**Name: Poorna Pallavi Tekkali**

**Part -1**

**1.Expanded Use-Case:**

**ADD:**

|  |  |
| --- | --- |
| Pre-condition: User Executes the Table Application | |
| Actor: User | System: Modifying Table |
| The Application is running already | The GUI displays a panel where user could see an empty space and four buttons: Add, Delete, Undo, Redo |
| User can select an option add | Now a new row and column gets displayed with ID and Item text. |
| User then enters ID and enters some text in boxes. | System will now take input and displays the text. |
| Post-Condition: The application displays all text added by user. | |

**Overview:**

The program opens a window with an empty table in it. Users can add, remove, undo, and redo using the four buttons at the bottom. When the user clicks the add button, a row is created with ID and Item text. The user now makes any changes to add the text according to the requirement and system displays it.

|  |  |
| --- | --- |
| Pre-condition: User Executes the Table Application | |
| Actor: User | System: Modifying Table-Using Delete |
| The Application is running already | The GUI displays a panel where user could see an empty space and four buttons: Add, Delete, Undo, Redo |
| User can select any row from the input given | Now that entire row is selected/highlighted. |
| User then selects Delete option. | System will now delete the entire row selected by user. |
| Post-Condition: The application deletes the row selected. | |

**Delete:**

**Overview:**

A window with an empty table appears when the application first launches. The user can add, remove, undo, and redo using the four buttons at the bottom. The user must first pick the row he wishes to delete from the table by clicking the delete button, then the selected row gets deleted.

**Undo:**

|  |  |
| --- | --- |
| Pre-condition: User Executes the Table Application | |
| Actor: User | System: Modifying Table-Using Undo |
| The Application is running already | The GUI displays a panel where user could see an empty space and four buttons: Add, Delete, Undo, Redo |
| User now selects the undo option. | After interpreting the user's instructions, the system will retrieve the most recent row index and remove it. |
| Post-Condition: The application performs undo operation. | |

**Overview:**

When a user selects the undo button, the controller will retrieve the index of the most recently inserted row and erase the most recently added row.

|  |  |
| --- | --- |
| Pre-condition: User Executes the Table Application | |
| Actor: User | System: Modifying Table-Using Redo |
| The Application is running already | The GUI displays a panel where user could see an empty space and four buttons: Add, Delete, Undo, Redo |
| User now selects the Redo option. | The system will retrieve the recently deleted row, restore it, and display it in the table once again after interpreting the user's commands. |
| Post-Condition: The application performs redo operation. | |

**REDO:**

**Overview:**

The recently deleted row index will be obtained by the controller and added back into the table if the user clicks the redo button.

**2.Nontrivial Step of Expanded Use Case:**

**Scenarios:**

* **User selecting Add button:**

1. The user clicks the view's "Add" button.

2. The controller receives an event from the view.

3. The controller creates a new instance of the AddCommand class and passes the newly entered content and ID to it.

4. The controller gives the executeCommand() method the AddCommand object.

5.The executeCommand() function updates the database model by adding a new row with the supplied ID and content, which calls the redo() method of the AddCommand object.

6.The AddCommand object is sent to the undo stack in step 6.

7. The updated view shows the new row.

* **User selecting Delete button:**

1. The user selects a row in the display by clicking on it.

2. The user selects the "Delete" button.

3. When an event occurs, the view notifies the controller.

4. The controller builds a fresh instance of the DeleteCommand class using the ID and content of the selected row as inputs.

5. The controller provides the executeCommand() method with the DeleteCommand object.

6. The redo() method of the DeleteCommand object's executeCommand() method is used to remove the selected row from the table model.

7. The object DeleteCommand is now part of the undo stack.

8. The specified row is deleted from the table.

* **User selecting Undo button:**

1. The user selects the "undo" option.

2. The event is sent to the ActionListener in the controller for the undo button.

3. The ActionListener searches the undo stack to see whether any commands are present.

4. It pops the most recently executed command from the undo stack if the undo stack is not empty.

5. Following that, the ActionListener invokes the popped command's undo function.

6. The command's undo mechanism reverses the most recently executed command.

7. Following that, the actionPerformed method adds the undone command to the redo stack.

8. The modifications made by the undone command are updated in the application by the actionPerformed method.

* **User selecting Redo button:**

1. When the "Redo" button is clicked, the actionPerformed method in the ActionListener class for the "Redo" button is called.

2. With "redo" as an argument, the actionPerformed method invokes the executeUndoRedo method in the controller class.

3. The executeRedo method determines whether the stack of redoable commands is empty.

4. The executeRedo function calls the redo method of the command at the top of the stack, if it is not empty.

5. By undoing the previous action taken by the command, the redo method of the command modifies the state of the model object, which in this case is the JTable component.

6. The command is then added to the stack of undoable commands via the executeRedo function.

7. The JTable component is updated to reflect the modifications.

**3.Scenario tables:**

**1.ADD**

|  |  |  |
| --- | --- | --- |
| Object | Action | Object Affected |
| User | Selects Add button | - |
| View | Notifies the Controller of an event | - |
| Controller | Makes a new AddCommand instance. | - |
| Controller | ID and Item name are sent to the AddCommand. | - |
| Controller | Calls executeCommand(AddCommand) | - |
| ExecuteCommand(AddCommand) | Calls AddCommand's redo() function. | A new row is created with ID and Item Text. |
| AddCommand | Is delivered to the Undo Stack | - |

**2.DELETE**

|  |  |  |
| --- | --- | --- |
| Object | Action | Object Affected |
| User | Selects a row | - |
| User | Clicks "Delete" button | - |
| View | Notifies the controller of the event | - |
| Controller | generates a new DeleteCommand instance. | - |
| Controller | Provides ID and Item text of the selected row to DeleteCommand | - |
| Controller | Calls executeCommand(DeleteCommand) | - |
| ExecuteCommand(DeleteCommand) | Calls the redo() method of DeleteCommand | Selected row gets deleted. |
| DeleteCommand | Is added to the Undo Stack | - |
| Table Model | Deletes the specified row | - |

**3.UNDO:**

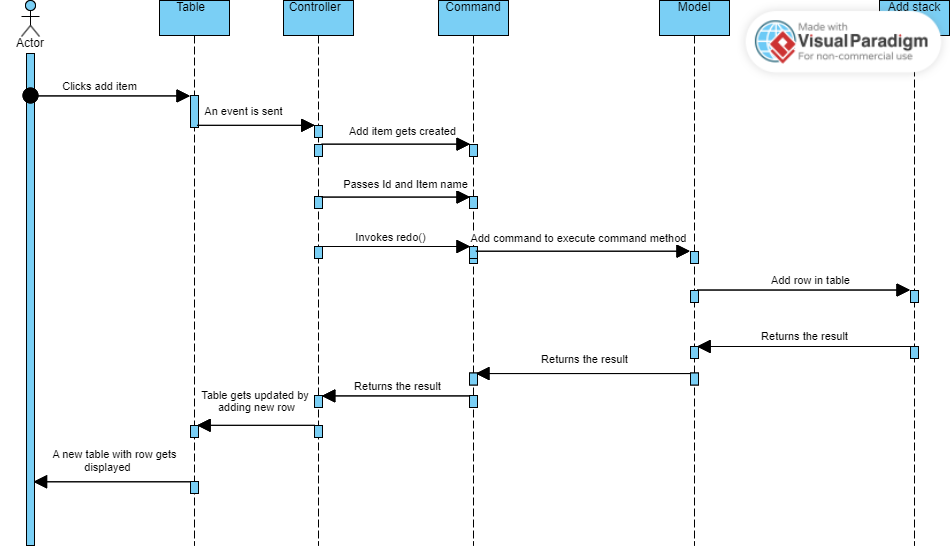
|  |  |  |
| --- | --- | --- |
| Object | Action | Object Affected |
| User | User selects "undo" option | - |
| Controller | The "undo" button's ActionListener receives the event. | - |
| Controller | searches for accessible commands in the undo stack | - |
| Controller | If undo stack is not empty, pops the most recent command | - |
| ExecuteCommand(UndoCommand) | Calls the undo() method of the popped command | reverses the most recent command execution |
| UndoCommand | Is moved to the top of the redo stack | - |
| Application | Applies the modifications made by the undo command to the program. | - |

**4.REDO**

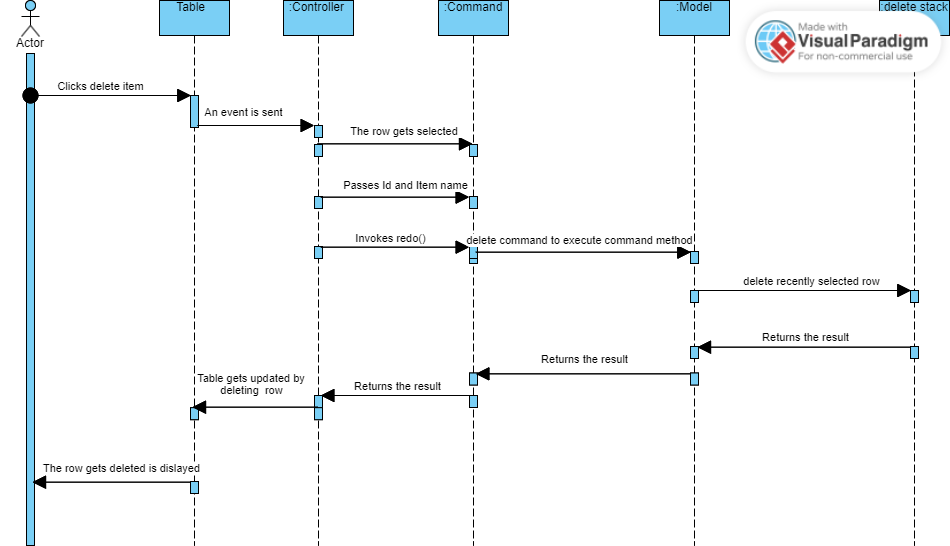
|  |  |
| --- | --- |
| Object | Action |
| User | Clicks Redo button |
| Controller | Actionperfomed() is called |
| Controller | Uses the parameter "redo" to invoke the executeRedo function. |
| Controller | Invokes the executeRedo function with the "redo" option. |
| Controller | pulls the top command out of the stack if it's not empty. |
| Redo command | Calls redo() |
| Redo command | Updates table by performing the previous action. |
| Controller | Pushes command and displays the changes made. |

**4.Informal Sequence diagram:**

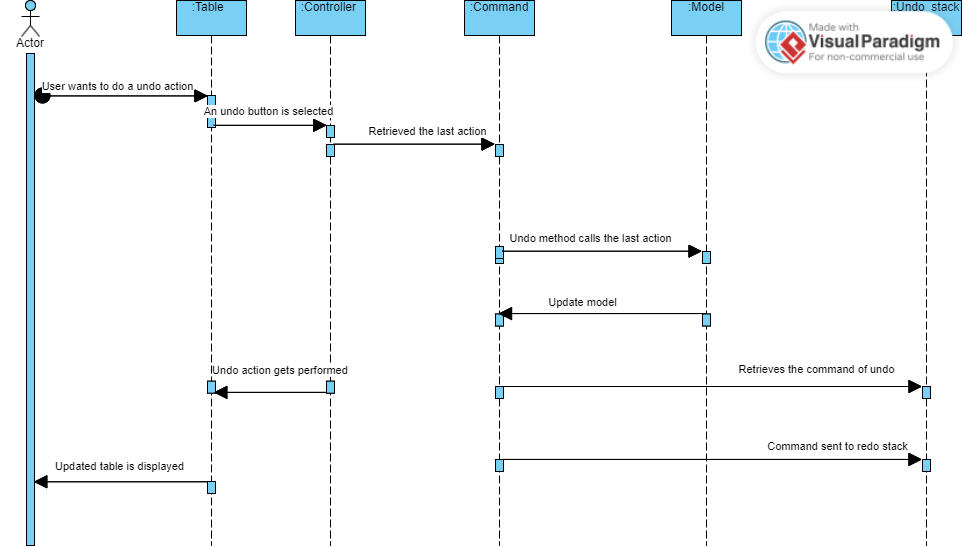
**1.Add**

****

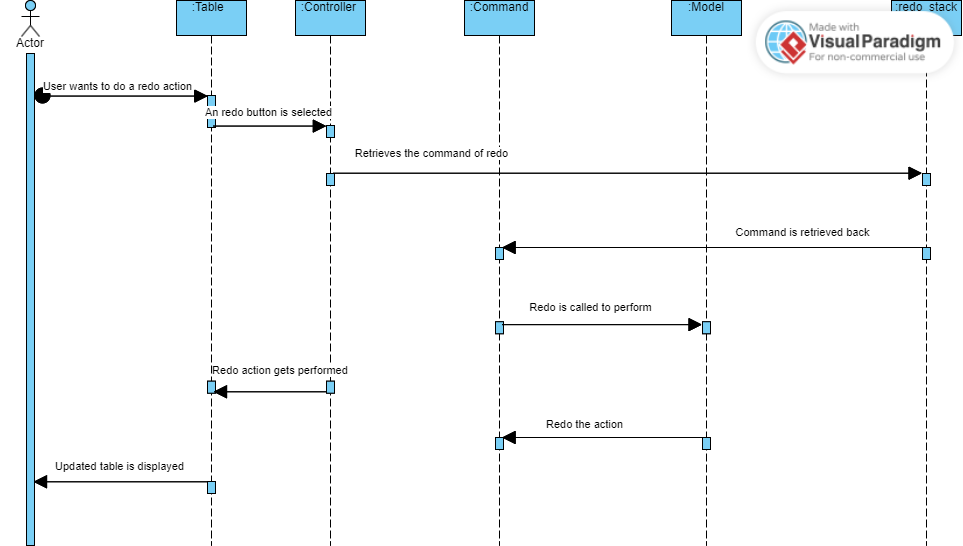
**2.DELETE**

****

**3.UNDO**

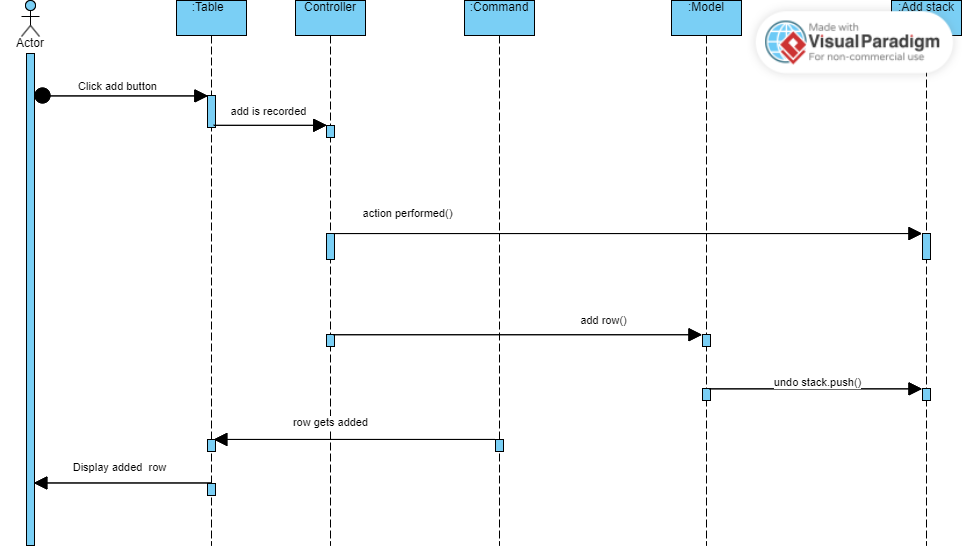
****

**4.REDO**

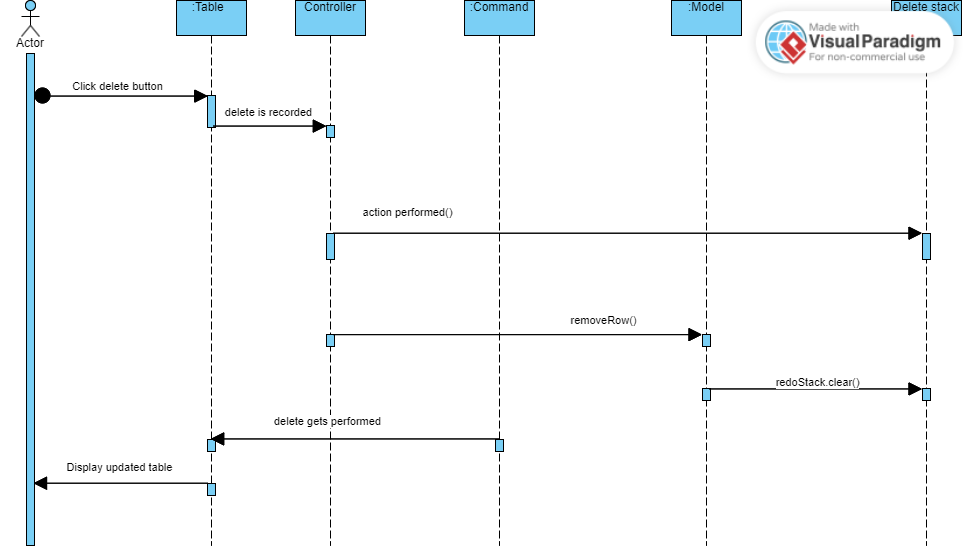
****

**5.Design Sequence diagram:**

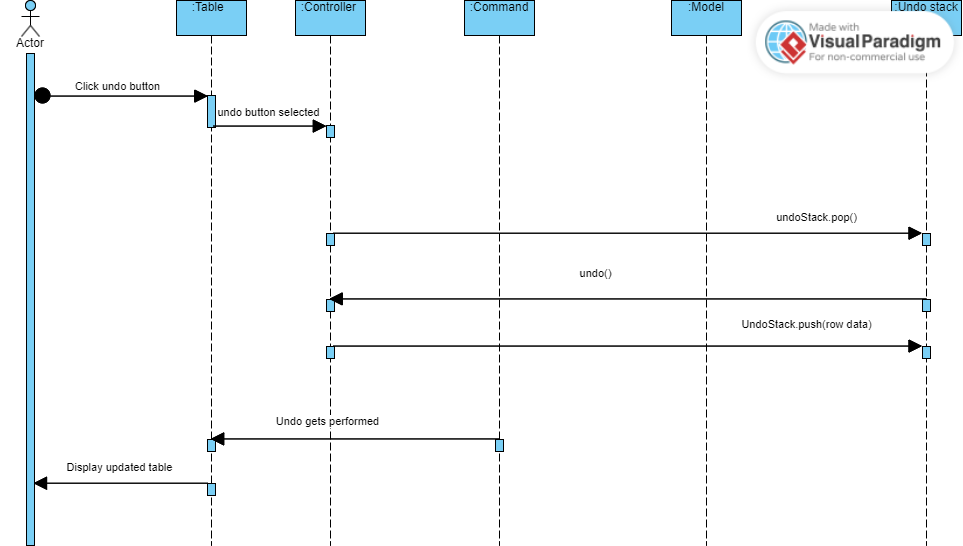
**1.Add**

****

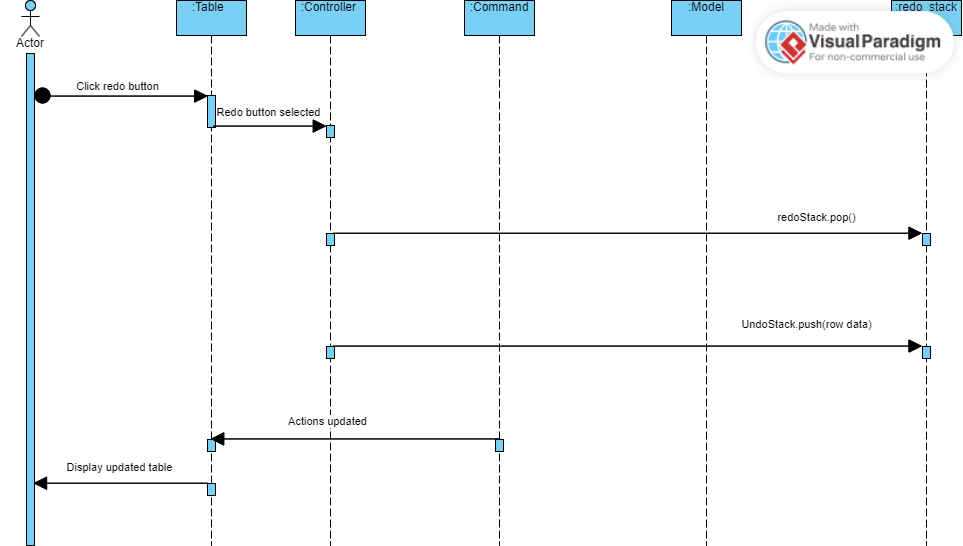
**2.Delete**

****

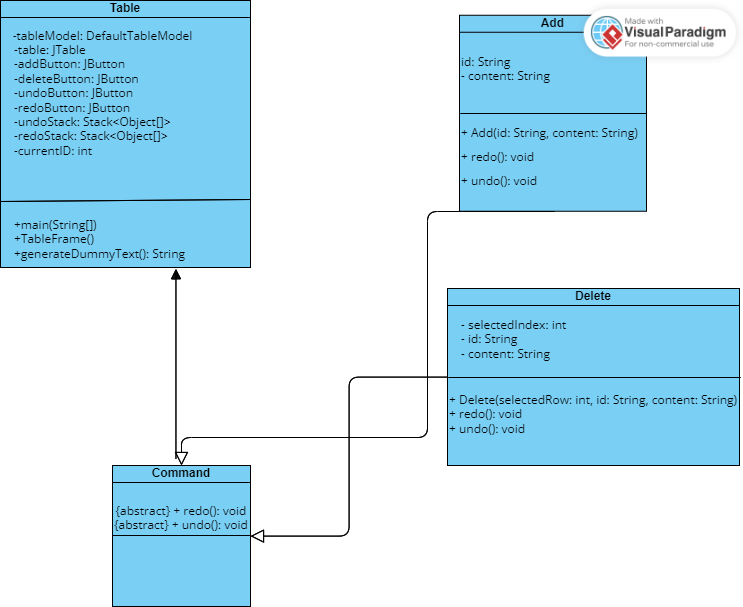
**3.Undo**

****

**4.Redo**

****

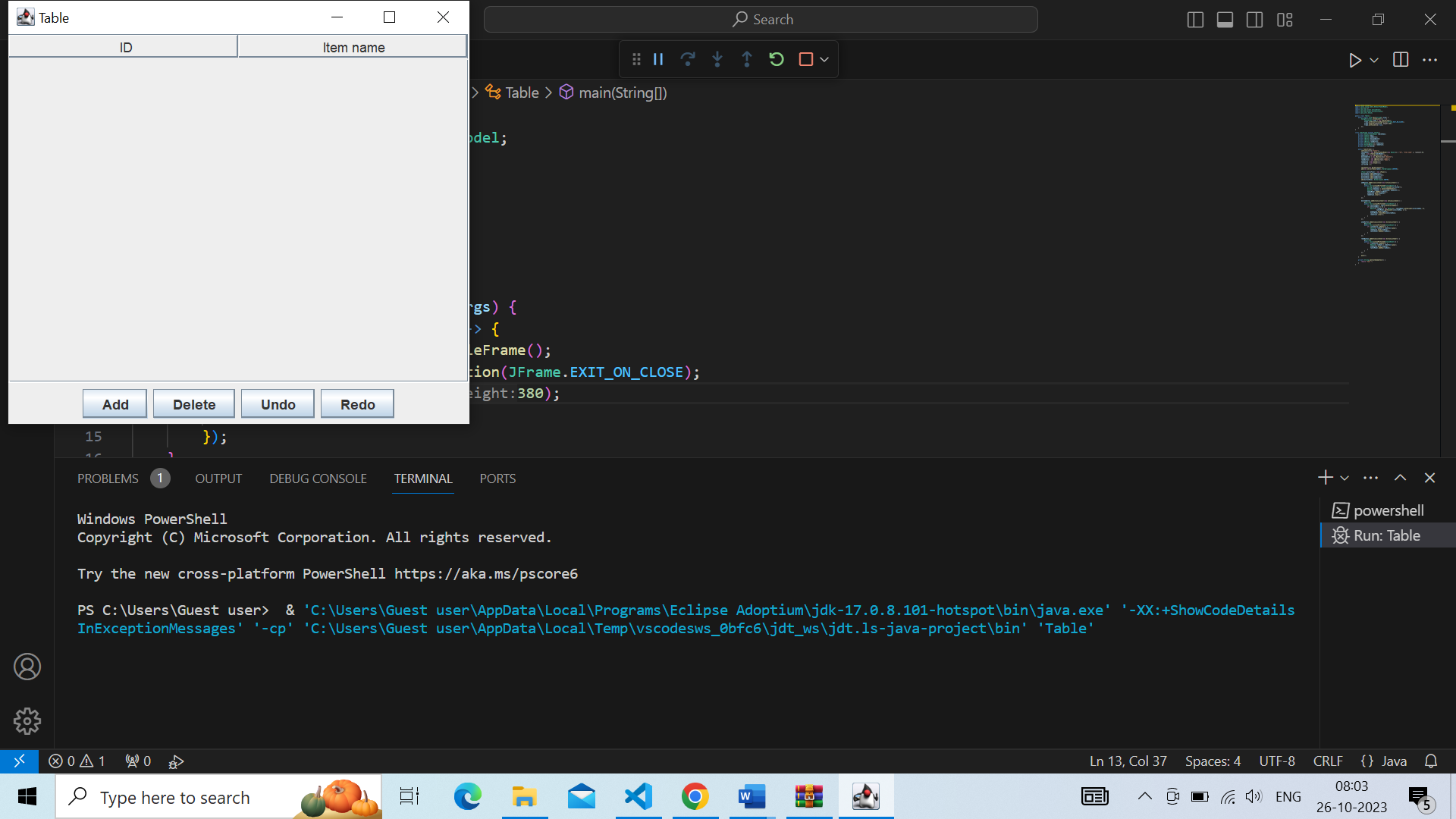
**6.Design Class diagram**

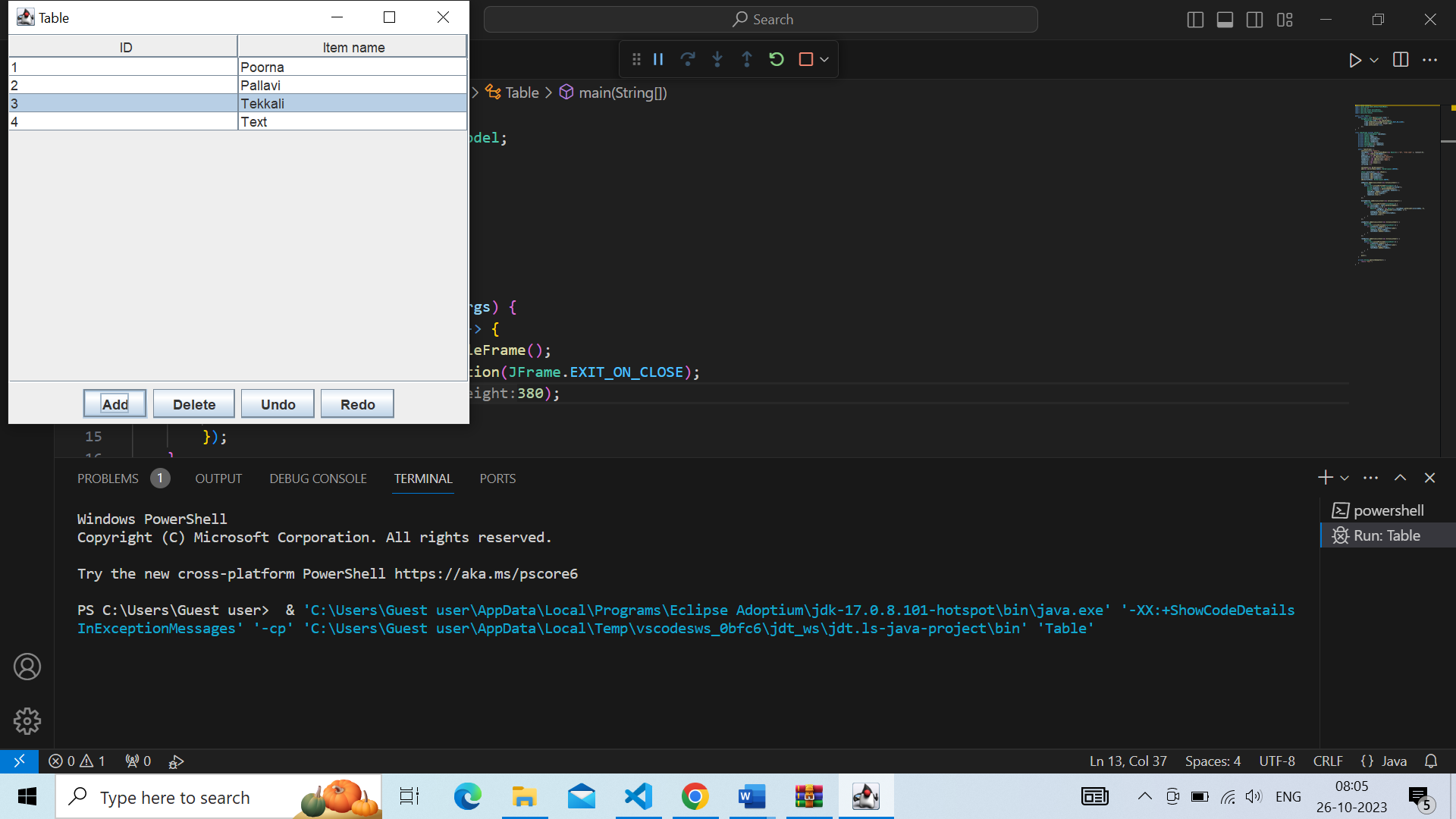
****

**7.Design in Java**

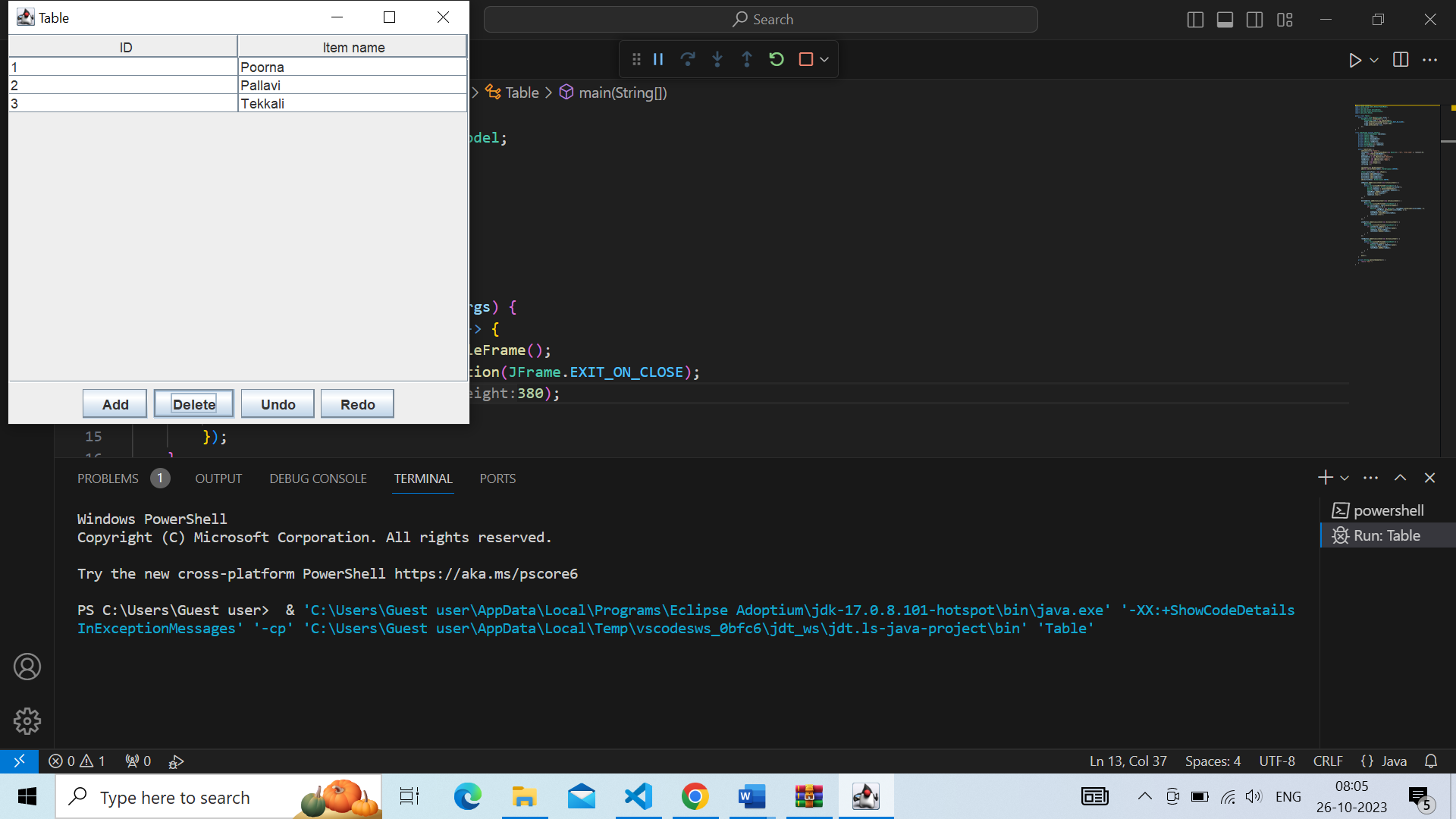
**Screenshots:**

**ADD**

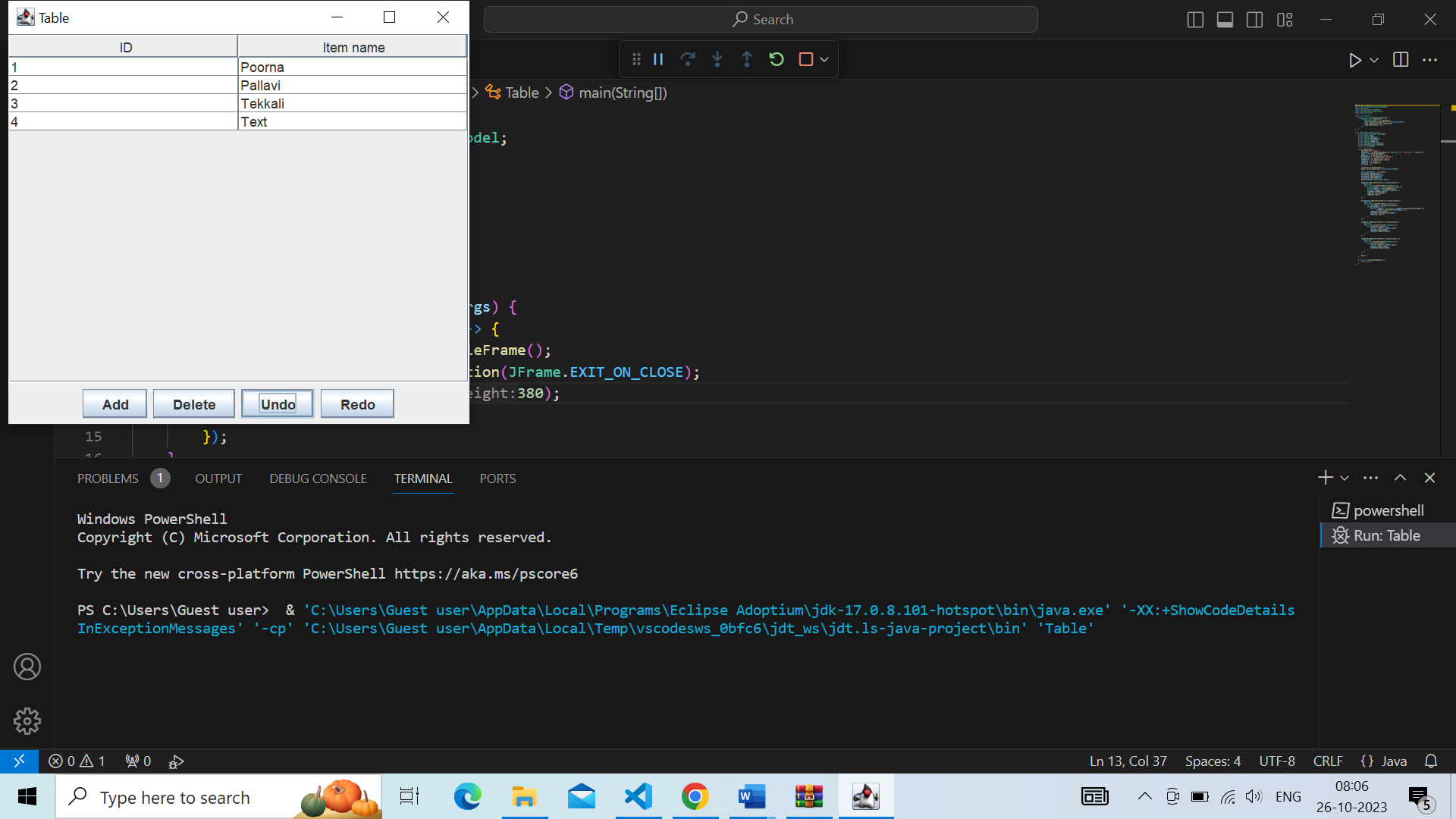




**DELETE**



**UNDO**



**REDO**

