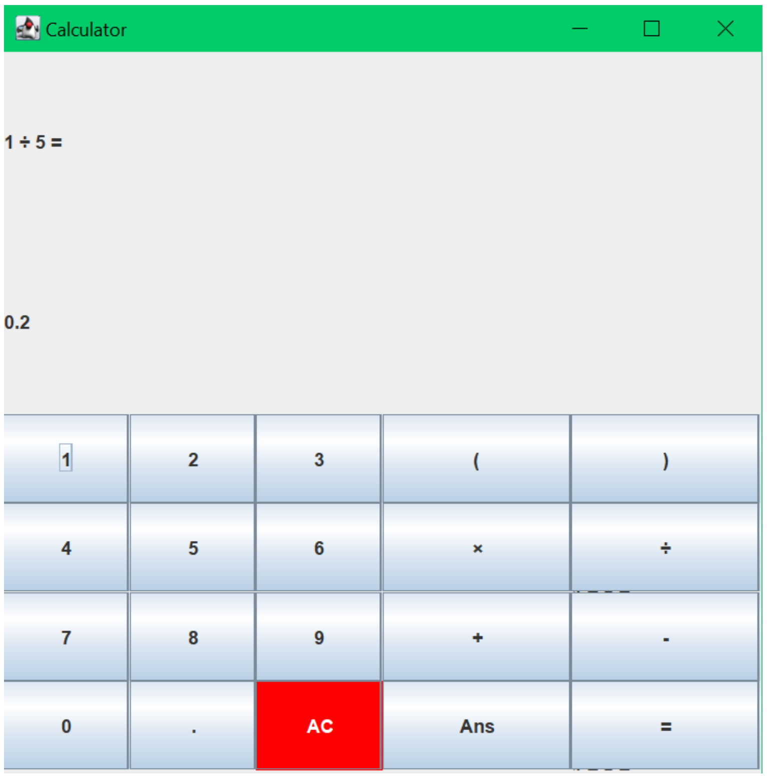
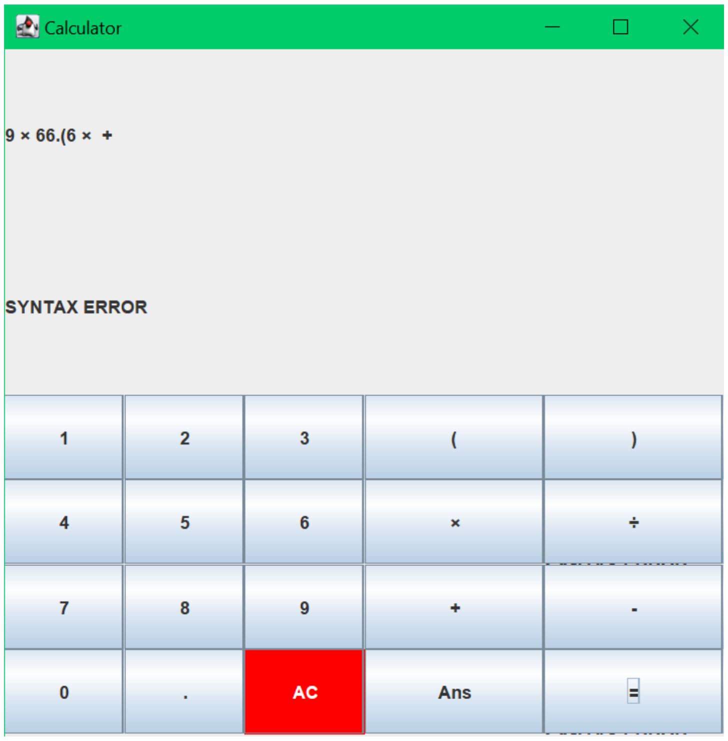


Figure 20: Calculator.

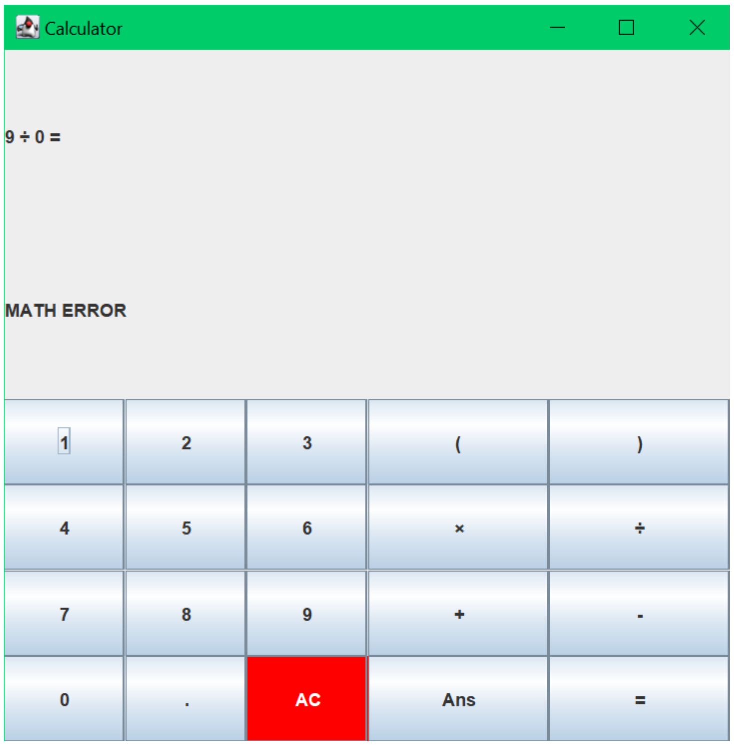
Pressing the equals button calls an evaluation method that uses parsing and conditional while loops to recognize the next symbol inputted. The method uses recursion to call itself when parentheses are detected, thereby evaluating the insides first before proceeding with the rest of the equation. I also implemented BEDMAS by separating the processes for different operands. Lastly, error messages are outputted when errors occur.



```
9 × 66.(6 ×  +

SYNTAX ERROR
```

Figure 21: Syntax error message.



```
9 ÷ 0 =

MATH ERROR
```

Figure 22: Math error message.

Notepad:

The notepad function is a screen that uses GridLayout to display a large, editable text area. User input is saved to notepad.txt in a new directory, named Notepad upon signing out.

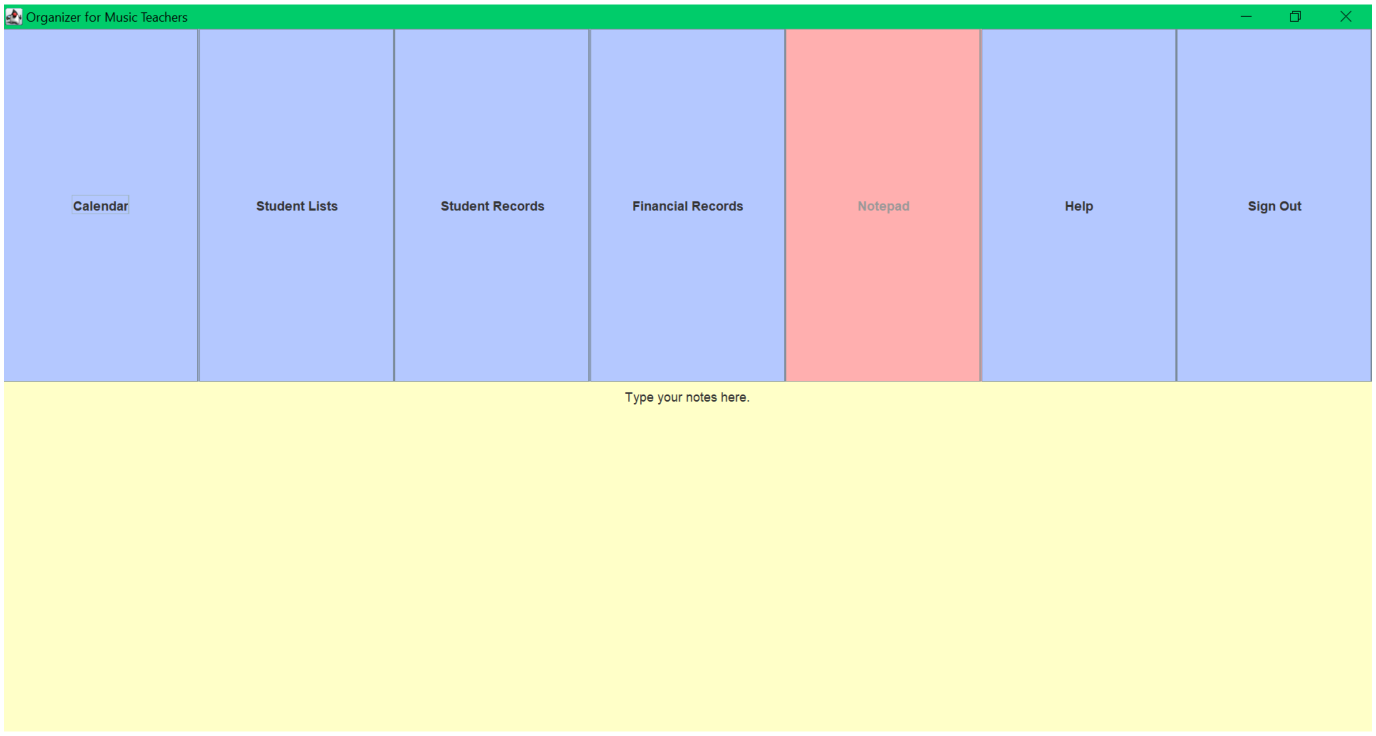
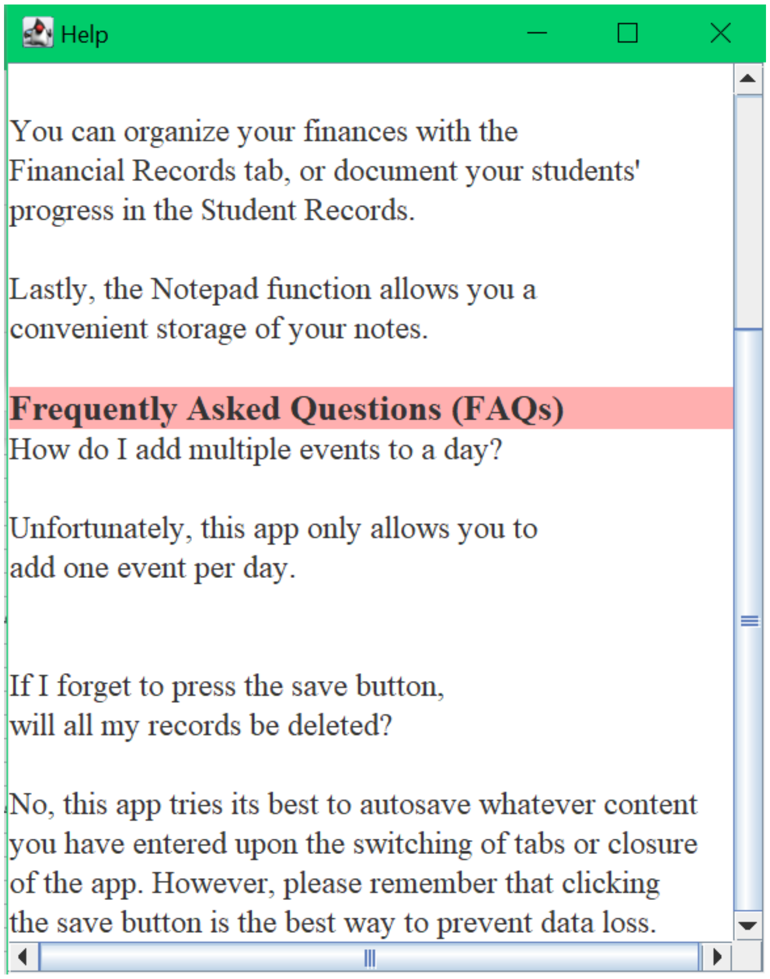


Figure 23: Notepad.

Help:

On the home screen, the help button appears as a frame with a JScrollPane containing a panel. It gives general instructions on how to use the program and answers to some "FAQs." Pressing the help button on other screens gives tab-specific information.


Help

You can organize your finances with the
Financial Records tab, or document your students'
progress in the Student Records.

Lastly, the Notepad function allows you a
convenient storage of your notes.

Frequently Asked Questions (FAQs)

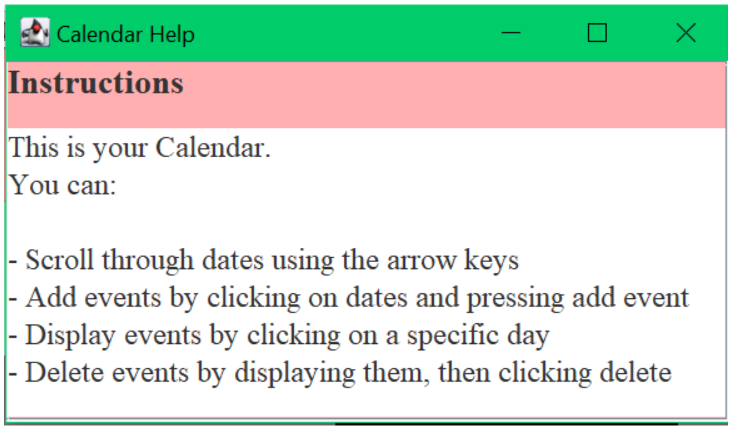
How do I add multiple events to a day?

Unfortunately, this app only allows you to
add one event per day.

If I forget to press the save button,
will all my records be deleted?

No, this app tries its best to autosave whatever content
you have entered upon the switching of tabs or closure
of the app. However, please remember that clicking
the save button is the best way to prevent data loss.

Figure 24: Home screen Help.

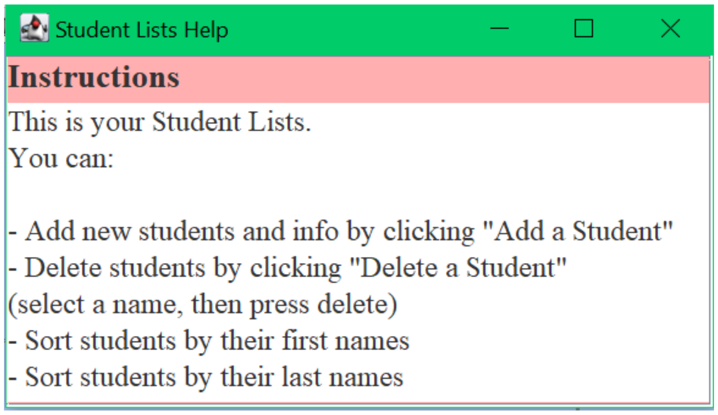

Calendar Help

Instructions

This is your Calendar.
You can:

- Scroll through dates using the arrow keys
- Add events by clicking on dates and pressing add event
- Display events by clicking on a specific day
- Delete events by displaying them, then clicking delete

Figure 25: Calendar Help.



Instructions

This is your Student Lists.
You can:

- Add new students and info by clicking "Add a Student"
- Delete students by clicking "Delete a Student"
(select a name, then press delete)
- Sort students by their first names
- Sort students by their last names

Figure 26: Student Lists Help.

Word Count: 1053

Bibliography:

(1) Java Platform SE 7. (n.d.). Retrieved April 03, 2021, from https://docs.oracle.com/javase/7/docs/api/.

(2) Java tutorial: Learn Java - JAVATPOINT. (n.d.). Retrieved April 03, 2021, from https://www.javatpoint.com/java-tutorial.