# Date Tracking Application

Richie Zhang

# Table of Contents

**NOTE: RUN MainGUI.java to start the application**

# System Analysis

# Project Description

## General Description

The date tracking application is designed to help its users manage and track important dates efficiently. Its primary function is to display important dates to users in ascending order based on their proximity, offering a clear and intuitive display of upcoming events. The application caters to both personal and professional needs, allowing users to categorize dates into various types such as work, holidays, anniversary, or any custom type the users create. Users can also adjust the display using the “sticky to top” option and selecting to only display a certain type of date to better organize their dates

## Goals and Benefits

- To provide a user-friendly platform for tracking important dates
- Enable users to flexibly categorize and prioritize dates based on type and urgency
- Offer a date calculator feature for computing days between two dates

## System Inputs and Outputs

**Inputs**

- Dates and date information (name, stickied status, customized date types)
- Two dates from the date calculator feature
- Type of date to display

**Outputs**

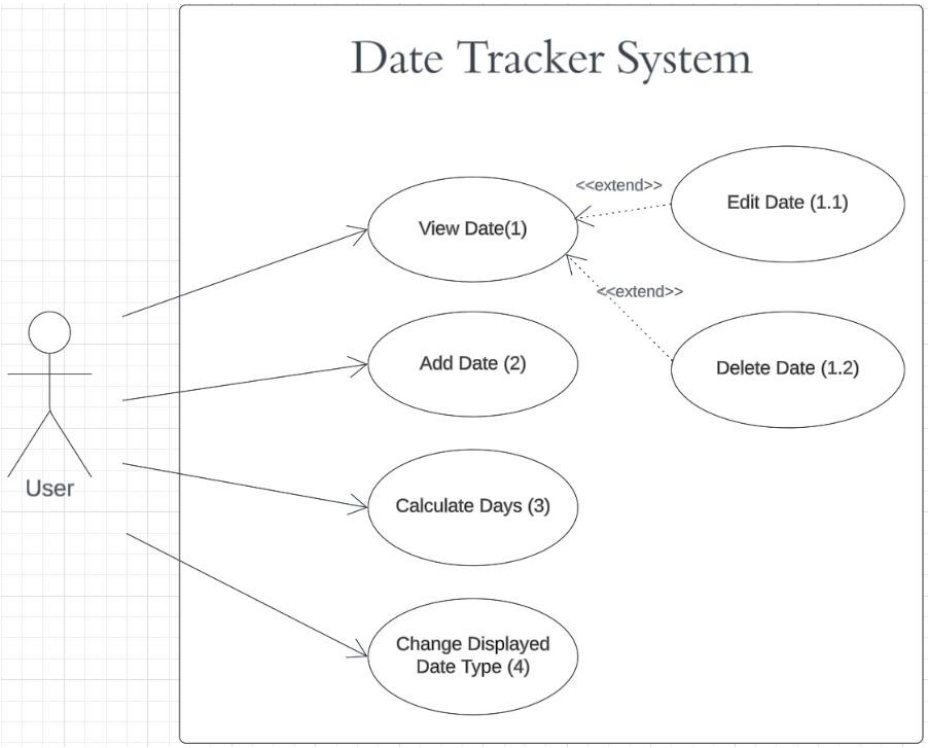
- Display of upcoming dates in an organized manner
- Calculation results from the date calculator feature

## Design Details

- The application's backend will be coded in JAVA
- GUI development will be based on Java swing
- Data is stored in .dat files and is read every time the application is opened

# Use case Diagram and Use Case Descriptions

## Use Case Diagram




## Descriptions

| UC number: 1 | |
|---|---|
| Overview | Allows the user to view a selected date |
| Related Use Case | 1.1 |
| Actors | User |

| UC number: 1.1 | |
|---|---|
| Overview | Allows the user to edit a selected date's previous information |
| Related Use Case | 1 |
| Actors | User |

| UC number: 1.2 | |
|---|---|
| Overview | Allows the user to delete a selected date |
| Related Use Case | 1 |
| Actors | User |

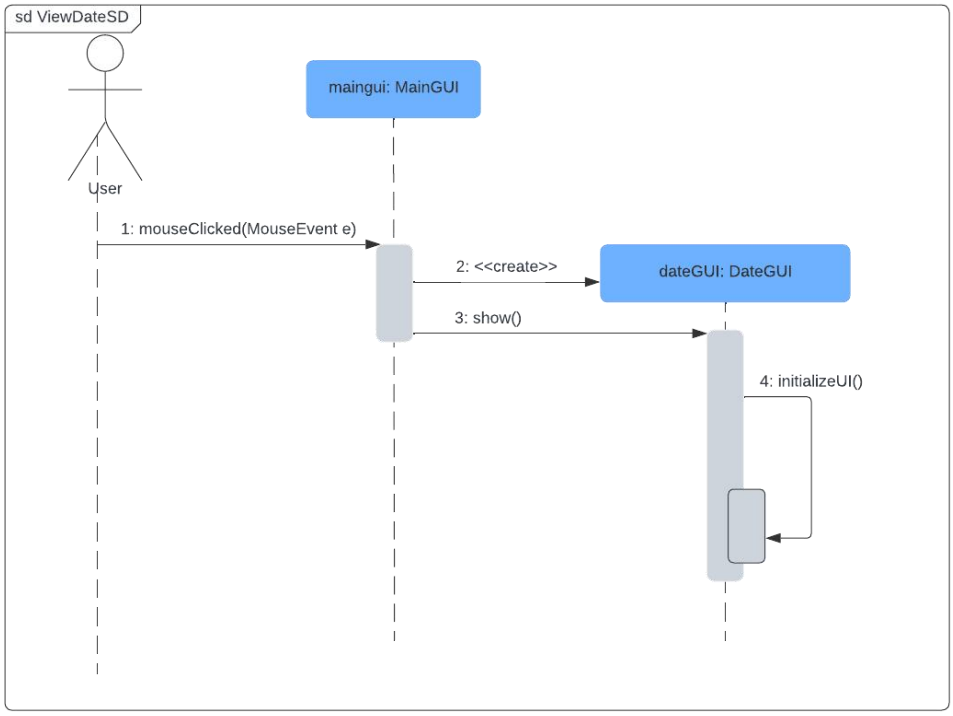
| UC number: 2 | |
|---|---|
| Overview | Allows the user to add a new date |
| Related Use Case | None |
| Actors | User |

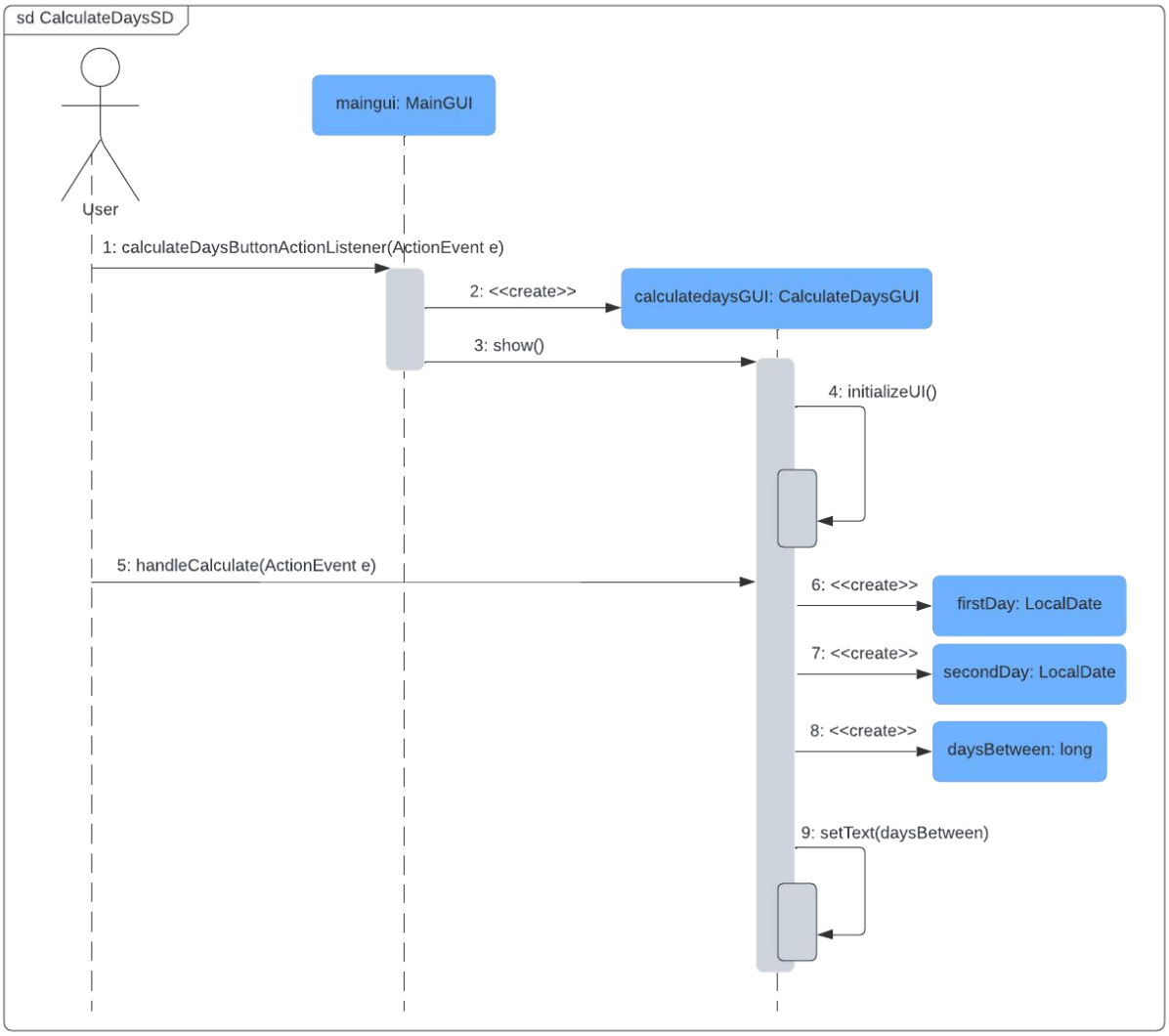
| UC number: 3 | |
|---|---|
| Overview | Allows the user to calculate the days difference between two dates |
| Related Use Case | None |
| Actors | User |

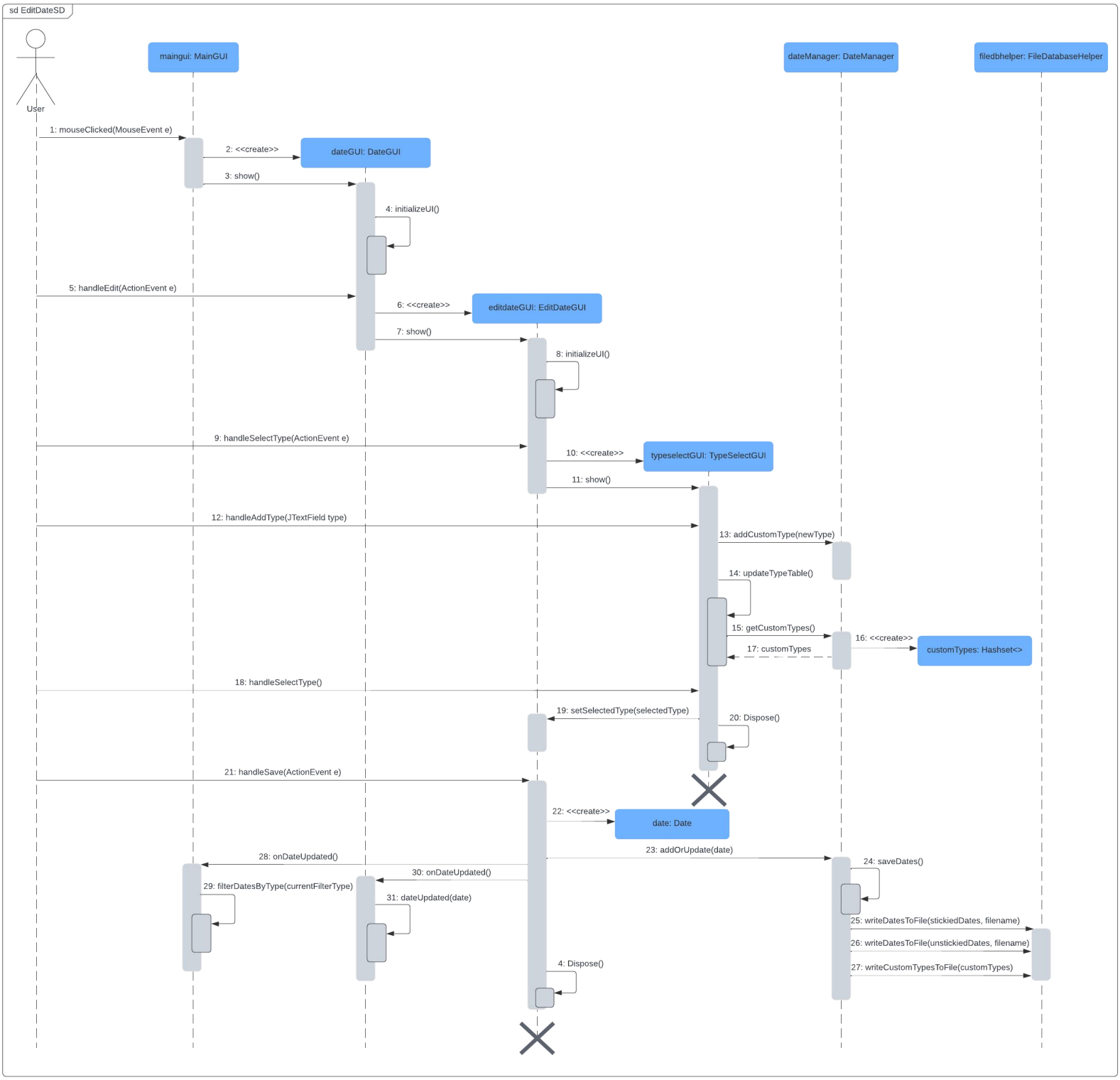
| UC number: 4 | |
|---|---|
| Overview | Allows the user to change the type of dates displayed |
| Related Use Case | None |
| Actors | User |

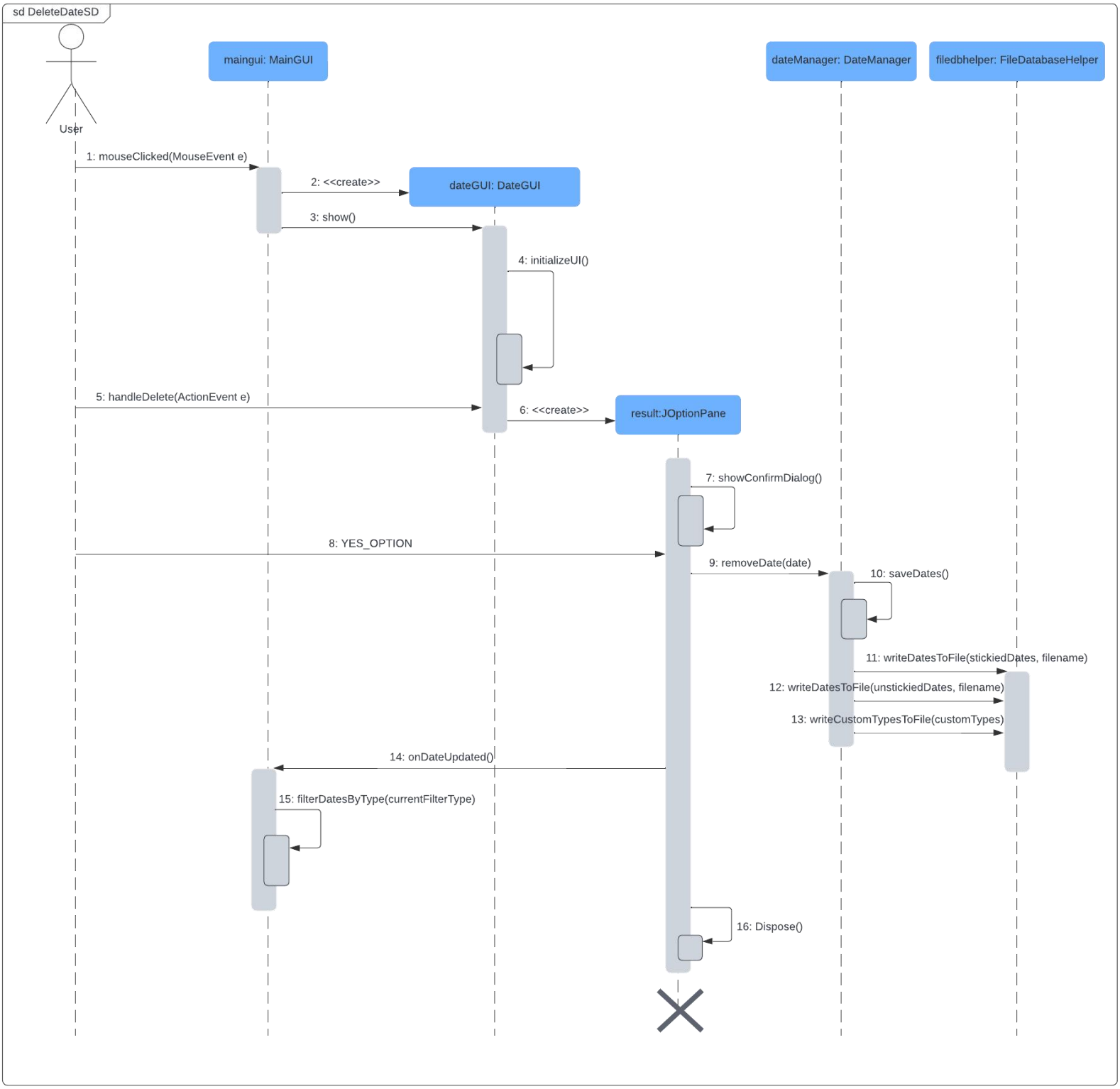
# System Design

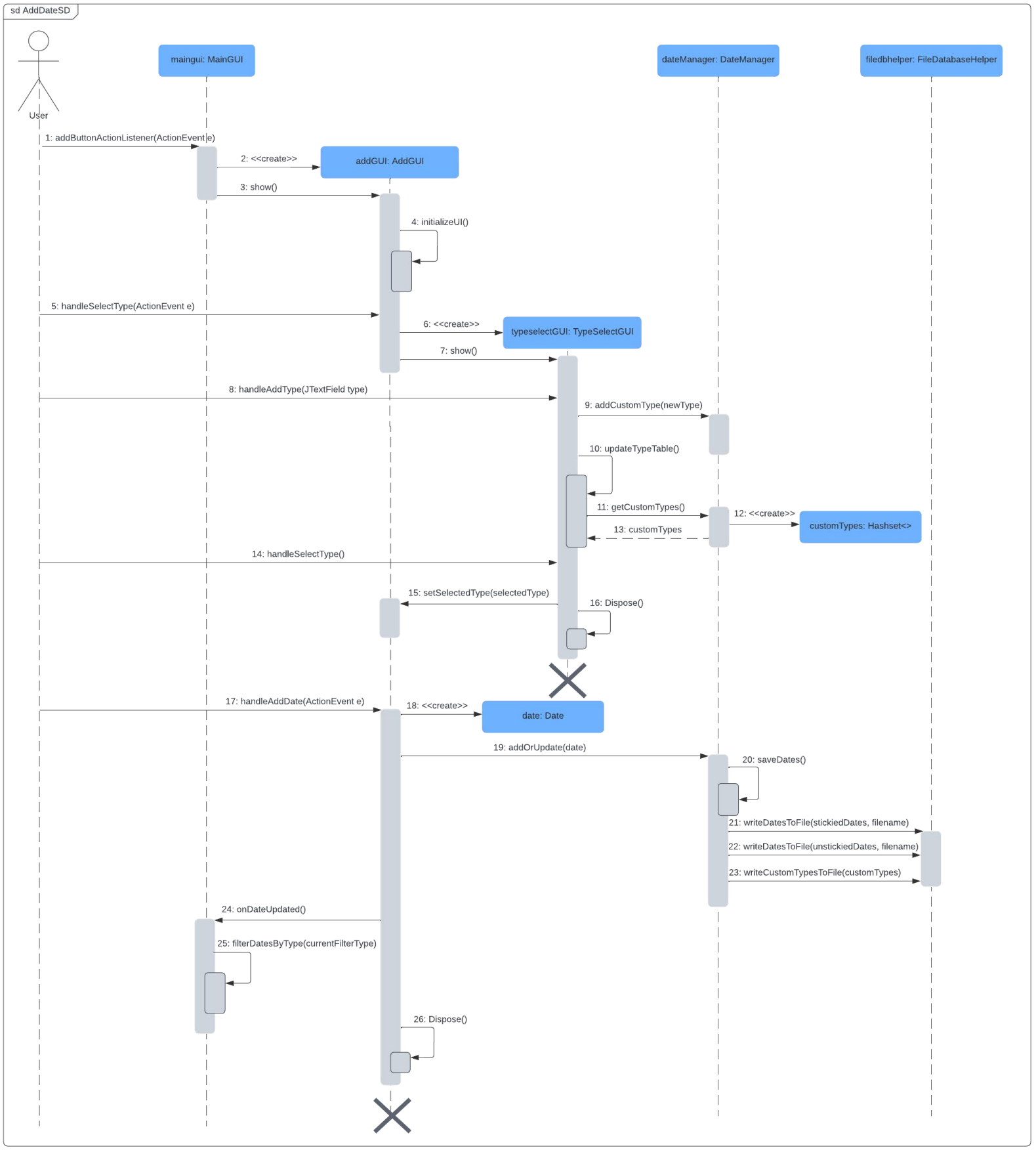
## Sequence Diagrams

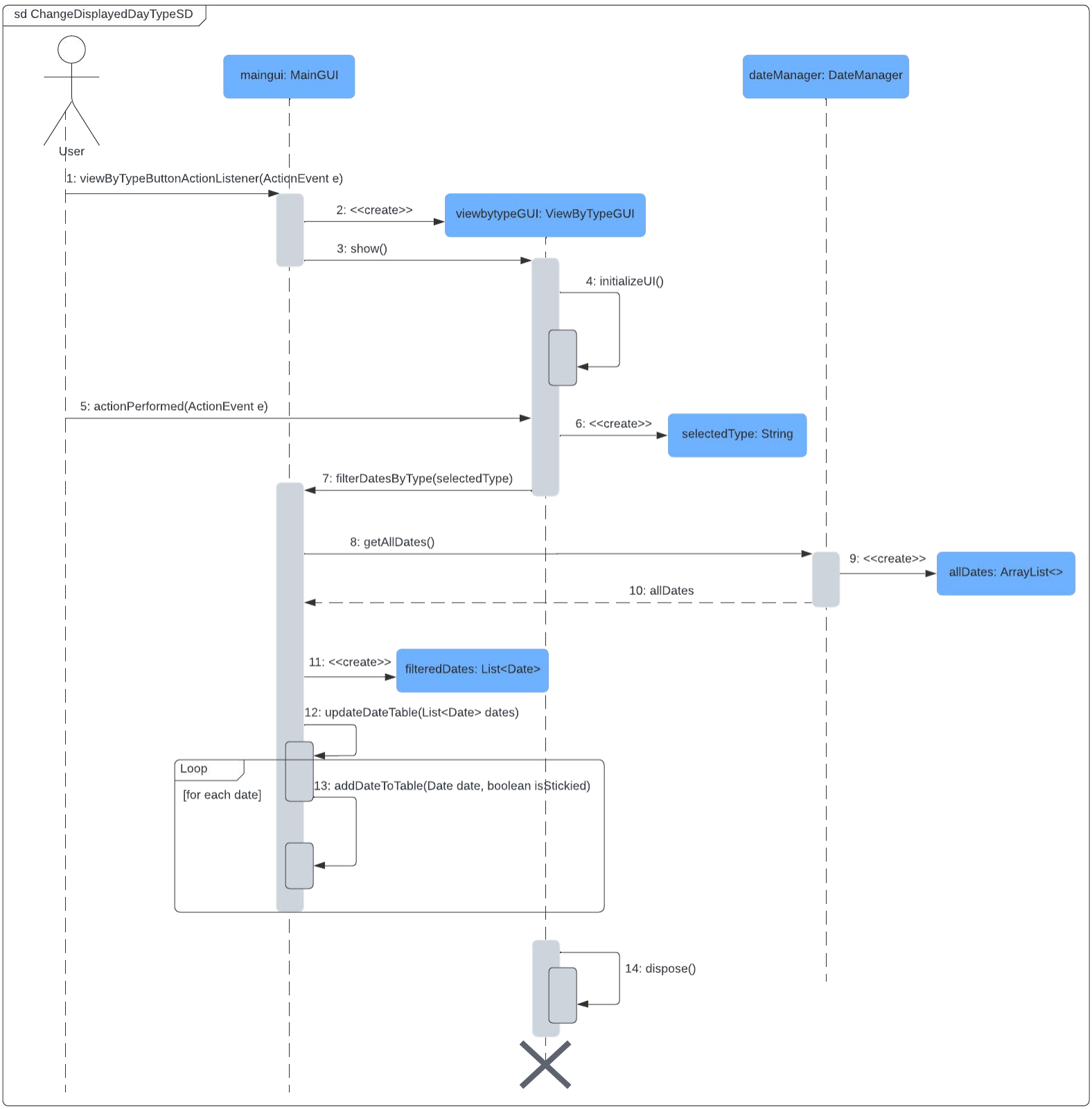


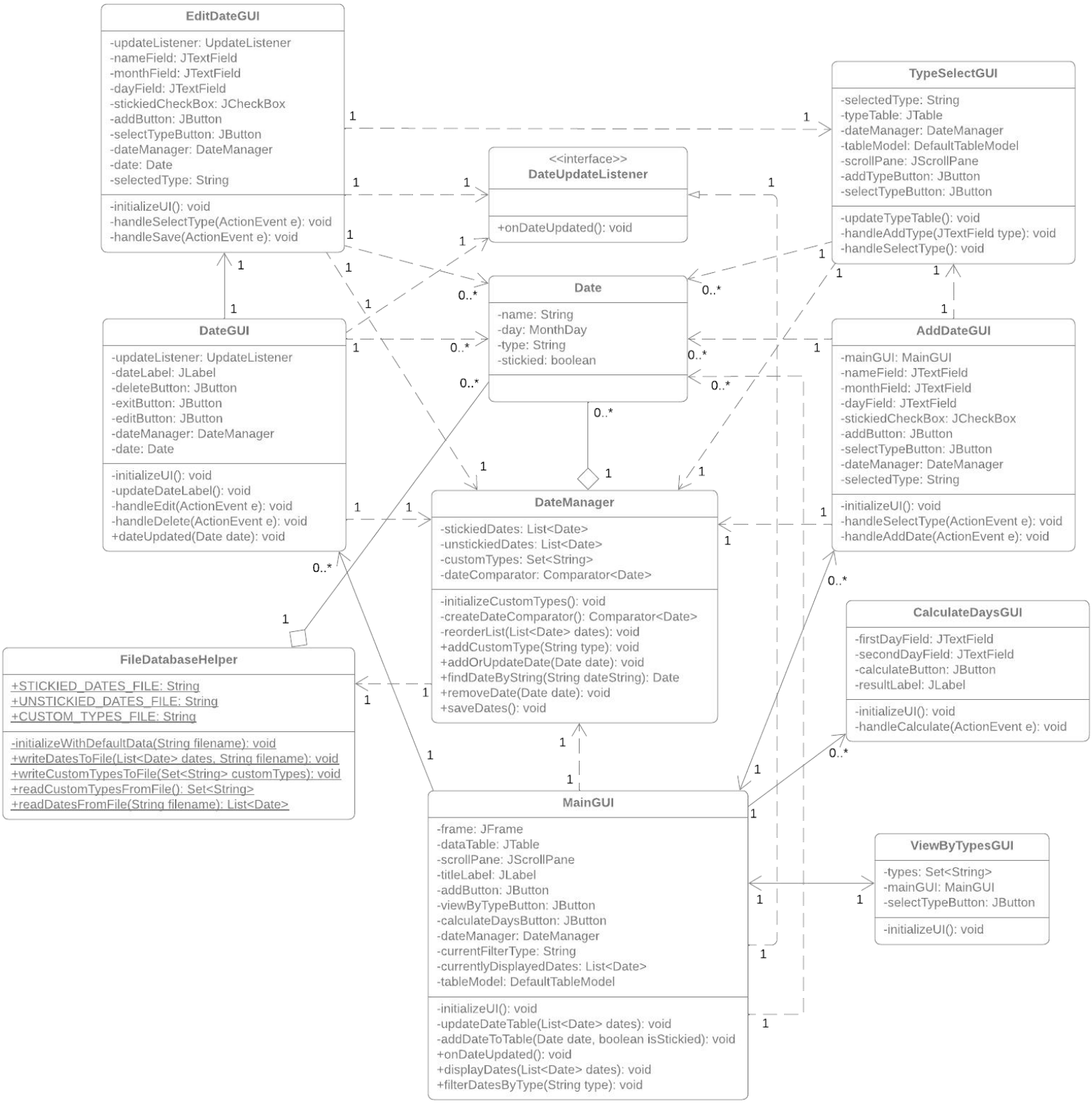

sd EditDateSD
User
maingui: MainGUI
dateManager: DateManager
filedbhelper: FileDatabaseHelper
1: mouseClicked(MouseEvent e)
2: <<create>>
dateGUI: DateGUI
3: show()
4: initializeUI()
5: handleEdit(ActionEvent e)
6: <<create>>
editdateGUI: EditDateGUI
7: show()
8: initializeUI()
9: handleSelectType(ActionEvent e)
10: <<create>>
typeselectGUI: TypeSelectGUI
11: show()
12: handleAddType(JTextField type)
13: addCustomType(newType)
14: updateTypeTable()
15: getCustomTypes()
16: <<create>>
customTypes: Hashset<>
17: customTypes
18: handleSelectType()
19: setSelectedType(selectedType)
20: Dispose()
21: handleSave(ActionEvent e)
22: <<create>>
date: Date
23: addOrUpdate(date)
24: saveDates()
25: writeDatesToFile(stickiedDates, filename)
26: writeDatesToFile(unstickiedDates, filename)
27: writeCustomTypesToFile(customTypes)
28: onDateUpdated()
29: filterDatesByType(currentFilterType)
30: onDateUpdated()
31: dateUpdated(date)
4: Dispose()


sd DeleteDateSD
User
maingui: MainGUI
dateGUI: DateGUI
dateManager: DateManager
filedbhelper: FileDatabaseHelper
result:JOptionPane
1: mouseClicked(MouseEvent e)
2: <<create>>
3: show()
4: initializeUI()
5: handleDelete(ActionEvent e)
6: <<create>>
7: showConfirmDialog()
8: YES_OPTION
9: removeDate(date)
10: saveDates()
11: writeDatesToFile(stickiedDates, filename)
12: writeDatesToFile(unstickiedDates, filename)
13: writeCustomTypesToFile(customTypes)
14: onDateUpdated()
15: filterDatesByType(currentFilterType)
16: Dispose()


sd AddDateSD
User
maingui: MainGUI
addGUI: AddGUI
dateManager: DateManager
filedbhelper: FileDatabaseHelper
typeselectGUI: TypeSelectGUI
customTypes: Hashset<>
date: Date
1: addButtonActionListener(ActionEvent e)
2: <<create>>
3: show()
4: initializeUI()
5: handleSelectType(ActionEvent e)
6: <<create>>
7: show()
8: handleAddType(JTextField type)
9: addCustomType(newType)
10: updateTypeTable()
11: getCustomTypes()
12: <<create>>
13: customTypes
14: handleSelectType()
15: setSelectedType(selectedType)
16: Dispose()
17: handleAddDate(ActionEvent e)
18: <<create>>
19: addOrUpdate(date)
20: saveDates()
21: writeDatesToFile(stickiedDates, filename)
22: writeDatesToFile(unstickiedDates, filename)
23: writeCustomTypesToFile(customTypes)
24: onDateUpdated()
25: filterDatesByType(currentFilterType)
26: Dispose()


sd ChangeDisplayedDayTypeSD
User
maingui: MainGUI
dateManager: DateManager
1: viewByTypeButtonActionListener(ActionEvent e)
2: <<create>>
viewbytypeGUI: ViewByTypeGUI
3: show()
4: initializeUI()
5: actionPerformed(ActionEvent e)
6: <<create>>
selectedType: String
7: filterDatesByType(selectedType)
8: getAllDates()
9: <<create>>
allDates: ArrayList<>
10: allDates
11: <<create>>
filteredDates: List<Date>
12: updateDateTable(List<Date> dates)
Loop
[for each date]
13: addDateToTable(Date date, boolean isStickied)
14: dispose()

# Class Diagram



# Conclusion

## Difficulties

I had trouble implementing the MainGUI table as I did not have prior experience. The table structure I envisioned was complicated, and I had to use some inefficient tricks to make it function properly. Specifically, it was difficult to separate the table into stickied and unstickied statuses and highlight only the stickied dates.

Formatting the GUIs by purely changing the code was tedious as I had to move them step by step to where I wanted them to be. I learned a ton of details around java's swing package as I utilized many features that were not covered during class. Some examples may include BorderFactory, HorizontalGlue, TableCellRenderer, Jdialog, and many more.

## Next Steps:

Throughout the project, there were many design details with room of improvement. Specifically, updating tables and lists were not optimized as I simply used sort() instead of potentially implementing a more suitable and efficient data structure such as a PriorityQueue.

Secondly, the UI's design could be improved drastically. I only built a simple basis of the GUIs due to how much time I committed on learning and implementing Jframe. Moreover, if I were to expand on this project, I would implement a feature where the application is always stickied on the user's desktop for convenience.

A final improvement could be to change the user data's storage method to using an actual database like MySQL to better support the application. It would also be a huge improvement to implement a user login system so each user can have their own dates stored in a common database instead of having to store their data in a local file.

# Appendix


https://www.geeksforgeeks.org/java-time-localdate-class-in-java/

https://www.geeksforgeeks.org/java-jframe/

https://www.geeksforgeeks.org/java-swing-jdialog-examples/

https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/javax/swing/Box.html

https://docs.oracle.com/javase/8/docs/api/java/time/temporal/ChronoUnit.html

https://www.geeksforgeeks.org/java-swing-jtable/

https://docs.oracle.com/javase%2Ftutorial%2Fuiswing%2F%2F/events/mouselistener.html

https://www.geeksforgeeks.org/comparator-interface-java/

https://docs.oracle.com/javase/8/docs/api/javax/swing/table/DefaultTableCellRenderer.html

https://www.tutorialspoint.com/swing/swing_windowadapter.htm