Name - Soham Balkothe
Student ID - 1001500980
Email - svb0980@mav.uta.edu

1. Expanded use case for Manage Items

Actor: User	System: Application
	0. The application displays four buttons (add, delete, undo, redo) and an empty table.
1. TUCBW the user clicks on the add button.	2. A dialog asks the user to enter the item ID and content.
3. The user enters the item ID and content and clicks the OK button to add the item.	*4. The table gets updated and the entered item is added to it.
5. The user selects the entered item and selects from the table clicks the delete button.	*6. The table gets updated and the item is deleted from it.
7. The user sees the table with the item deleted and clicks on the undo button.	*8. The table gets updated and the entered item is added to it.
9. The user sees the table with the previously entered item and clicks the redo button.	*10. The table gets updated and the item is deleted from it.
11. TUCEW the user sees the table with the item deleted	

2. Scenarios for each non-trivial step

We have 4 non-trivial steps in the expanded use case - steps 4, 6, 8, and 10.

Scenario for Step 4

3. The user enters the item ID and content and clicks the OK button to add the item.

4.1. The GUI sends the entered item to the controller.
4.2. The controller creates an instance of the item class and the AddCommand class and calls it execute method.
4.3. The AddCommand updates the itemsList by adding the item entered by user.
4.4. The AddCommand returns the updated itemsList back to the controller.
4.5. The controller pushes the instance of AddCommand to the UndoStack and the RedoStack.
4.6. The Controller returns the updated itemsList to the GUI.
4.7. The GUI displays the items from the itemsList in the table.

5. The user selects the entered item and selects from the table clicks the delete button.

Scenario for Step 6

5. The user selects the entered item and selects from the table clicks the delete button.

6.1. The GUI sends the selected item to the controller.
6.2. The controller creates an instance of the item class and the DeleteCommand class and calls it execute method.
6.3. The DeleteCommand updates the itemsList by removing the item selected by user.
6.4. The DeleteCommand returns the updated itemsList back to the controller.
6.5. The controller pushes the instance of DeleteCommand to the UndoStack and the RedoStack.
6.6. The Controller returns the updated itemsList to the GUI.
6.7. The GUI displays the items from the itemsList in the table.

7. The user sees the table with the item deleted and clicks on the undo button.

Scenario for Step 8

7. The user sees the table with the item deleted and clicks on the undo button.

8.1. The GUI sends the undo request to the controller.
8.2. The controller checks if the UndoStack is empty.
8.3. If the UndoStack is not empty, the controller pops out the topmost Command object from the UndoStack and calls its undo method.
8.4. The Command class adds the item associated with it to the itemsList.
8.5. The Command class returns the updated itemsList to the controller.
8.6. The controller return the itemsList to the GUI.
8.7. The GUI displays the items from the itemsList in the table.

9. The user sees the table with the previously entered item and clicks the redo button.

Scenario for Step 10

9. The user sees the table with the previously entered item and clicks the redo button.

10.1. The GUI sends the redo request to the controller.
10.2. The controller checks if the RedoStack is empty.
10.3. If the RedoStack is not empty, the controller pops out the topmost Command object from the RedoStack and calls its redo method.
10.4. The Command class adds the item associated with it to the itemsList.
10.5. The Command class returns the updated itemsList to the controller.
10.6. The controller return the itemsList to the GUI.
10.7. The GUI displays the items from the itemsList in the table.

11. the user sees the table with the item deleted

3. Scenario Tables for each scenario

Scenario table for step 4:
Table 1

Step	Subject	Subject action	Other data/objects	Object acted upon
3	The user	adds	Item details	GUI
4.1.	The GUI	sends	item	the controller
4.2	The controller	calls	execute method	AddCommand
4.3.	AddCommand	adds	item	itemList
4.4	AddCommand	returns	itemsList	Controller
4.5.	Controller	pushes	AddCommand object	UndoStack and RedoStack
4.6.	Controller	returns	Updated itemsList	GUI
4.7.	GUI	displays	The items	User

Scenario table for step 6

Table 2

Step	Subject	Subject action	Other data/objects	Object acted upon
5	The user	Clicks delete	row	GUI
6.1	The GUI	sends	item	the controller
6.2	The controller	calls	execute method	DeleteCommand
6.3.	DeleteCommand	deletes	item	itemList
6.4	DeleteCommand	returns	itemsList	Controller
6.5.	Controller	pushes	DeleteCommand object	UndoStack and RedoStack
6.6.	Controller	returns	Updated itemsList	GUI
6.7.	GUI	displays	The items	User

Scenario table for step 6:
Table 3

Step	Subject	Subject action	Other data/objects	Object acted upon
7	The user	Clicks undo		GUI
8.1.	The GUI	sends	Undo request	the controller
8.2	The controller	checks	the undostack	Controller
8.3.	Controller	Calls undo	Topmost stack command	Command object
8.4	Command	adds	item	itemsList
8.5.	Command	returns	itemsList	Controller
8.6.	Controller	returns	Updated itemsList	GUI
8.7.	GUI	displays	The item	User

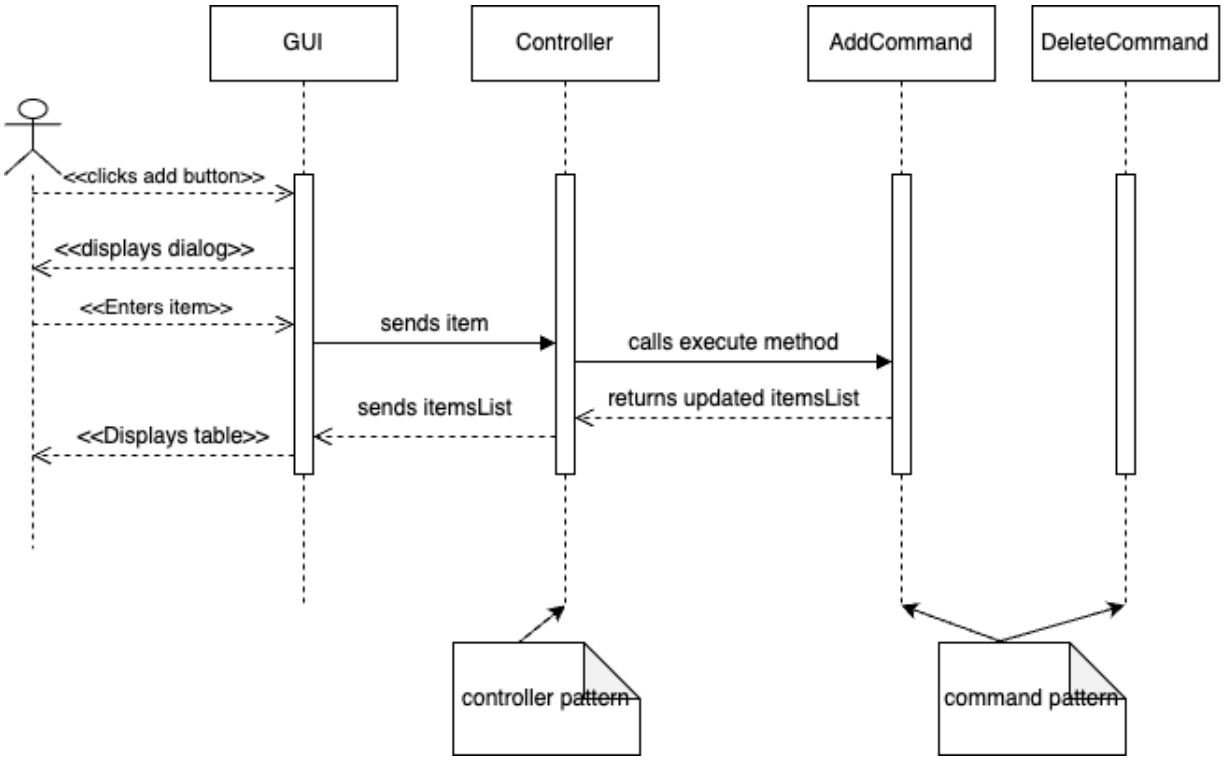
Scenario table for step 8:

Table 4

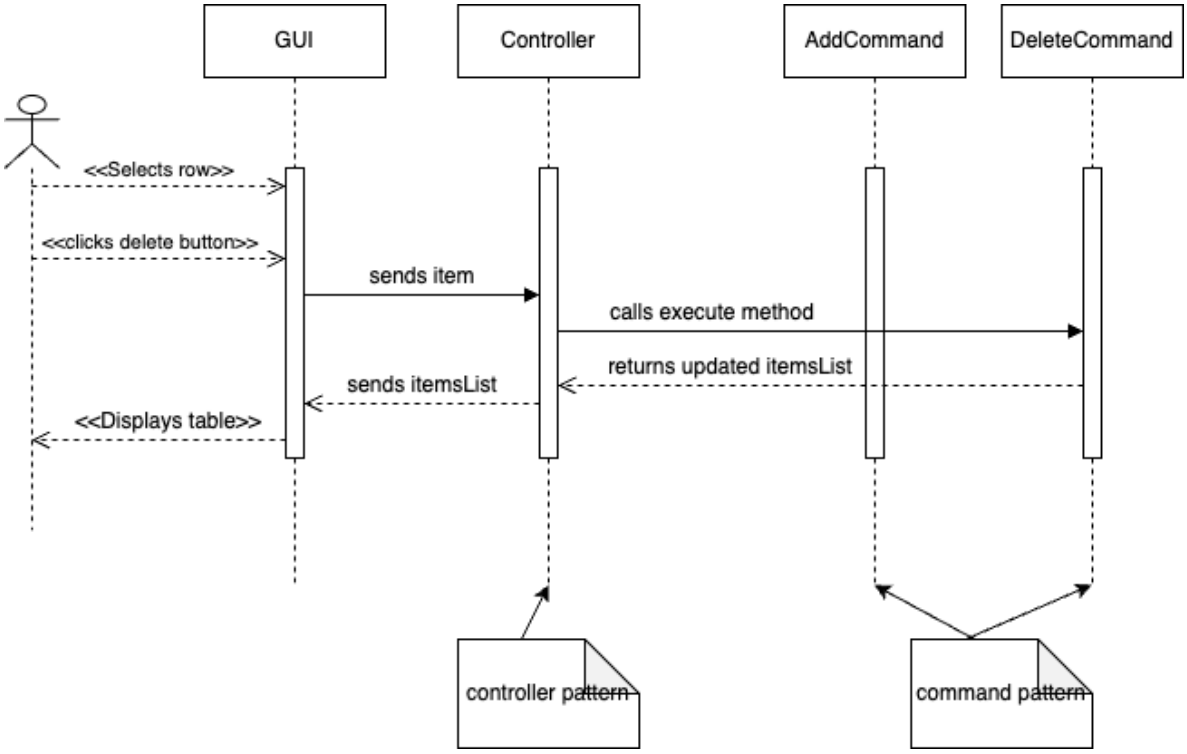
Step	Subject	Subject action	Other data/objects	Object acted upon
7	The user	Clicks redo		GUI
8.1.	The GUI	sends	redo request	the controller
8.2	The controller	checks	The redostack	Controller
8.3.	Controller	Calls undo	Topmost stack command	Command object
8.4	Command	deletes	item	itemsList
8.5.	Command	returns	itemsList	Controller
8.6.	Controller	returns	Updated itemsList	GUI
8.7.	GUI	displays	The item	User

4. Informal Sequence Diagrams for each scenario table

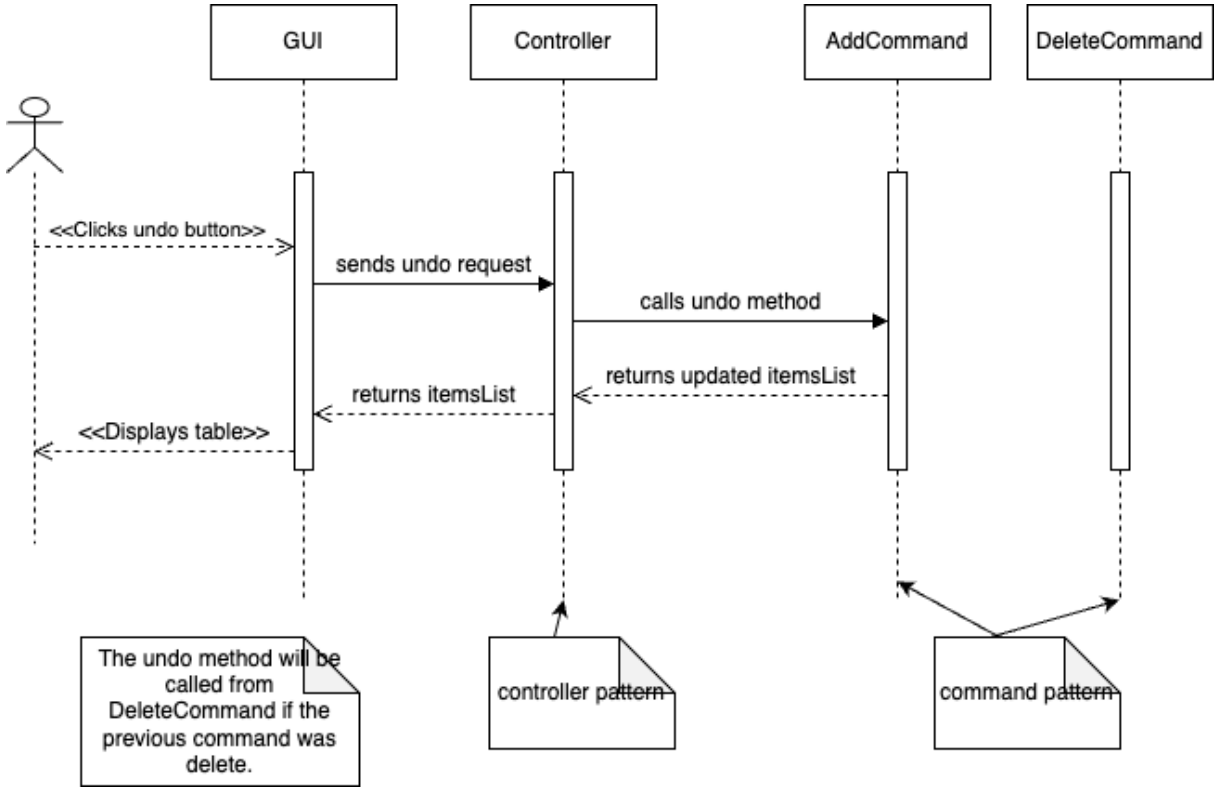
Informal Sequence Diagram for Table 1.



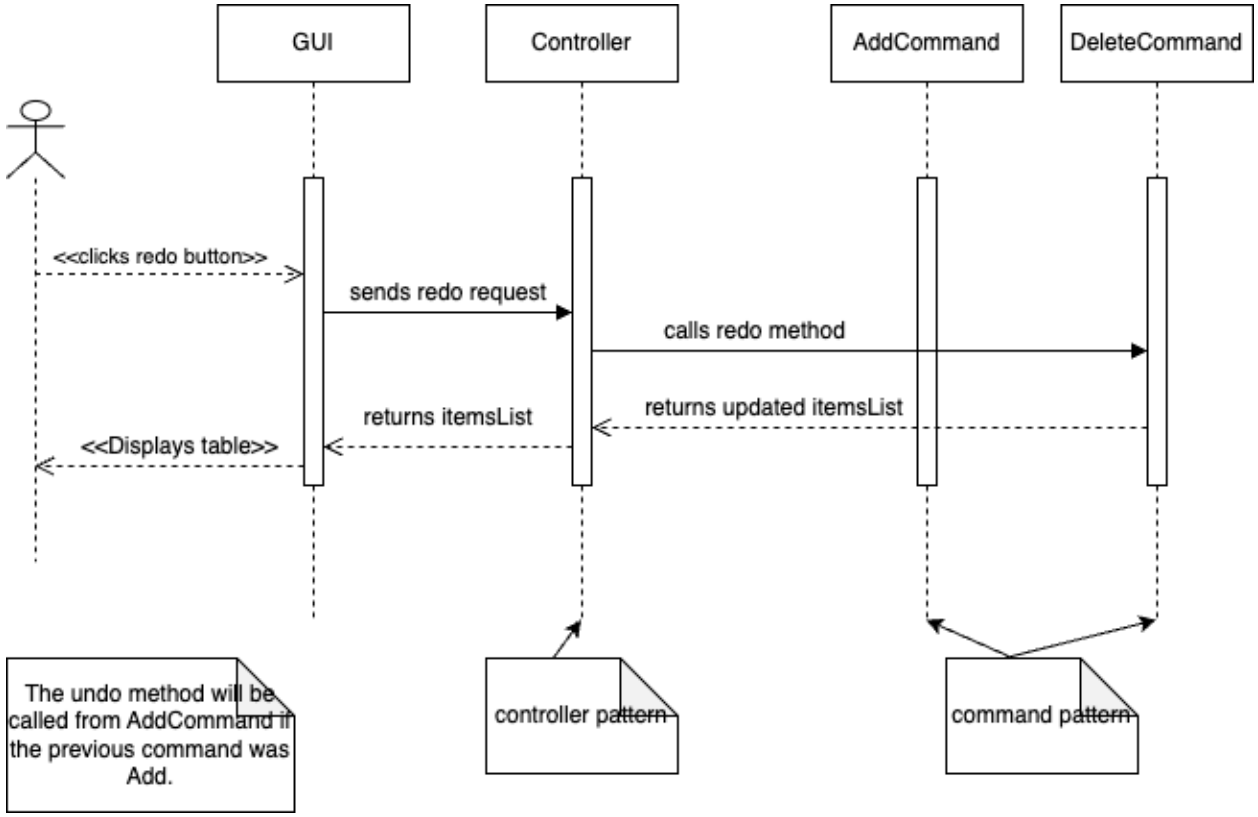
Informal Sequence Diagram for Table 2



Informal Sequence Diagram for Table 3

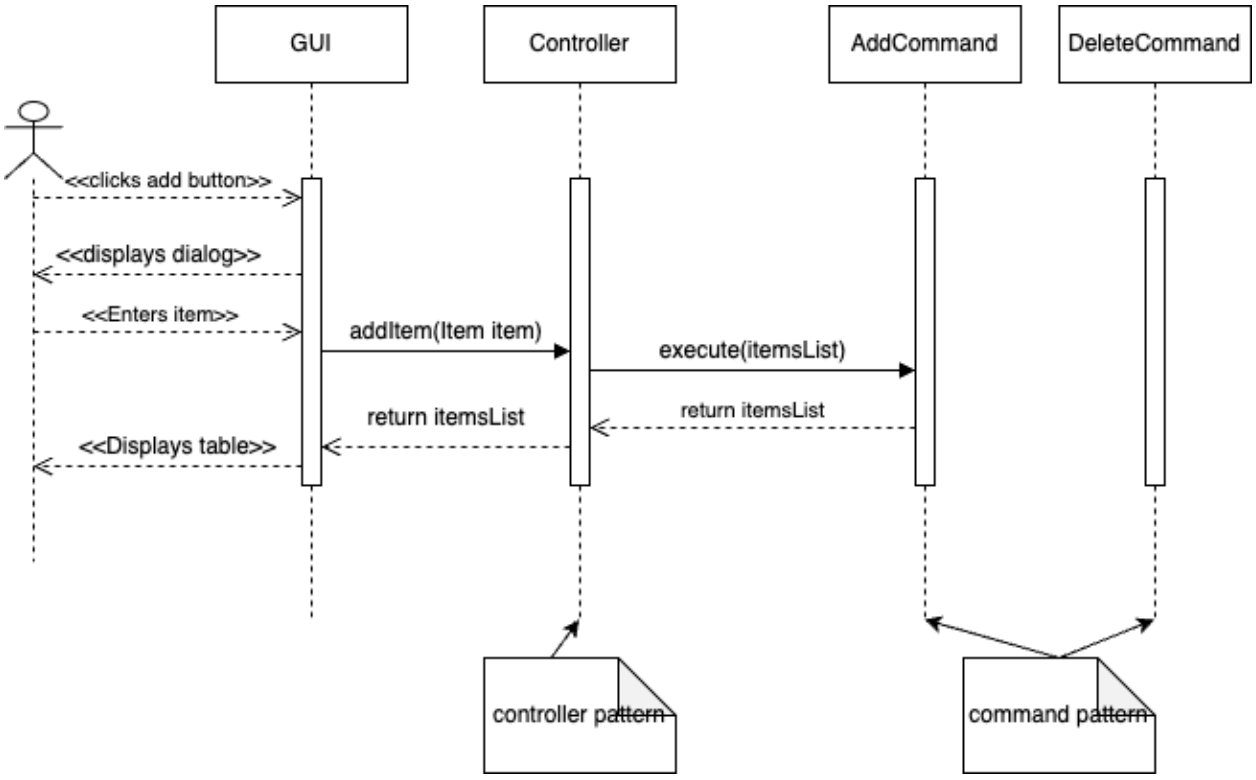


Informal Sequence Diagram for Table 4

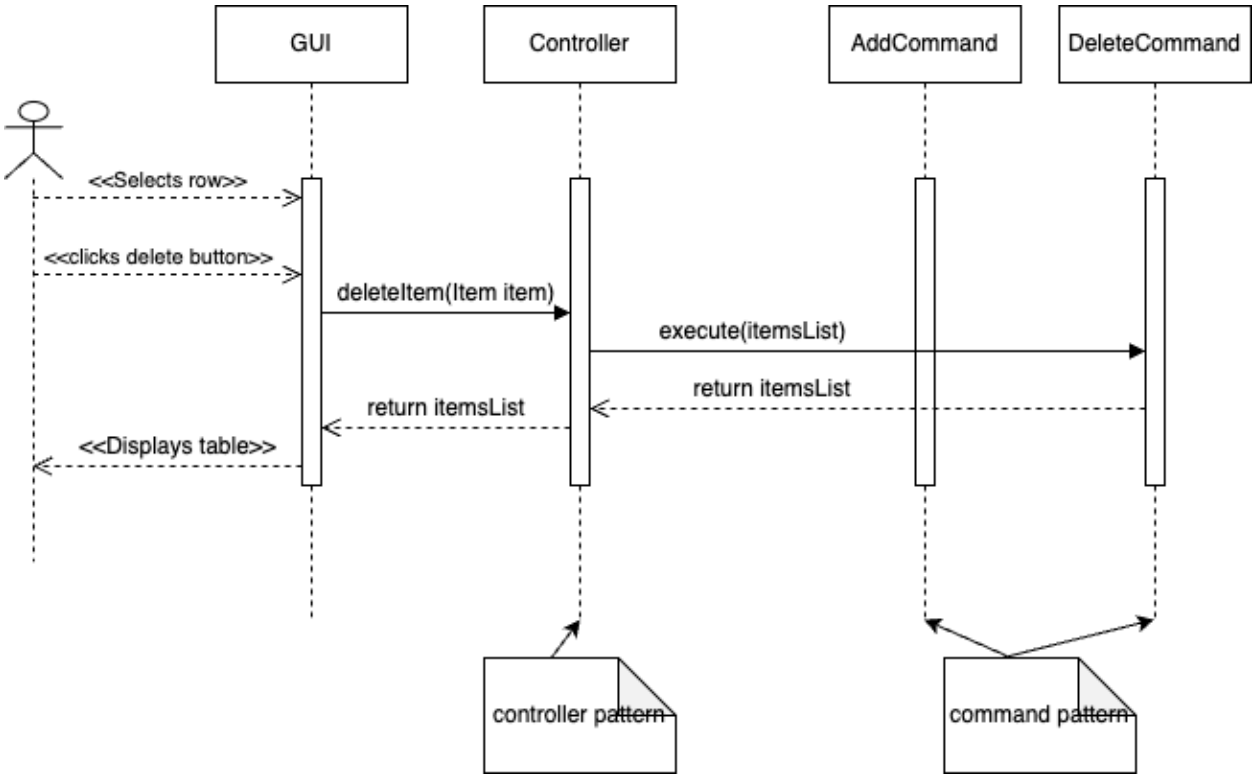


5. Design Sequence Diagrams for each informal sequence diagram

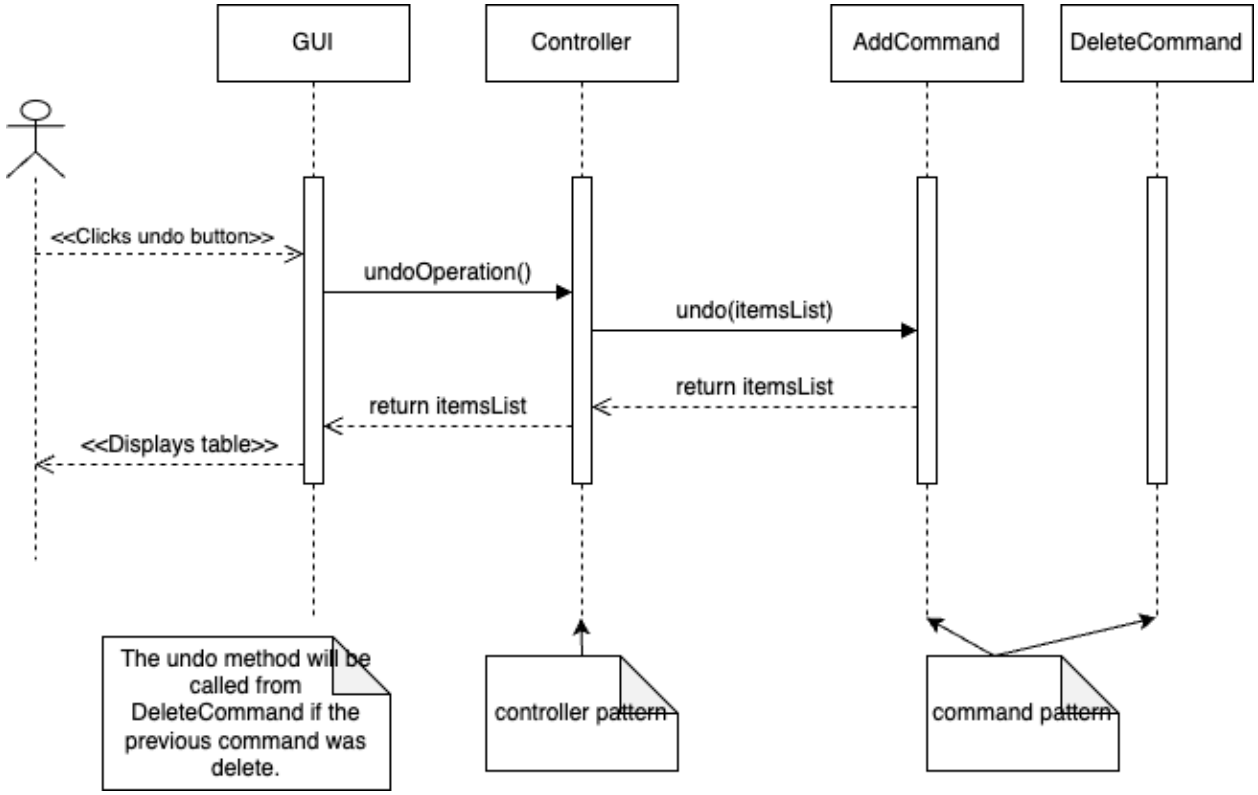
Design Sequence Diagram for Informal Sequence Diagram 1.



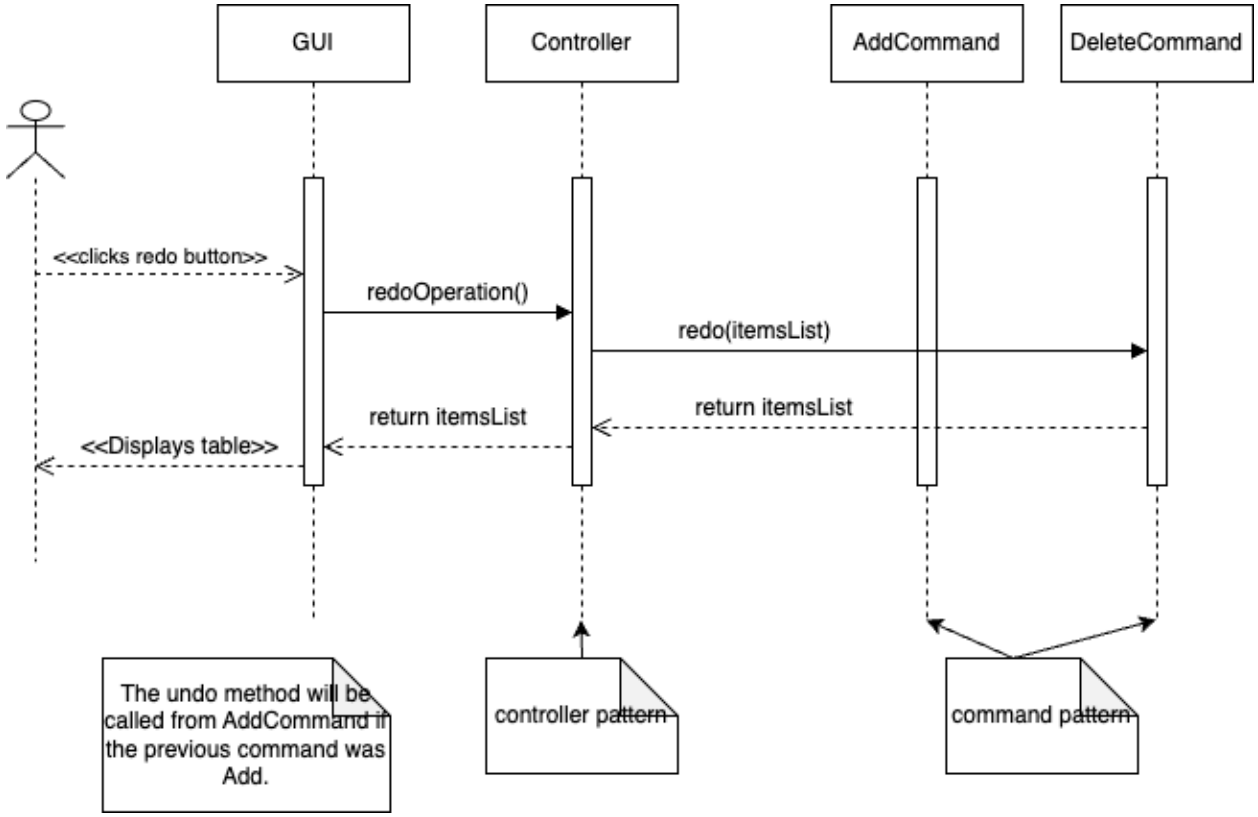
Design Sequence Diagram for Informal Sequence Diagram 2.



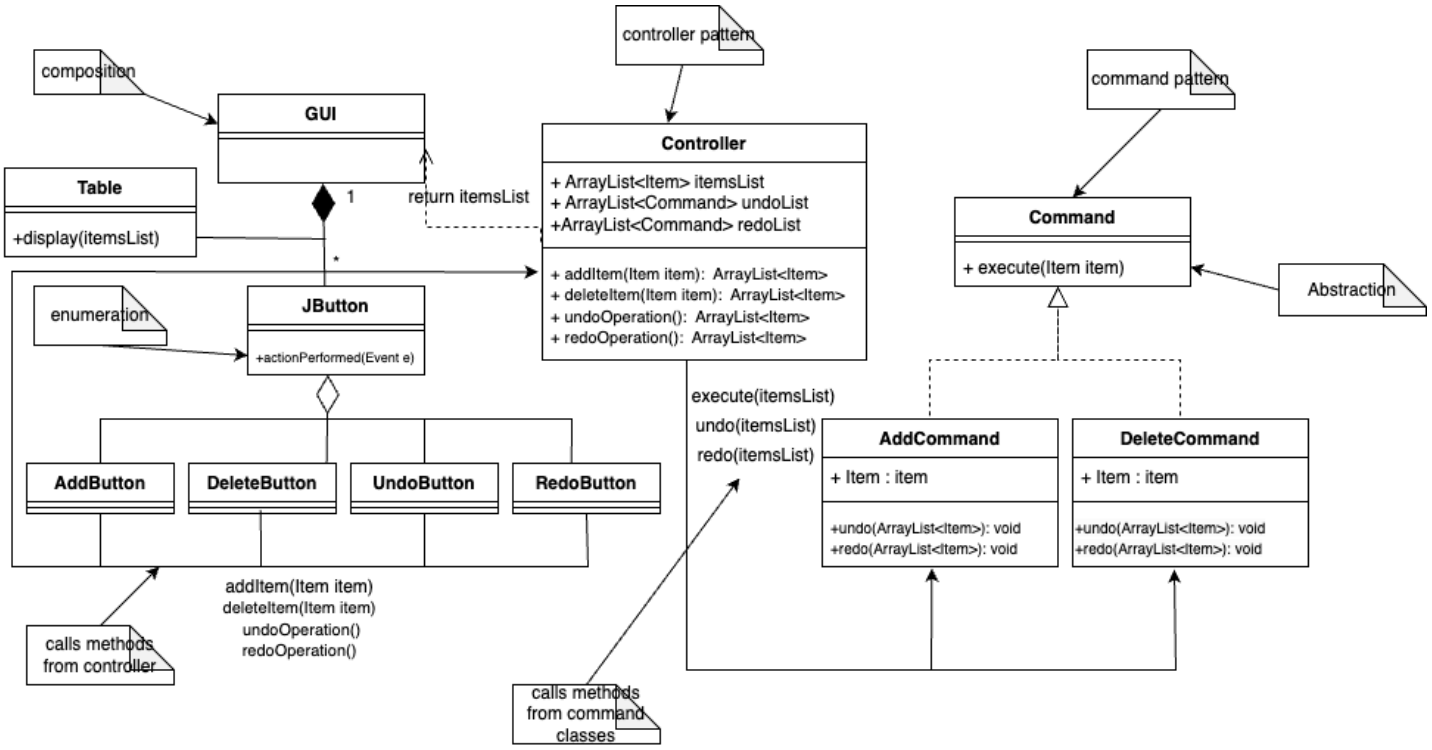
Design Sequence Diagram for Informal Sequence Diagram 3.



Design Sequence Diagram for Informal Sequence Diagram 4.



6. Design Class Diagram



7. Application implemented using java in implementation folder.