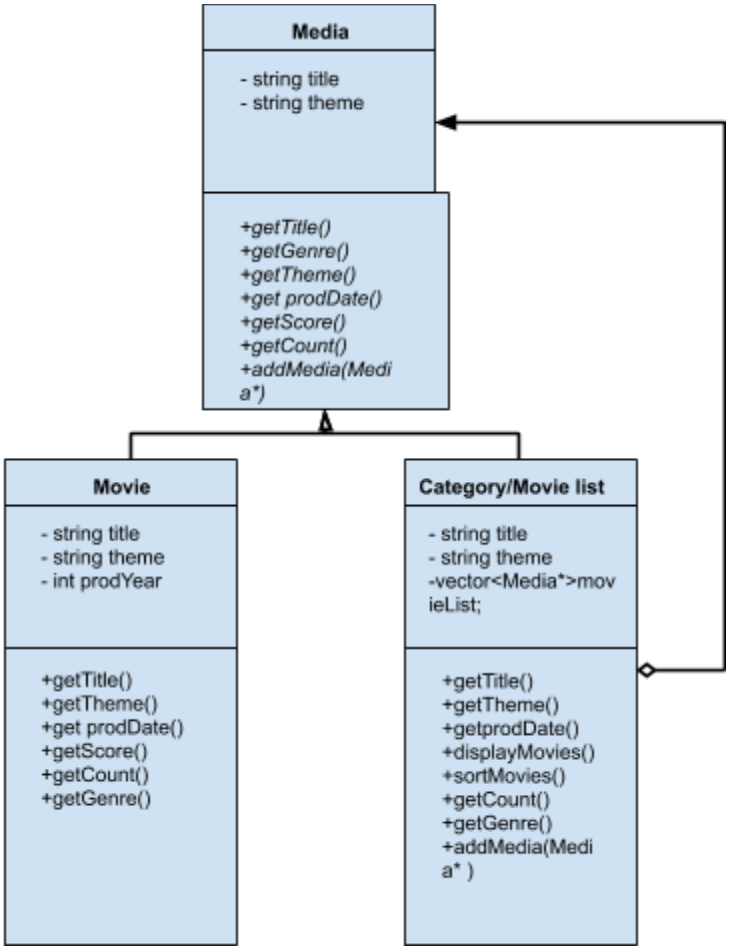
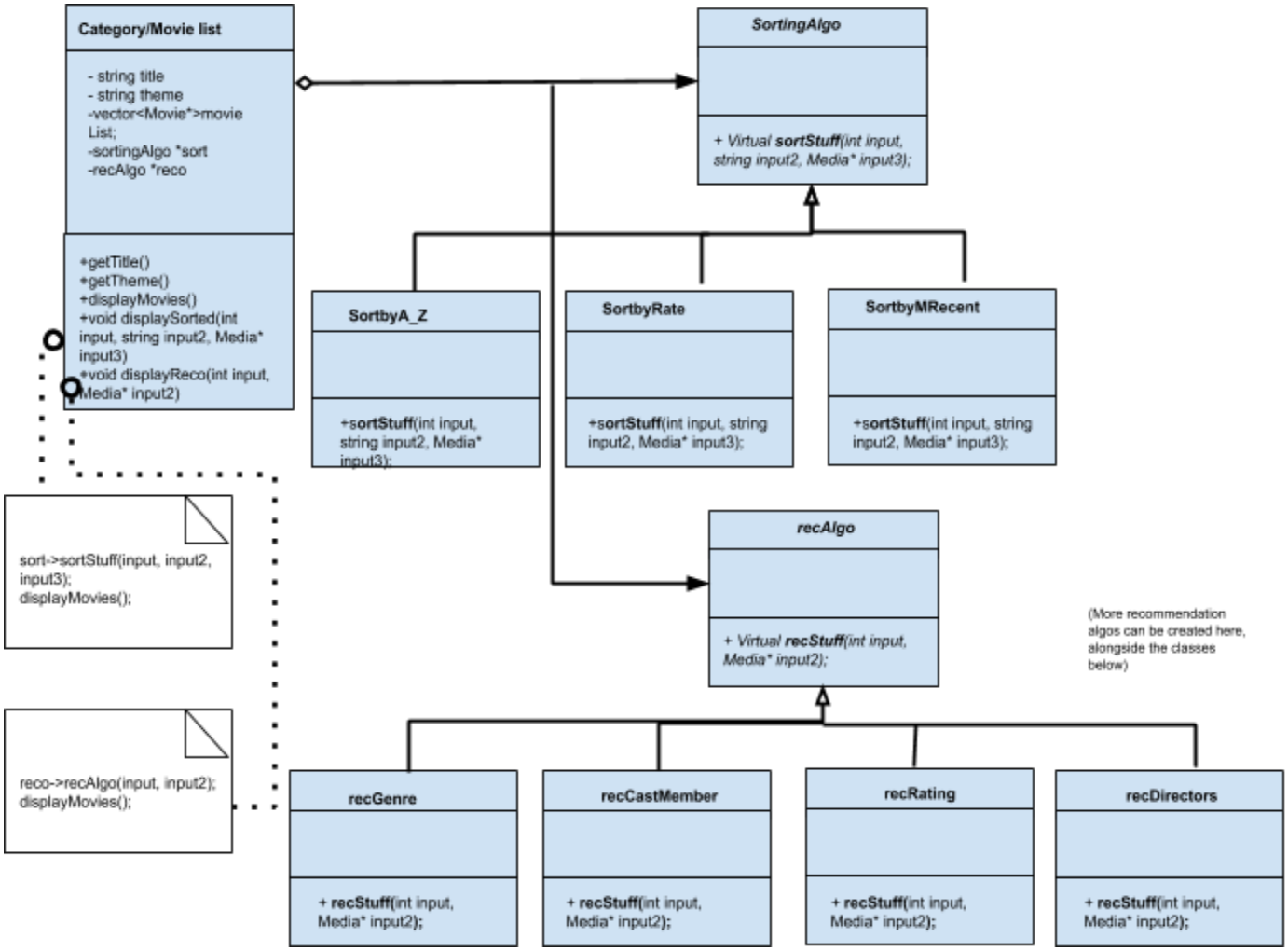
Composite Pattern:



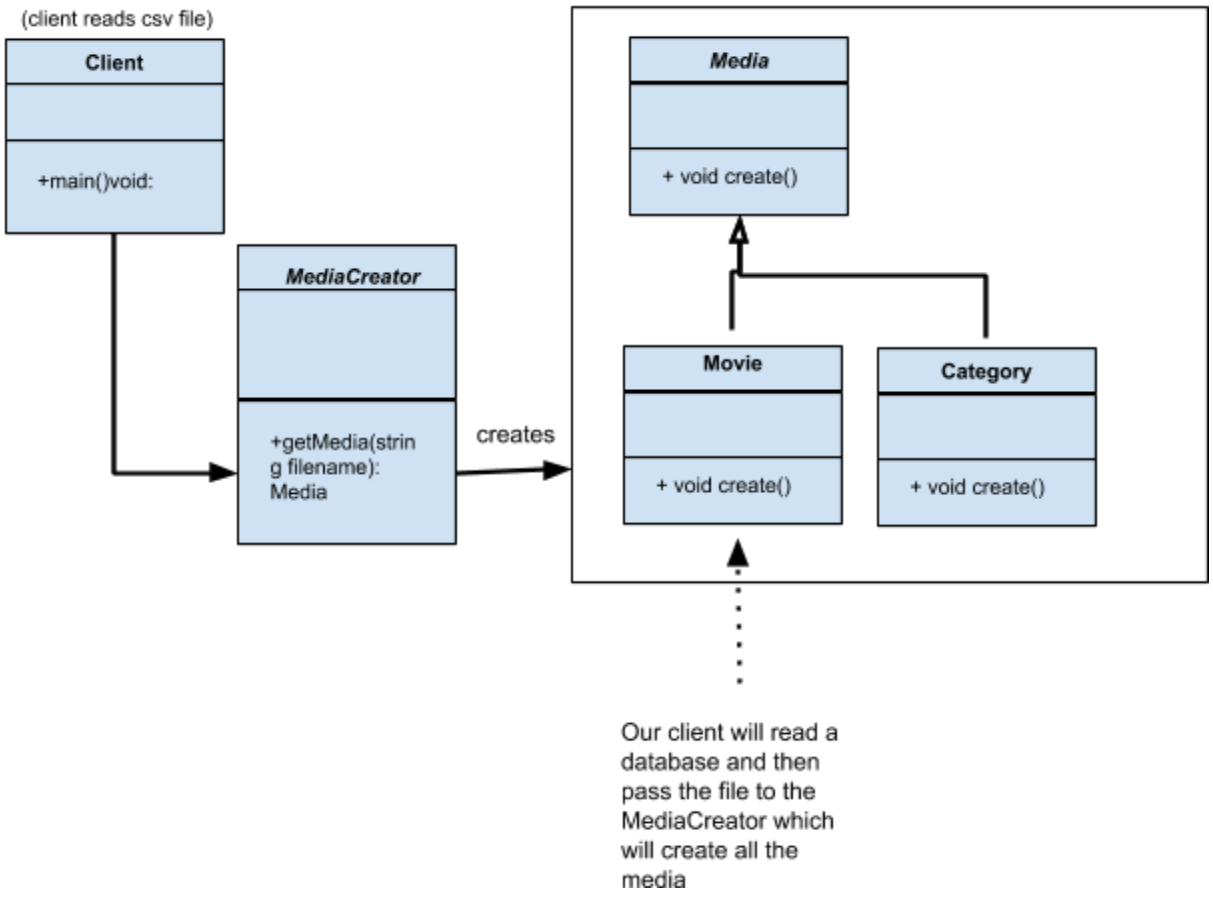
Composite pattern best fits our development model since we will be having leaf object movies that can stand alone or within greater categories or themes(christmas, comedy, etc). These greater categories will be the composites of the component and able to create more leaves through inheritance. These leaves will store the movie title, theme, year of release, rating score, and genre. Due to this composite pattern, we will have a very organized structure of the movie objects allowing for the ease of search and hierarchy structure.

Strategy Pattern:



Since our project deals with the usage of multiple variables, as a way to manage and deal with all these variant classes, we chose to use the strategy pattern as one of our design patterns. The main way we are going to find recommendations is through sorting, but since not all the sorts are not the same, we have different sorts based on the way the user wants the movie to be recommended. This allows us to recommend based on numerous different categories, allowing a more diverse and precise way to recommend the movie the user wants.

Factory Design Pattern:



Our movie project uses a lot of similar objects with very similar features. Thus, we decided to use the Factory Design Pattern, which would enable us to create a lot of movie and category objects, in which each movie would be identical to others, aside from the different values for themes, actors, and other variables. The client would get the file name of the .csv file which contains all the data of the various movies. It would then pass this filename to the media creator which would then create the various objects that our program needs, which each instantiate a new object happening through the MediaCreator (essentially the “factory” in this design pattern).