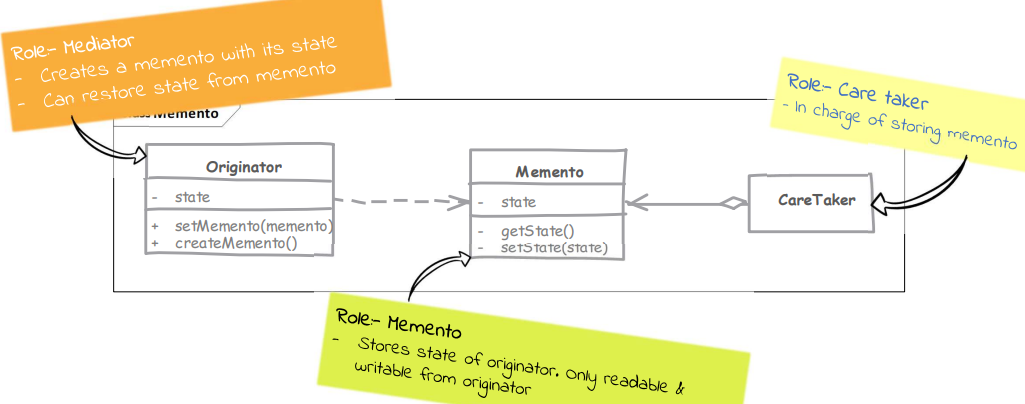
Memento

Behavioral Design Pattern

**What is a Memento:**

1. Store object’s state without exposing internal details about the state.
2. The main intent behind saving a state is often because we want to restore the object to a saved state.
3. Using memento, we can ask an object to give its state as a single, “sealed” object & store it for later use.
4. Often combined with Command design pattern to provide undo functionality in application.

**UML:**



**Implementation steps:**

1. Find originator state which is to be stored in memento. Let’s you save and restore the previous state of an object without revealing the details of its implementation.
2. Implement the memento with requirement that it can’t be changed & read outside the originator.
3. Originator provides a method to get its current snapshot out, which will return an instance of memento.
4. Another method in originator takes a memento object as argument and the originator object resets itself to match with the state stored in memento.

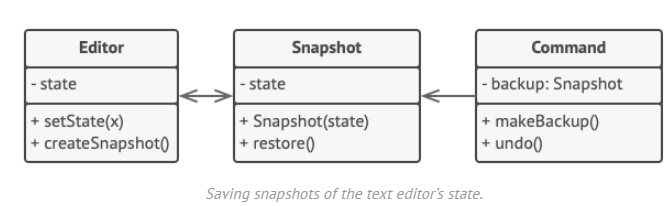
**Classes:**

1. Memento – contains state of an object to be restored.
2. Originator – creates and stores the states in Memento objects
3. Caretaker – responsible to restore object state from Memento.

**Implementation Considerations:**

1. Keep track of the size of state stored in memento. A solution for discarding older state may be needed to handle large memory consumptions scenarios.
2. Memento often end up being an inner class due to the requirement that it must encapsulate all details of what is stored in its instance.
3. Resetting to previous state should consider effects on states of other objects.

**Structure:**



Memento pattern for storing snapshots of a complex text editor’s state and restoring an earlier state from these snapshots when needed.