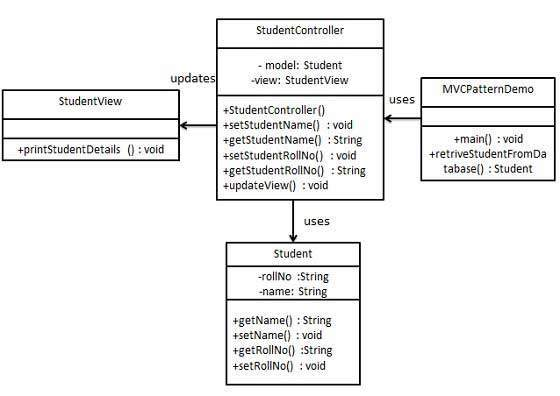
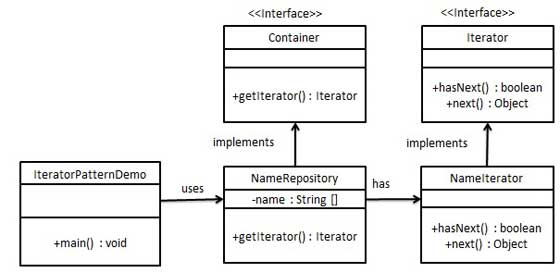
Design Patterns for Assignment 6

**Model View Controller Design Pattern Implementation:**



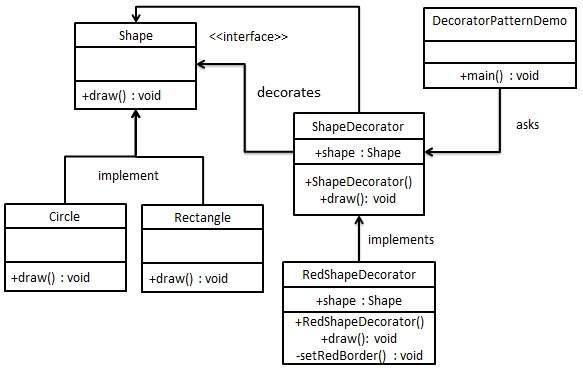
The general usage of this program already contains a very simplified version of the MVC design pattern as illustrated by the overall functionality of the program itself. When analyzing the way that MVC pattern design works we can clearly see that this program is a dedicated design based off this pattern. The Model in this context being the Tweet and Abstract tweet class respectively which is then accessed by the controller which in this program can easily be classified by Tweet Handler class. This class in turn provides all access and changes done to our model by creating databases and lists from the model as well as classifying and then searching said items created from the model. Our view in turn is essentially the GUI which hides all back-end processes through a nice and clean interface that only the user interacts with, which is itself described as a self-proclaimed controller. In turn the program already executes a fine example of the Model View Controller design pattern using the three main classes in the program, TweetHandler, Tweet, and MainApp.

**Iterator Design Pattern Implementation:**



The iterator design pattern is something that allows the program to create a traversal object to iterate through a set of data without having to understand Data Types by said object list. While normally many classes are needed to provide usage of the Iterator design pattern a simpler way is defined upon research. In types of for loops we come across something known as the For-Each loop or Advanced For loop. This in turn reflects the Iterator design pattern which due to the multithreading used in our Classify Tweets function used in both the main app as well as Tweet handler provides us with a simplistic usage of the Iterator design pattern. In the Main app starting at lime 351 we can see th e usage of Advanced for loops for traversing the created database Array List to individually classify each tweet pushed into the List. While there are plenty of other ways to implement this efficient method of traversal this was the most useful implementation of the Iterator design pattern.

**Decorator Design Pattern Implementation:**



The final design pattern that was implemented was the Decorator class. In turn this is implemented already by symbiotic relationship between Abstract tweet and Tweet itself. The idea of this design pattern is that the main class description is wrapped up in a separate or “Decorator” class that extends or implements the main class allowing for extra decorating or editing of the main class through the decorator without sacrificing efficiency. For the sake of time efficiencies sake, a Tweet decorator class was added to the program and then in the classify tweets in the TweetHandler the decorator was called instead of the main tweet class. Effectively wrapping the main class yet again and increasing the efficiency in what is called. This was the final design pattern implemented and in turn through relooking at the program it was already technically implemented in a sense due to the ideal of the Tweet class being the decorator.