Abderahman Haouit
19404086

In assignment 3 I implemented a command user pattern which was very complex and confusing, somehow I couldn't get the redo function to work but the undo was working just fine. In this assignment ,assignment 4 I done a memento design pattern where it was as hard and was quit straight forward , it did take me some time to get it working but at the end I managed to get the undo and redo functions to work perfectly of course using a bit of johns code and mixing it with mine as john's example wasn't fully functional to where the assignment requirements where meant to be. Therefor the Memento pattern worked best for me simply because it used less code and rather than dealing with user commands a simple copy of a list<Shape> was made and using the memento algorithm I was able to read the second last element and bring it to life which corresponded to the success of my undo/redo functionality. My solution works alongside the algorithm of the memento where the showHistory() method is used to extract the second last element of the copy List and still be able to go back in and redo(bring back to life) the last element which wasn't deleted from the copy List was deleted from the original. This was done by the Caretaker as it knows why and when the Originator needs to save and restore itself, while the originator knows how to save itself only and then the Memento class (the lock box in which the Originator writes and reads, and which the Caretaker guards). In other terms in my code The command pattern isn't always associated with the ability to undo (though they like using them). A command is essentially a single method wrapped up in an object, along with the arguments used to invoke it. Anyone with access to the execute interface can run the command after it has been set up. The memento, on the other hand, just holds information that represents the current state of the originating class, without disclosing any state specifics to the caretaker who keeps the memento.
Undo with memento ,undo by restoring a snapshot of the originator's state.
Undo with commands ,undo by executing compensating actions.