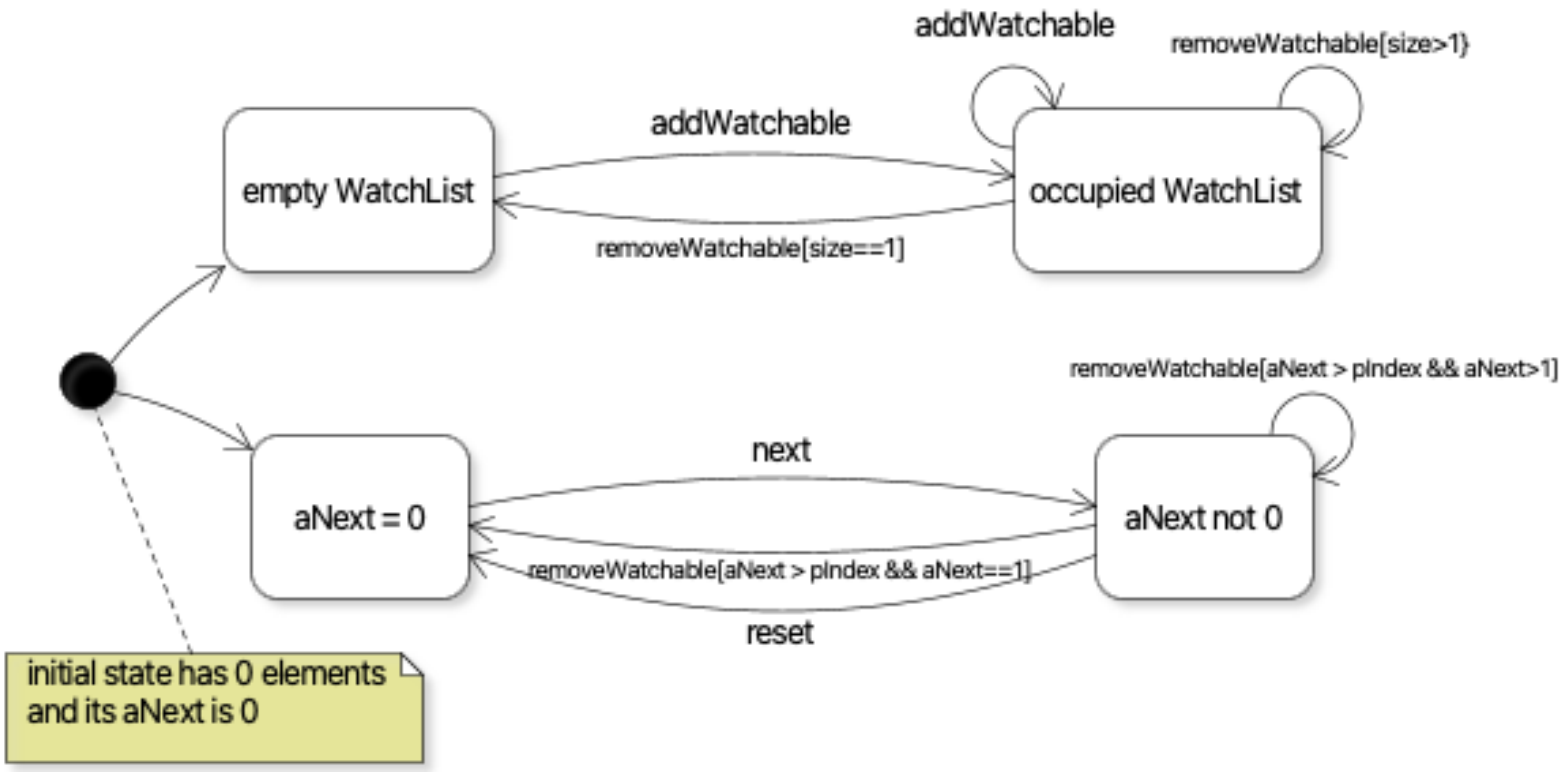
COMP 303 Neyir Erdeser
Assignment 5 260736004

Part 1:
I used OBSERVER design pattern to achieve this. WatchList class is the Observer class and it observes Watchable objects. Whenever a Watchable is added to a watchlist, that watchlist get added to the list of observers for the watchable. Everytime a Watchable is watched, it notifies all of its observers, so the Watchlist(s) related to the watchable know that, that is the last watched watchable within their elements.

Part 2:
Making use on TEMPLATE design pattern here, I have made an abstract watchable class that implements Watchable, instead of all watchables implementing it. It creates a skeleton by either implementing the methods itself or declaring them as abstract methods. The Concrete watchables then inherit, override, or implement these methods as they need them. I did pay attention not to break any rules of Liskov Substitution.

Part 3:
I used COMMAND design pattern, I have a Command interface and an AbstractCommand class that implements it. The concrete command classes (setName, add/remove watchable, next/reset) then inherit and override the Command methods. In WatchList, there are now two stacks keeping track of executed commands and redo/undo methods that use Command objects and methods to undo and redo executed methods by the client.



I displayed Watchlist contents and next as separate states, while in fact the states would be permutations of them, for clarity. Intial state has both an empty aList and aNext=0.

Neyir Erdeser
260736004