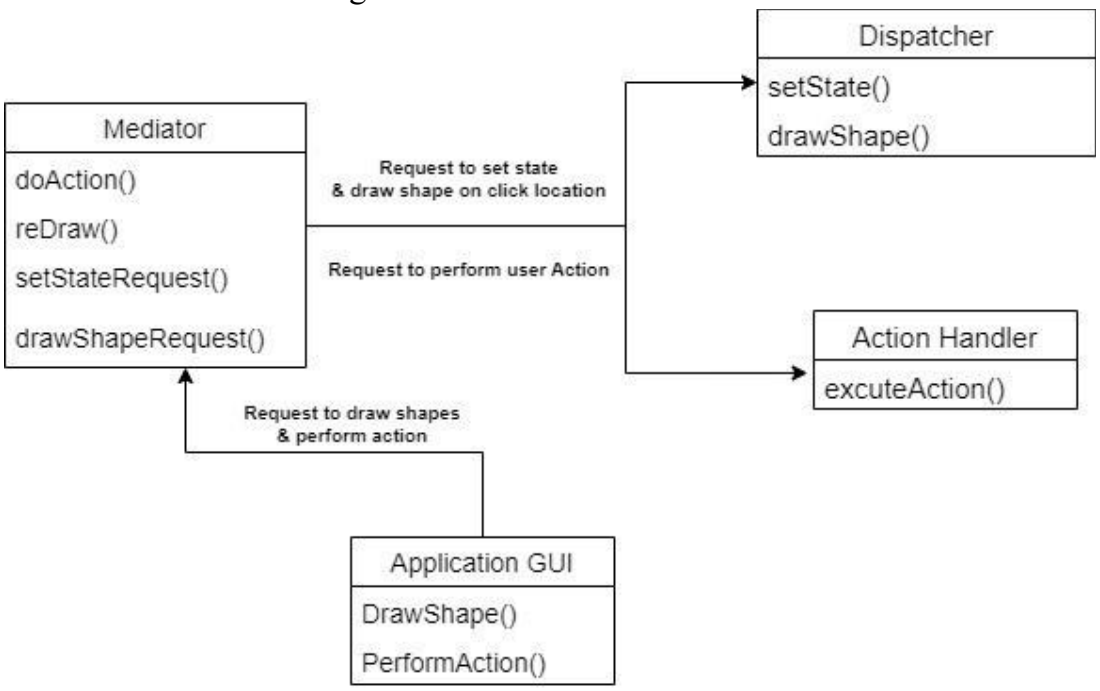
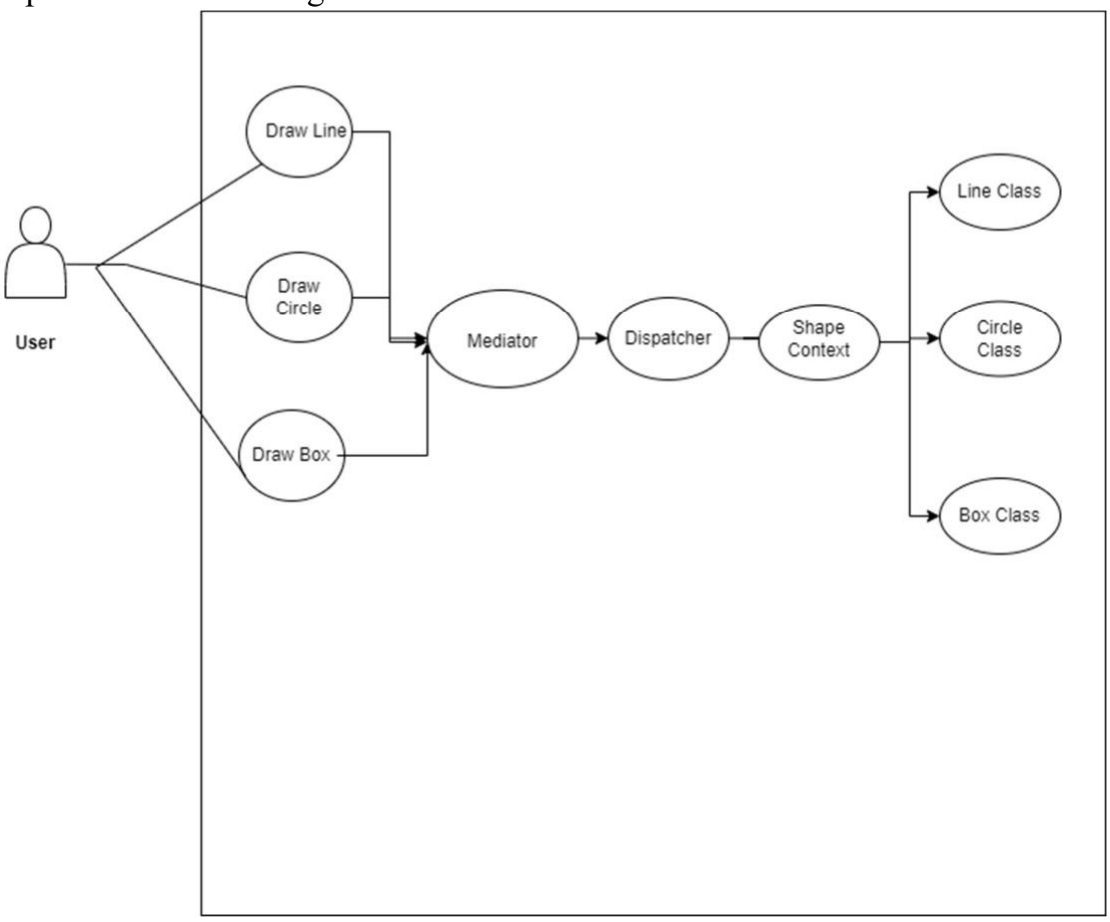
Software Design Patterns

1. Domain model class diagram



2. Expanded Use case diagram



3. Nontrivial steps from expanded use case

-> Graphical User Interface gives us the option to choose the Line, Box and Circle. Also provides the options for undo and redo.

-> User TUCBW chooses the given options on the User Interface to draw Box, Line and Circle.

-> System gives the action corresponding to the button pressed.

-> User clicks the shapes present (Line, Box, Circle) and interacts with the canvas present.

-> The GUI provides the shape chosen and draws on the canvas where the user clicks on the canvas.

-> If the user is not satisfied with the diagram, choose the undo button.

-> The previous action created by the system is deleted.

-> User chooses the redo button.

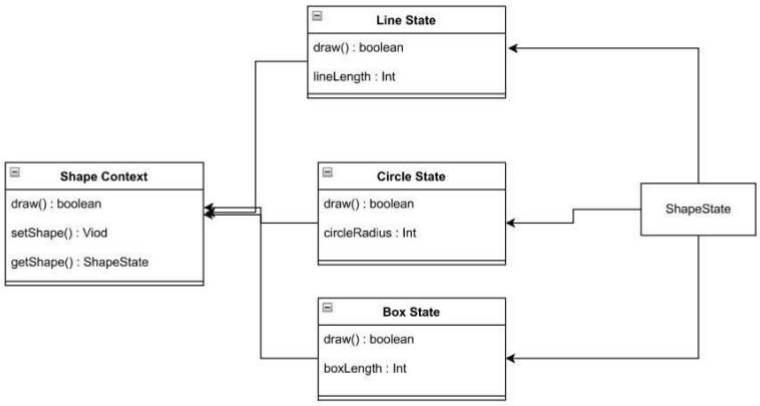
-> The deleted action is undone, and the previous action is restored.

-> User repeats the process until he gets the desired result.

-> System provides the same functions while the user interacts with the GUI ->

User TUCEW clicks the close button on the GUI.

4. State Diagram



5. Converting the nontrivial steps to scenario's

1. User TUCBW chooses one of the options shown on the GUI, such as line, box and circle.

    1.1 Clicks on the Line button and clicks on the canvas where they desire the diagram to be.

        1.1.1 the paint GUI reads the option selected and uses the controller pattern to produce the line on canvas.

        1.1.2 The paint GUI helps the user to generate multiple lines by using iterator pattern on the canvas.

    1.2 Clicks on the Box button and clicks on the canvas where they desire the diagram to be.

        1.2.1 the paint GUI reads the option selected and uses the controller pattern to produce box on the canvas.

        1.2.2 The paint GUI helps the user to generate multiple boxes by using the iterator pattern.

    1.3 Clicks on the Circle button and clicks on the canvas where they desire the diagram to be.

        1.3.1 the paint GUI reads the option selected and uses the controller pattern to produce circles on the canvas.

        1.3.2 the paint GUI helps the user to generate multiple circles by using the iterator pattern.

2. After performing the draw action successfully, the paint GUI provides the User with undo option to perform.

    2.1 The user chooses the undo option from the options given.

    2.2 The paint GUI sends the action to the controller and helps to execute the undo operation.

2.3 When the undo option is evoked, previously performed action will be removed on the canvas.

2.4 User may invoke the undo option until he removes option which he performed earlier.

3. The paint GUI have the option redo after the undo, to restore the undo option performed.

3.1 The user chooses the redo option if he wants to restore the previously deleted item.

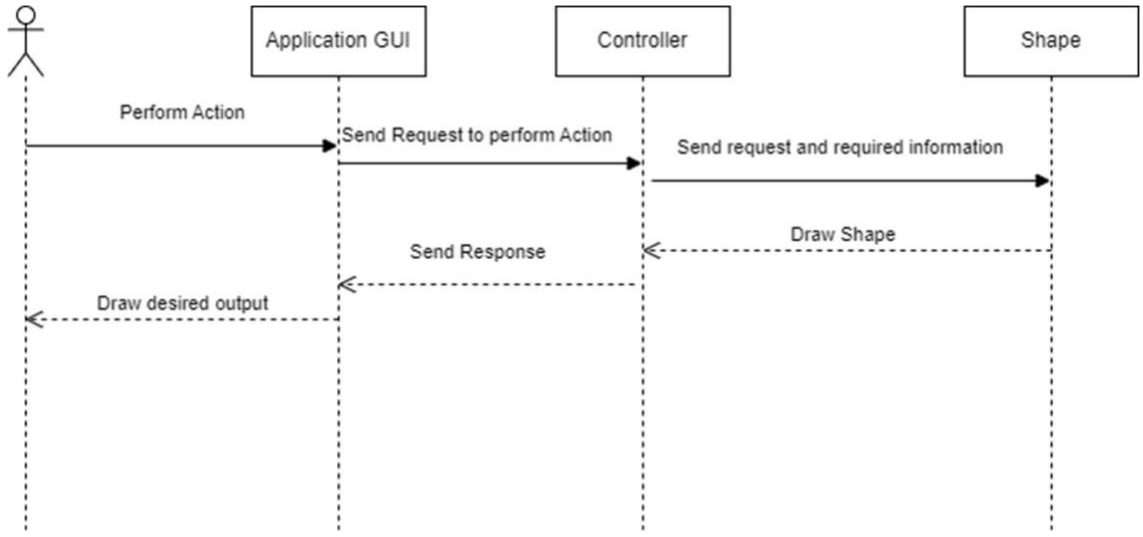
3.2 The paint GUI sends the action to the controller and helps to execute the redo operation.

3.3 When the redo option is evoked, the diagram will be restored if the undo option is performed in step 2.

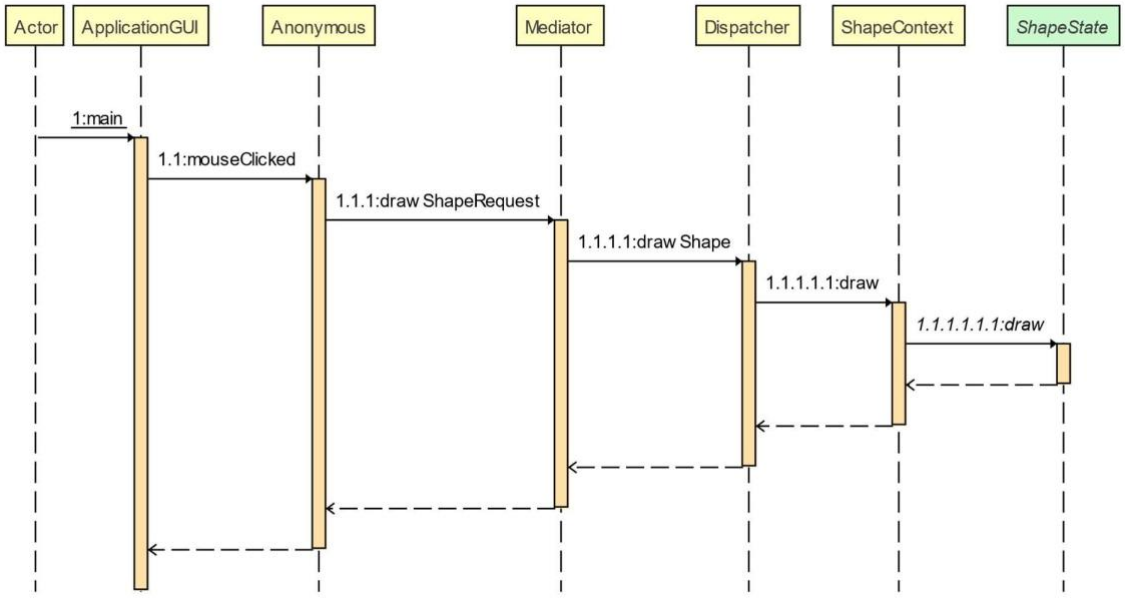
3.4 User may invoke the function until he gets the required output.

4. The user will perform step 1.1, 1.2, 1.3 until he gets the required diagram and manipulates by using step 2 and 3. 5. User TUCEW closes the paint GUI once satisfied.

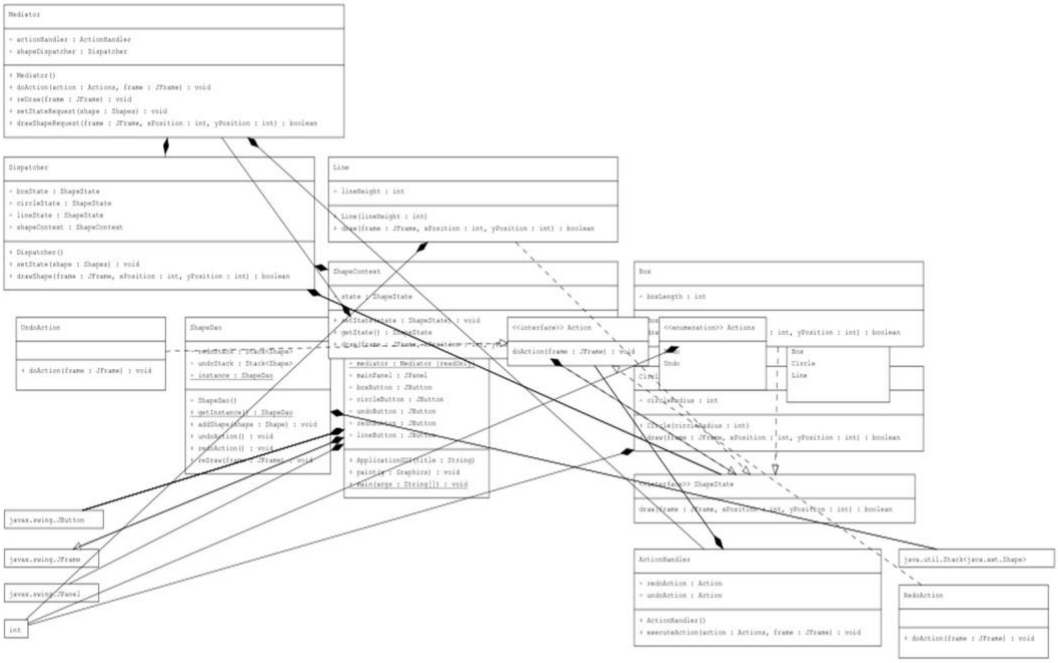
6. Informal Sequence Diagram



7. Sequence Diagram



8. Design Class Diagram

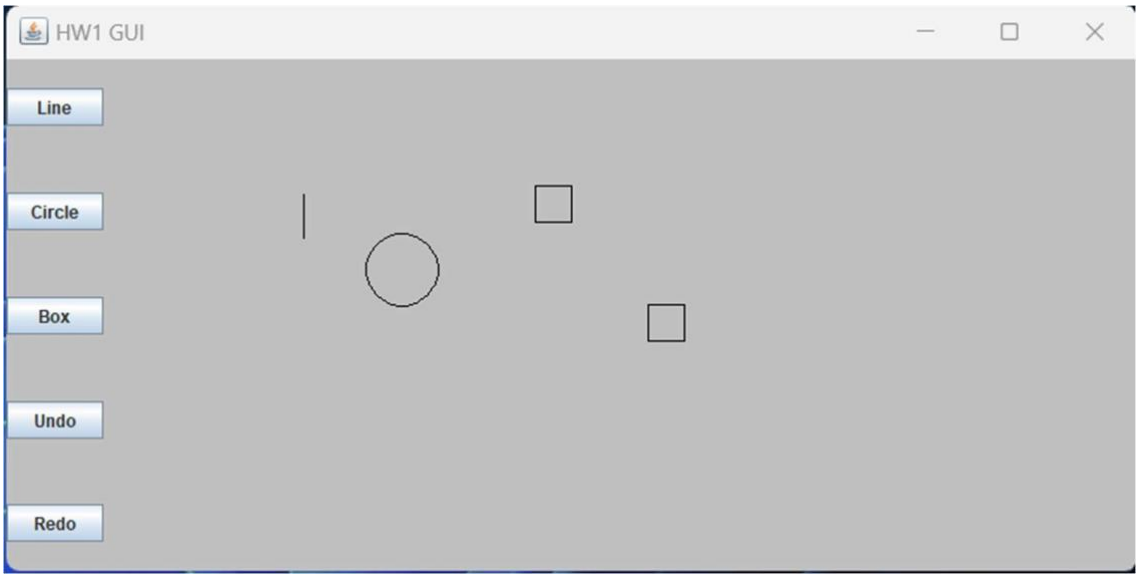


9. Jar File

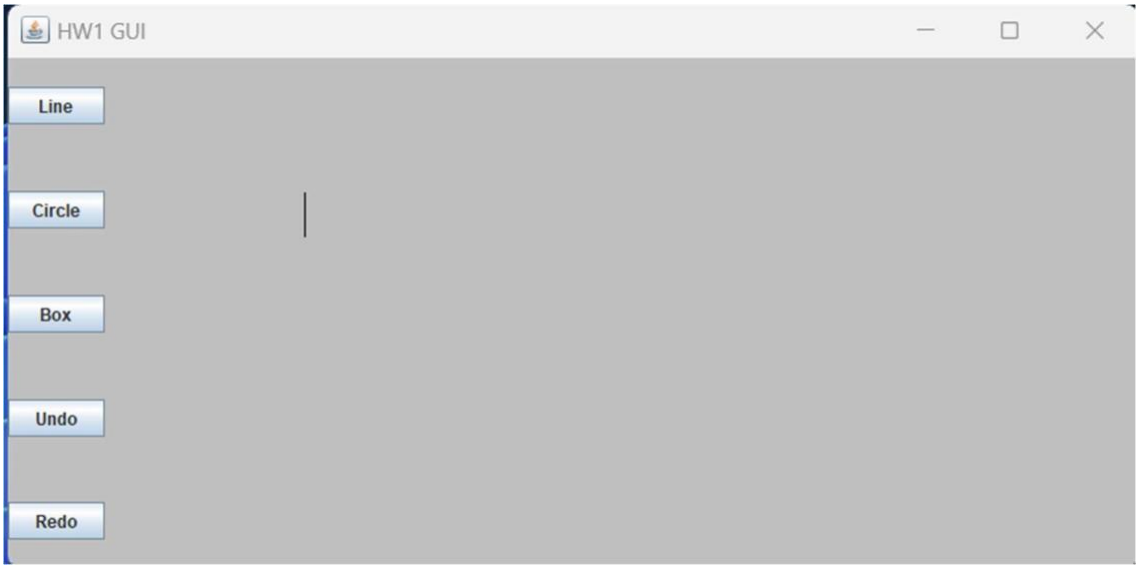
-> It is available in zip folder.

10. Output Image

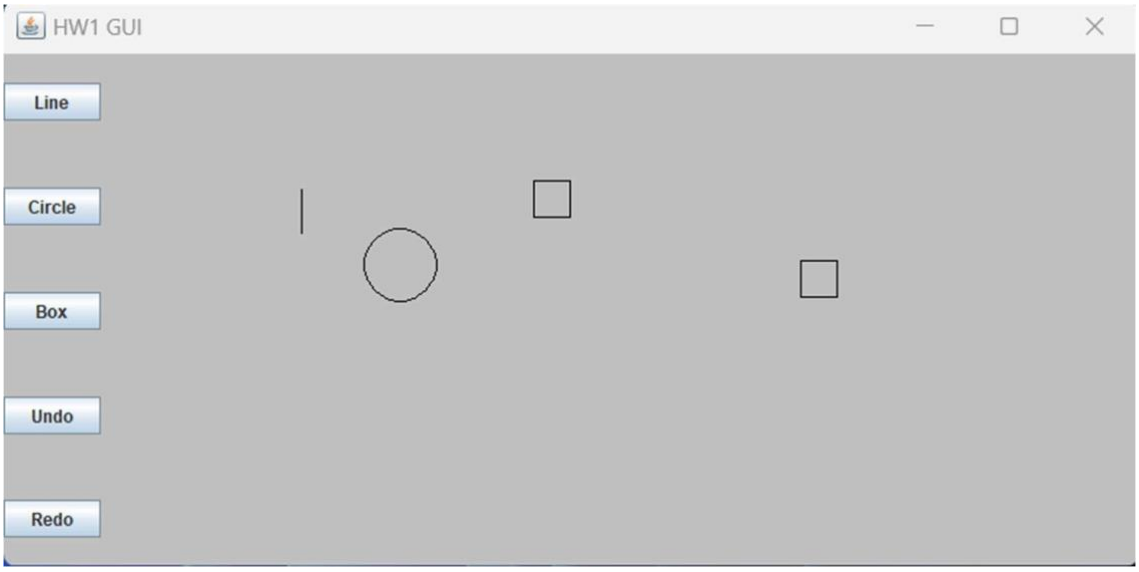
(a)



(b)


HW1 GUI
Line
Circle
Box
Undo
Redo

(c)


HW1 GUI
Line
Circle
Box
Undo
Redo